Study of Novice Programming: Plans, Object Design, and the Web Plan Object Language (WPOL)

by

Christina Schweikert

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy,

The City University of New York

2008

This manuscript has been read and accepted for the
Graduate Faculty in Computer Science in satisfaction of the
dissertation requirement for the degree of Doctor of Philosophy.

	Dr. Danny Kopec
Date	Chair of Examining Committee
	Dr. Ted Brown
Date	Executive Officer

Dr. David Arnow

Dr. Paula Whitlock

Dr. Frances Bailie

Supervision Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Study of Novice Programming: Plans, Object Design, and
the Web Plan Object Language (WPOL)

by

Christina Schweikert

Advisor: Professor Danny Kopec

Programming has evolved through a number of different paradigms, including the Object Oriented Paradigm, which aims to make programming more efficient and better understood. Despite various enhancements in programming languages, environments, and pedagogical approaches, novices are still faced with many challenges when learning to program, particularly the additional layers of abstraction presented by OOP. Computer science departments in universities worldwide are in a crisis of high dropout and failure rates, as well as low enrollment in computer science courses. This work seeks to capture the way expert programmers represent programming knowledge and visualize this knowledge for novices to enhance their learning of programming in the object oriented paradigm. It has been shown that experienced programmers utilize plan representations to encode programming concepts and tasks. Studies of novice programmers reveal that most major errors are a result of incorrect plan integration and misconceptions related to objects, such as correct object representation and incorporation of OOP

concepts into problem solving. A Plan-Object learning paradigm that reinforces concepts of object design through plan representation can aid students' ability to design and implement objects, as well as increase their ability to utilize objects into problem solving. Web Plan Object Language (WPOL) is an online learning environment that utilizes the Plan-Object approach with three phases of learning: plan observation, integration, and creation. To characterize novice difficulties with problem solving and object design, new error categories are developed. An empirical study is conducted to measure novice's performance on a sample case study involving objects and problem solving. Due to the visual experience of Plan and Object design, integration, and implementation, the online course exposed to the Plan Object Paradigm and WPOL exhibited fewer errors related to plans and OOP. Student programs also reflected better understanding of correct object representation and incorporation of plans and objects into a solution.

Content

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Problem Description

Society is becoming increasingly dependent on computer software in all aspects of life such as in medical, transportation, financial, and security sectors. The National Institute of Standards and Technology (NIST) reports that software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated $59.5 billion annually. In addition, 80% of development costs are spent on identifying and correcting errors [NIST 02]. Despite the study of programming languages and paradigms for over fifty years and development of various paradigms (structured, procedural, functional, object-oriented) and environments, programming is still considered a difficult task. Every programmer is a novice at some point and it is necessary for novices to have the proper foundation and understanding of programming to produce correct and sound software that can easily be maintained and modified. Concerns have been raised that the resulting skill set produced by current trends in computer science education is insufficient for today's software industry, particularly for safety and security purposes [Dewar and Schonberg 08]. The increasing use of outsourcing also serves as a motivation to provide a better foundation for programmers who will be of greater value and will not be easily replaced. By incorporating the concept of plans in programming,

problem solving ability and the chances for implementation of correct programs are increased. Improved environments for learning programming will contribute to the effective teaching and learning of programming.

Learning object-oriented programming is a challenging task for many beginning students and there is much debate about how objects should be taught. The shift to Java and other object-oriented languages in CS1 courses has generated much controversy among college faculty on how and when objects should be taught. There has also been some backlash in regard to teaching Java as the first language, such as industry complaints and poor student performance that have caused many departments to switch back to C++ [Dewar and Schonberg 08]. A discussion among members of SIGCSE on this subject brought up issues such as: weaker students having a very difficult time dealing with the additional layers of abstraction resulting from the use of objects and dropout rates in introductory courses as high as 30 or 40% in some cases. The debate indicates that the object oriented paradigm requires a new way of thinking and teaching. Bjarne Stroustrup, designer of C++, has stated that it takes a proficient C programmer a year to 18 months to become proficient in C++ [Bruce 05]. Studies indicate that approximately 30% of students in the US and UK do not understand the basics of objects [Wei 05, McCracken 01]. Educators and students have expressed difficulties with regard to strategies for introducing and applying object oriented concepts in early programming courses. The challenge of effectively and meaningfully incorporating objects into programming courses for

novices presents the motivation to reconsider existing approaches and try something new.

Due to its accessibility and variation of choices (customized experiences), use of the web can facilitate the learning of programming. Design and development of a new teaching and learning model that incorporates plans, objects, and the web would be valuable in effectively teaching programming to novices. Motivation for developing new teaching and learning techniques and environments is beneficial to university students, training of programmers for industry, and supplemental self-help for end-user programmers.

1.2 Research Objectives

The overall objective of this research is to gain a better understanding of the challenges facing novice programmers and to find new ways to enhance the experience of learning programming while improving novice's comprehension and programming skills. This can be broadened to any audience learning to program, such as for training in industry or end-user programmers.

Examination of novice programming errors and challenges, especially misconceptions related to objects, exposes a need for a new approach for teaching programming. A goal of this research project is development of a new learning paradigm for object oriented languages. The Plan-Object Paradigm applies the concept of plans to object oriented programming. The Plan-Object approach aims to

enhance novice programmers' ability to design, implement, and integrate objects into their programs.

Another research objective is proposal and design of an online learning environment for novice programmers. WPOL (Web Plan Object Language) is an environment that utilizes the Plan-Object approach. In WPOL, there are three phases of learning: observation, integration, and creation. The observation phase demonstrates a visual solution to a problem in terms of plans and their integrations, from description to the final program. We believe that facing current trends it will be a big accomplishment to get students to think in terms of observation and plans. Today's students are so "doing" oriented that it is a great challenge to get them to step back, be observant and introspective about the elements of a program. To improve teaching for students learning object oriented programming, it would be useful to make explicit the areas of attention and use an environment that will force object modeling as well as visualize objects and their behavior [Kaasboll *et al.* 04]. The observation identifies the objects necessary for the program and utilizes an object plan to design each object, its components, and set any relationships. The object plans are then implemented according to the plan properties. Plans are then integrated to demonstrate the rest of the solution, as well as depicting how objects are incorporated into the program. The purpose of the integration phase is to test novice's ability to correctly integrate plans for a sample solution. In the creation phase, novices can design their own plans by customizing plan properties.

In order to better characterize novice errors in their programs related to plans and object design, new error categories are developed. An empirical study is conducted to measure novice's performance on a sample case study involving plans, objects, and problem solving.

1.3 Dissertation Overview

The rest of the document is organized as follows:

Chapter 2: Study of Novice Programming and Problem Solving and Using Plans in a Web Learning Environment

This chapter provides an overview of research on programming errors, empirical studies, and environments for learning programming. The concept of plans as applied to programming is also introduced.

Chapter 3: Novice Object Design: Challenges and Solutions

Discusses difficulties novices have with objects and a debate regarding how objects should be taught. This chapter also introduces the concept of the Plan-Object learning paradigm as a new approach to teaching and learning objects.

Chapter 4: Design of WPOL (Web Plan Object Language)

This chapter describes the design of proposed online learning environment, WPOL. Examples demonstrate the three phases of learning: observation, integration, and creation.

Chapter 5: Data Analysis and Results

Presents new error categories for capturing errors related to objects and problem solving. This chapter also includes the results and samples from an empirical study comparing a traditional online programming course with one utilizing a Plan-Object approach and WPOL.

Chapter 6: Conclusion and Future Work

This chapter discusses the contributions of this research project and directions for future work.

2 Study of Novice Programming Error Patterns, Visualization of Programming, and Problem Solving Using Plans in a Web Learning Environment

2.1 Introduction

Investigation into the problems novices have with programming enables us to understand their obstacles and explore possible solutions. Most novice programmer errors are related to: plan composition errors, comprehension of Object Oriented concepts, language construct misconceptions, programming environments, and a lack of instructional assistance. Novice programmer difficulties are also attributed to textual and linear representations of programming languages, lack of problem solving skills, and the attempts to use objects in programs without a sufficient foundation. The web can be incorporated to visually represent objects, language constructs, plan integration, and program execution. Plans enhance problem solving ability and facilitate the correct implementation of programs and understanding of OOP. Web environments have contributed to the effective teaching and learning of programming.

2.2 Why Study Programming Errors and Issues?

Study of novice programmer errors helps us to better understand problem-solving strategies and enables us to improve the teaching of programming and programming languages themselves. Empirical studies have been performed to try to identify and analyze causes of novice programmer errors. Computer science educators are currently faced with the problems of students failing or withdrawing from the beginning programming courses because of their inability to assimilate basic concepts taught at an early stage [Truong 05]. Study of novice programmer performance improves both the productivity and the quality of programming, while enhancing our knowledge and understanding of the effectiveness of educational techniques used for the training of programmers [Johnson 86, Spohrer 85, Soloway 86].

Every year more than a thousand students take Software Development 1 as the first programming course at Queensland University of Technology with Java and an average of 35% of the students have failed over seven semesters. A design of ELP, an Environment for Learning to Program, has helped students by eliminating some of the unnecessary problems confronting them [Truong 05].

In his “Practical Programmer” column of Communications of the ACM, Robert Glass points out the summaries of studies on Object-Oriented (OO) approaches versus functionally focused approaches [Glass 05]. These findings are very important to computer science in general, and education of programming specifically, contradicting common assumptions. Glass states that “There are also

mixed claims about OO benefits - it is supposed to be a novice–friendly approach, but studies have shown that novices do better with functionality focused rather than object-focused approaches." The author argues that the benefits of the Object-focused approach compared to the non Object-based approach in terms of reusability are not evident; however, Glass also notes that the OO approach is not fully OO since the OO field has now embraced "use cases", a decidedly functionality-focused approach to define system requirements. The author concludes, "Once again as the hype of OO dies away, we are beginning to see clearly that it is a good but not best approach to building software" [Glass 05]. There are some that still old-school programmers that argue that: "A traditional style of software development can be just as productive as a more modern approach". [Jenkins, 06]

Continuing empirical studies on novice programmers will shed light on some of the difficulties novices are encountering with current programming issues. The study of programming languages, paradigms, and novice programming enables us to gain insights into novice problem solving and enables us to make contributions to the refinement of programming languages, environments, and teaching methods.

2.3 Novice Difficulties and Contemporary Solutions

The programming language of choice for introductory programming courses has changed over the years. Programming languages started in the 1950's with FORTRAN, which was designed for scientists and used arcane notation for novices. The trend then moved to BASIC (1967), which tried to make it simple but lost many theoretical concepts. Pascal was introduced in the 1970's as a

pedagogically designed language, but was limited in power. C++ (1988) then dominated with focus on objects, and currently many courses are moving to Java (1995 to present) due to the marketability of Internet programming. While the choice of a programming language for an introductory course may be influenced by industry demands, critical concepts can be successfully taught in a variety of languages. Integration of web languages such as HTML and JavaScript into the first course is increasing. The paper by [Kopec *et al.* 00] considers the breadth-first approach brought forth by ACM/IEEE-CS Joint Curriculum Task Force and the importance of providing a broad overview of the computer science field in an introductory course. It concludes that if students are presented with details of computer architecture and machine language as the first topics, they may be discouraged. Students benefit more from a combination of hands-on experience with programming basics as well as theoretical concepts. In the second course in computer science, focus is on further development of programming skills. Choosing a programming language for this course has been reflective of student interests and preferences (e.g. web and game programming) and of what would make students attractive to employers, such as knowledge of Java for web programming and Visual Basic. Programming problems should be practical and require the students to create a program to solve some relevant problem. This will give students hands on experience with problem solving, ability to choose from several programming approaches, understanding of language-specific issues related to solving the problem at hand, and value for programming skills [Kopec *et al*. 02]. Web-based

tools and environments have begun to revolutionize computer science education. Examples of web and programming learning environments will be discussed.

2.4 Language Construct and Semantic Errors

Comprehension of language constructs is one obstacle for novice programmers. Language constructs are the building blocks of a program and a clear understanding of language constructs is essential for writing a correct program. Novice programmers have misconceptions about the syntax and semantics of language constructs such as: differences between the loops, the default rules of a language, update of for loop control variables, how values are bound to variables, how nested ifs and loops work, etc.

A survey was conducted among faculty from 58 colleges to identify what they considered the most important Java programming errors. [Hristova *et al*. 03] The errors were categorized as syntax, semantic, and logic. Syntax refers to mistakes in spelling, punctuation, and the order of words in a program. Syntax errors are frequently identified by the compiler, but the error messages may not give the students the information needed to fix the code. Semantic errors refer to the meaning of the code, such as a mistaken idea of how the language interprets instructions. Logic errors tend to arise from fallacious thinking on the part of the programmer rather than problems with the language; however, logic errors may result in improper syntax or semantics, or from improper understanding of their effects. Some errors may fit into more than one category. Prevention of the

following errors is considered essential for a solid foundation to learning how to program effectively [Hristova *et al*. 03]:

Syntax errors:
= vs. ==
== vs. .equals
mismatching, miscounting, and/or misuse of { }, [ ], ( ), " ", and ' '
&& vs. &
| | vs. |
incorrect semicolon after an if selection structure before the if statement or after the for or while repetition structure before the respective for or while loop
wrong separators in for loops (using commas instead of semicolons)
an if followed by a bracket instead of parenthesis
using keywords as method names or variable names
invoking methods with wrong arguments
forgetting parenthesis after method call
incorrect semicolon at the end of method header
leaving a space after a period when calling a specific method
>= and =<

Semantic errors:
invoking class method on object

Logic errors:
improper casting
invoking a non-void method in a statement that requires a return value
flow reaches end of non-void method
methods with parameters: confusion between declaring parameters of a method and passing parameters in a method invocation
incompatibility between the declared return type of a method and in its invocation
class declared abstract because of missing function

Figure 2.1 – Sample Syntax, Logic, and Semantic Errors

Classifications of errors help identify a variety of types and causes of errors in many languages, environments, and at different levels of expertise. [Ko and Myers 03]

2.5 Plan Representation of Programming Knowledge

In order to gain a thorough understanding of human knowledge, it is necessary to determine what forms of knowledge exist and find out how they are applied. The idea of plans used in programming builds on the concept of scripts and plans used for structured knowledge representation in natural language processing. A script in NLP is defined as a structure that describes an appropriate sequence of events within a particular context. Scripts are predetermined, stereotyped action sequences that define a well known situation. A script is made up of slots and restricted types of values that can fill those slots. Plans account for general knowledge about new situations. Plans describe the set of choices that a person has when setting out to accomplish a goal. By finding a plan, one can make guesses about the intentions of an action in an unfolding story and use these guesses to make sense of the story. [Schank *et al*. 75].

It has been shown that expert programmers can remember programs better than novices when the programs had some meaningful structure; experts use their higher level knowledge – plan knowledge – in order to code the programs for easier

recall [Soloway *et al.* 82]. Novices lack this tacit knowledge of plans that experts have gained through experience.

2.6 Plan Composition Errors

Alternatives to construct-based programming misconceptions are plan composition errors. Novices may not detect negative interactions between sections of code that are locally correct, but globally incorrect. For example, the code to perform the output may be correct, but in the wrong place in the program [Spohrer 86].

Several researchers argue that programmers utilize plan-based representations when composing or comprehending code. Study of programmers using Pascal and FORTRAN indicates that programmers segment the program into major components which are defined by goals in a plan representation [Yu and Robertson 88].

Plan composition understanding is crucial for novice programmers to correctly put together the pieces of a program to solve a given problem. Plans are template-like solutions used in problem solving. A Plan is represented in programming as a set of code segments that together perform a task in solving a problem. Plans may be a single statement (e.g. Increment Plan), or may be more complex, consisting of several sub-plans (e.g. Sort Plan) [Ebrahimi 92].

Rainfall Problem: Plan Composition Error Analysis

Several studies of novice programmers have used the rainfall problem as an exercise to analyze and categorize the errors [Soloway 86, Ebrahimi 89]. The rainfall problem is a simple statistical program with error checking and sentinel value for the termination of input loop. Rainfall is a meaningful exercise for plan composition because it uses several plans such as Sum Plan, Count Plan, Average Plan, and Maximum Plan. It also employs a variety of language constructs such as input, output, assignment, decision making, and loop.

The following explains how the rainfall problem was presented to the students:

Write a program that will read the amount of rainfall for each day. If a negative value of rainfall is entered, the program should reject it since negative rainfall is invalid and inadmissible.

The program should print out the number of valid recorded days, the number of rainy days, the average rainfall per day over the period, and the maximum amount of rainfall that fell on any one day. Use a sentinel value of 9999 to terminate the program.

The input and output sample:

```
0.0
1.6
0.3
0.0
0.1
-0.8   *invalid data entry
2.6
2.0
1.2
0.0
0.9
0.0
-1.8  *invalid data entry
0.0
0.0
9999
NUMBER OF VALID RECORDED DAYS = 13.
NUMBER OF RAINY DAYS RECORDED = 7.
AVERAGE RAINFALL PER DAY OVER PERIOD = .67 INCHES
MAXIMUM RAINFALL IN ONE DAY = 2.6 INCHES
```

Figure 2.2 – Sample Input and Output for Rainfall Problem

For the above program four languages by four different groups of novice programmers were used to solve the rainfall problem and the following is the analysis of C programs according to the Plan Composition method.

Plan Difference: Spurious (35.0%), Missing (31.3%), Malformed (22.5%), and Misplaced (11.3%). In all languages of study, most errors occurred as a result of novice programmers adding extra unnecessary code to the program (Spurious Plan), not including part of the problem specification (Missing Plan), not writing the code

for the plan correctly (Malformed Plan), and putting the code of a plan in the wrong place (Misplaced Plan).

The following analyzes the errors based on plan components. For example, the Plan Component categories for the C group are as follows: Initialization (28.8%), Guard if (27.5%), Update (13.8%), Guard Loop (8.8%), Input (7.5%), Declaration (7.5%), and Output (6.3%).

To pinpoint the exact cause of error, a plan category with its components is used and is shown as follows: Missing Guard if (20.0%): Highest error percentage, missing if statement to check for division by zero. Novices do not check for special cases and write a program around given input. Also, texts do not emphasize error checking.

Spurious Initialization (20.0%): Due to lack of understanding when variables need to be initialized.

Malformed Loop (7.5%): Students did not realize that the sentinel value is the very last value in the data. Checking for an EOF is not needed; therefore it was considered a malformed error.

Misplaced Update (6.3%): Major cause of this error was updating within the loop when it should have been done outside of the loop, e.g. the computation of the average was inside the loop.

Malformed Initialization (5.0%): Most errors in the Malformed Initialization group appeared in the for loop. One possible reason is that the for loop in C is very different from the for loop in Pascal where the initialization, termination test, and update are done in the same line explicitly. This caused a great deal of logical confusion for students. They did not understand why the third variable (update) in the C for loop was necessary since it was not required in the Pascal for loop.

Missing Output (5.0%): Errors could be due to carelessness or students thought that as the appropriate information was calculated, it would automatically print the result. For example, in LISP, the function returns and displays a value without being told to write it to the screen.

Spurious Update (5.0%): The C programming language uses short intensional notations such as c += 1, ++c, or c++, for c = c + 1 and s += c for s = s + c. The novice programmer not familiar with this notation becomes confused. It was found that declarations of unnecessary variables also led to Spurious Update errors.

Missing Declaration and Malformed Declaration (3.8%): Some of these errors were due to the incorrect declaration of variables, e.g. declaring variable average as integer instead of float.

The study of errors by novice programmers for all languages concludes that “missing guard if” was the most prevalent among errors. This error was the result of

missing error checking or misunderstanding of a decision-making component of the program [Ebrahimi 89].

The following program was used as a solution for the rainfall problem in the analysis of plan errors. We have chosen the C version over other languages since most of today's conventional languages are C based and the programming code is familiar to most if not to all.

```
#include <stdio.h>
main( )
{
        int validday=0, rainyday=0;
        float avg, max, rainfall, raintotal=max=0.0;
        printf("RAINFALL\n");
        scanf("%f",&rainfall);
        while(rainfall != 9999)
        {
                if (rainfall >=0)
                {
                        ++validday;
                        if (rainfall)
                        {
                                ++rainyday;
                                raintotal += rainfall;
                                max = (rainfall > max ? rainfall :max);
                        }
                        printf("%5.2f\n", rainfall);
                }
        else
        printf("%f * INVALID DATA ENTRY\n", rainfall);
        scanf("%f", &rainfall);
}
if (validday > 0)
{
        avg = raintotal/validday;
        printf("\n\n\nNUMBER OF VALID RECORDED DAYS = %d\n\n", validday);
        printf("NUMBER OF RAINY DAYS RECORDED = %d\n\n", rainyday);
        printf("AVERAGE RAINFALL PER DAY OVER PERIOD = %5.2f
```

```
            INCHES.\n", avg);
        printf("MAXIMUM RAINFALL IN ONE DAY = %5.2f INCHES.\n", max);
}
else
        printf("NO DATA ENTERED");
}
```

Figure 2.3 – C Solution for Rainfall Problem

The following is a sample of a buggy program written by a student for the rainfall problem that is analyzed according to plan errors. The student unnecessarily initializes variables, the guard for the loop is done incorrectly, and a guard for divide by zero is missing; two update statements are misplaced as well.

```
#include <stdio.h>
main( )
{
        float rainfall, sum, average, max;
        int day, valid_days, rainy_days;
        valid_days = 0;
        sum = 0;
        average = 0;                                    /*Spurious Initialization*/
        max = 0;
        day = 0;
        rainy_days = 0;
        printf("day                rainfall\n\n");
        for(rainfall = 0; rainfall != 9999; ++rainfall)  /*Spurious Initialization*/
        {
                scanf("%f", &rainfall);
                if(rainfall == 9999)                            /*Malformed Guard*/
                        goto end;
                        ++day;                          /*Misplaced Update*/
                        printf("%d%f\n", day, rainfall);
                        if(rainfall >= 0)
                        {
                                ++valid_days;
                                if(rainfall > 0)
                                        ++rainy_days;
                                sum = sum + rainfall;
```

```
                                        average = sum / valid_days; /*Missing Guard If*/
                                                                    /*Misplaced Update*/
                                        if(rainfall > max)
                                                max = rainfall;
                                }
        }
end;
printf("number of valid recorded days = %d\n\n", valid_days);
printf("number of rainy days recorded = %d\n\n", rainy_days);
printf("average rainfall per day over period = %3.2f inches\n\n", average);
printf("maximum rainfall in one day = %2.1f inches\n", max);
}
```

Figure 2.4 – Sample Buggy Solution for Rainfall Problem

2.7 Correlation of Language Construct Errors and Plan Errors

Novice programmers are assessed based on their ability to correctly chose, implement, and integrate plans. Most novice programming errors arise from mismanagement of the plans - incorrect Plan Composition. Language constructs comprehension, plan composition, and their relationships to each other are found to be two major causes of errors. These errors have also shown to be highly correlated. The study investigates novice programmer errors with experiments on language constructs and plan composition. The relationship between language constructs and plan composition errors is analyzed. The study was conducted using four programming languages – Pascal, FORTRAN, C, and LISP. In addition the study concludes that languages tend to generate their own specific errors as a result of their peculiar design [Ebrahimi 94].

2.8 Environments for Learning Programming

Visual Plan Construct Language (VPCL)

Visual Plan Construct Language (VPCL) is a programming learning environment that emphasizes the importance of plans and their manipulation. The novice programmer can learn about plans, how to decompose problems into plans, and ultimately how to put plans together to create a working program. It allows new plans to be created or existing ones to be modified. VPCL helps novice programmers understand different language constructs and their alternatives by providing a language construct library that is not language-dependent and can thus be adjusted to use the current conventional languages. Initially, VPCL starts with a library of common programming problems and other utilities known as the plan library and this library can be expanded by the educators as well as learners. VPCL consists of three selectable phases. These phases are rehearsal, partial implementation and full implementation of plans; known as plan observation, plan integration and plan creation. Empirical study suggests the relative superiority of visual learning (VPCL) over traditional textual learning [Ebrahimi 92].

In the Plan Observation phase, the learner observes how a problem is solved step by step and can repeat the process until it is understood. The process illustrates the steps involved in a task, starting with the initial specifications of the problem to its final solution, from generalization to specification. A solution to a problem consists of several sub-plans that need to be integrated correctly.

The Plan Integration phase tests and reinforces learners' understanding of the relationship between plans such as how the plans are put together, spatial relationships, and order. There are four ways plans can be integrated: an appended plan (plans are one after the other sequentially e.g. steps); an interleaved plan (go back and forth between plans, e.g. relationship between day and night); a branched plan (a plan can be diverted to other plans based on some condition e.g. a plan will lead to one plan or another); or an embedded plan (a plan can be entirely embedded within another plan, e.g. Russian dolls).

Plan integration is a good measure of understanding the solution to a problem at an abstract level, rather than its detailed steps.

The following diagrams show the Plan Integration modes:

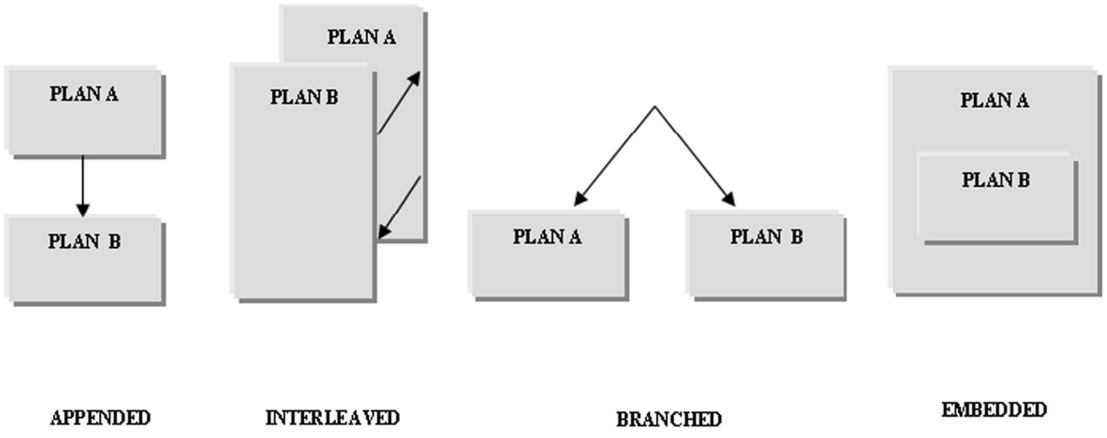

Figure 2.5 – Plan Integration Modes

Let's say we want to write a program to compute the average of a series of numbers. The find average plan could be decomposed into the following sub-plans: getnumbers, countnumbers, sumnumbers, exception, errormsg, computeavg, and

output. In this example, the sumnumbers, countnumbers, and getnumbers plans are implemented so that they are interleaved with each other. This means that when a number is input (getnumbers), the counter is incremented (countnumbers), and this number is added to the sum (sumnumbers); now the process returns to the getnumbers plan and starts again. When all numbers are entered, the process follows sequentially to the exception plan (an appended plan), which checks for an error, such as division by zero. The errormsg and computeavg plans are branched plans, meaning that the errormsg plan will be entered on the condition that an error exists, otherwise the computeavg plan will be entered. The output plan is appended to the computeavg plan and outputs the result.

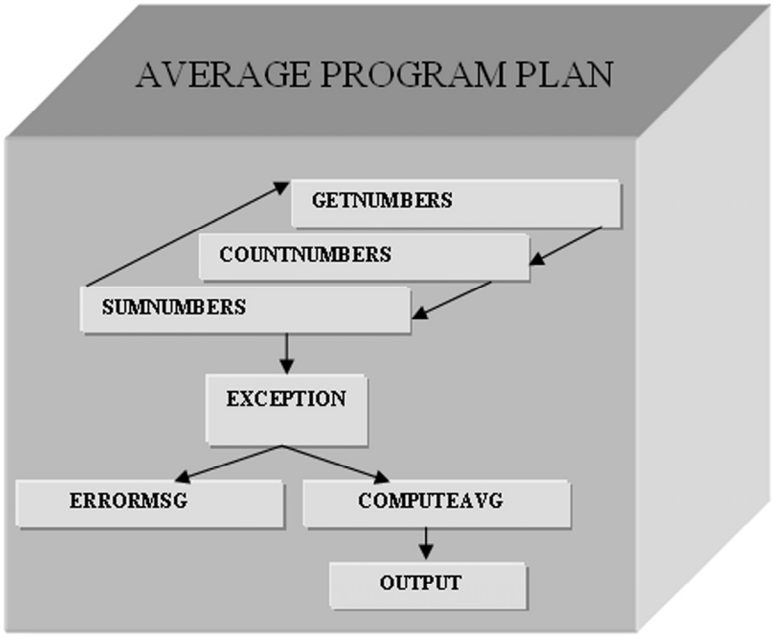

Figure 2.6 – Find Average Plan Example

In the Plan Creation phase of VPCL, the learner is responsible for creating a plan to solve a problem entirely, from problem specification and requirements to plan decomposition, plan integration, and finally testing the correctness of the plan and its explanation.

A Novice Programmer's Support Environment

Programming knowledge can be represented by five cognitive levels: Lexical, Syntactic, Semantic, Schematic, and Conceptual. [Liffick 96] A network of knowledge based on these levels can be used to annotate example programs for novices. The major problem for students appears to be making the leap from understanding individual program statements to the tasks accomplished by groups of statements. The Semantic level deals with the semantics of individual statements. The Schematic level refers to groups of statements which together form a programming plan (semantically meaningful knowledge unit). The relationships between these plans are represented in a hierarchical format. The Conceptual level deals with definable functions within the problem domain. ANNET is a hypertext environment which links each token and program statement in an example program to a detailed explanation of their lexical, syntactical, and semantic significance. [Liffick 96]

Alice: A 3D graphics programming environment

Alice is a 3D graphics programming environment which enables a broad audience of end-users to create 3D interactive content with no graphics or

programming background. [Conway *et al.* 00] Alice is a tool for describing the time-based and interactive behavior of 3D objects. Alice consists of two phases: creating an opening scene and scripting. Users add objects to the scene and then modify the object's properties and methods. The authors made several observations of novices who used Alice. For example, surface characteristics of programming languages matter to novices, especially case sensitivity and careful name choices. Users often assumed the environment would correspond to the real world and expected automatic collision detection and gravity. Interesting interactive 3D graphics authoring still involves some logic specification/programming for representing conditional behavior, which requires an "if-then" construct. Currently, Alice focuses on scripting; however the author mentions that future systems would need a combination of techniques such as programming-by-demonstration and visual programming. [Conway *et al*. 00] Alice may also demonstrate some pedagogical pitfalls: the object model used may lead to misconceptions and the lack of focus on syntax is detrimental to students' transitioning to C++ or Java. [Powers *et al.* 07]

2.9 Learning to Program via Web

Web technology enables students to enhance their learning experience by allowing exploration of programming concepts, environments, and tools. Incorporating web technology into programming balances between two traditional ways of teaching CS1 as either by breadth, by emphasizing an overview of computer science, or depth, through intensive programming, students are able to maintain

strong programming skills with web technology [Reed 01]. The following are examples of web based programming environments.

WebToTeach

WebToTeach has been developed for courses at Brooklyn College and offers an interactive programming exercise system online [Arnow *et al*. 99]. It is based on an automatic program checking software and enables instructors to specify programming problems and acceptable solutions. WebToTeach enables students to write code fragments, single source programs, as well as several source files and receive immediate feedback.

Environment for Learning to Program (ELP)

Environment for Learning to Program (ELP) is an interactive web-based system that helps teaching programming to the novice by not having the need for a Program Development Environment and installation of a programming language. [Truong 05] Using ELP, students use template exercises through the web to build their programming and problem solving skills. ELP also contains a program analysis framework that analyses students' programs and provides feedback on the quality and correctness of their solutions. The static analysis focuses on the structure and quality of code as well as comparison to correct solutions. The following functions and checks are provided: program statistics, number of logical decisions in a gap, unused parameters, redundant logical expressions, unused variables, hard coded

numbers or strings in solutions, correctness of access modifiers, and a check for default case statements in switch statements. The dynamic analysis includes black box and white box tests to verify the correctness of students' solutions and detect at which gap errors occur so feed back can be provided to the student. [Truong 05]

2.10 Summary

Based on preliminary research, it may be concluded that the cause of novice programmer errors is not the misunderstanding of language constructs or knowing how to solve a problem; the errors arise at the stage of plan creation and integration. Novices also have difficulty with object oriented concepts and correct utilization of objects in their programs. Design and incorporation of objects is a major obstacle for novices and the extra time spent on learning objects has taken away focus on problem solving and algorithms

Use of web, visualization, and plans can be used to create an effective learning programming environment, especially for object oriented concepts. A web-based environment that extends plan use to incorporate objects, reinforces plan understanding, and provides assessment would further enhance the learning experience for a novice programmer.

3 Novice Object Design: Challenges and Solutions

3.1 Novice Difficulties with OOP Concept

With the introduction of objects into programming courses, novices are lacking programming problem solving skills due to spending more time understanding objects and their usage. As a result, novice programmers have less focus on problem solving and analysis of algorithms at an early stage of programming. In addition, the standard programming teaching sequence cannot be followed when objects are taught first; for example, to implement an object, students need to know variables, functions, and pass of parameters beforehand. In order to understand what objects are and what objects are made of, it is necessary to have a foundation of basic programming fundamentals. By introducing objects as the first topic, many details that are vital to understanding objects are overlooked.

Often novice programmers attempting to write a program to solve a problem using OOP lose sight of the goal and misuse objects. The concept of objects (OOP) can be ambiguous and overwhelming for novice programmers.

The “Hello World” programs used in many programming textbooks present objects in a way that is harmful to novices. For example, classes including only methods and no encapsulated data mislead novices about the concept of objects and provide no real advantage to problem solving. Learning multiple paradigms including procedural programming gives students the opportunity to see how objects oriented programming solves problems in terms of simulating real-world objects. [Hu, 2005]

In the March 2008 issue of *Communications of the ACM*, Chenglie Hu discusses key timely issues about novice programming education that are closely tied to this work. Hu points out that it is important for novices to have a consistent understanding of data types. It is essential for novices to get a sense of what a class is, what it can do, and have a proper understanding of a user-defined type. “Objects-first” is a popular approach in programming courses and textbooks, despite the existing controversies. Programming software, such as Alice, use visual objects such as animals, robots, and games, but these do not clearly translate to developing programming skills. Textbooks supporting the “objects-first” approach also have some inherit problems and lack problem solving. Many textbooks tend to use, what Hu describes as, “toy objects” that cannot be used in a real world application. Novices need to learn what data and methods should be included in an object and would benefit from more practical examples. Many computer science educators have even advocated a “back to basics” approach. Hu states, “Learning computer programming essentially involves two things: computer algorithms and how data

types are defined, designed, and used to solve problems." At this point in time, a new approach, other than "objects-first" or "back to basics" is needed. Some issues to be addressed in a new approach are: the role of data types in problem solving through programming, more problem solving with user-defined types, inheritance and its role in problem solving, and problem solving case studies. [Hu 2008]

Novices often fail to properly represent objects in their programs. Common errors are related to determining the necessary attributes and functions for an object. Empirical studies of novices in traditional and online courses reveal errors related to plan integration and incorporation of OOP concepts into problem solving. [Ebrahimi and Schweikert 06]

3.2 Plan-Object Paradigm

The Plan-Object Paradigm (Plan Oriented Objects) is a learning paradigm that applies concepts of plan representation and integration to objects. This paradigm represents the conceptualization and design stage of objects. Plans are used to define objects, their components, relationships, and integration into other plans. The goal is to increase the novice's ability to design and implement objects, as well as incorporate objects into their program to form a complete solution to a problem. The plan representation, along with plan properties, is then transitioned into code (class implementation).

By setting a relationship between plans and objects, the Plan-Object Paradigm presents a method for early assimilation of OOP. Objects are explicitly defined and given context within the plan framework. Plans are used to guide object creation and aid novices to properly create and utilize objects in their programs. A plan represents an object by setting its properties and sub-plans. Inheritance relationships and the proper attributes and functions necessary to solve a problem are also represented and defined. A plan determines how an object will be constructed and how it will be used. As new plans are incorporated, more information and/or functionality will be added to the object(s) as needed.

An Object Plan consists of Data Member, Member Function, and Object Utilities sub-plans (class components). Depending on the application, appropriate Variable(s) and/or Function(s) will be created and integrated. An Object's Utilities include Set Plan, Get Plan, Constructor, and Destructor sub-plans.

For example, a Payroll Plan will necessitate an Employee Object. According to the tasks of the Payroll Plan, the proper data members and member functions will be created and integrated into the Employee Object. In this case, a Payroll Plan would require hourlyRate and hoursWorked data members as well as a computeGrosspay member function in the Employee Object.

4 Design of WPOL (Web Plan Object Language)

4.1 Overall Description

Web Plan Object Language is an online environment for learning to program in the object oriented paradigm; WPOL incorporates concepts of plan programming and the Plan-Object learning paradigm. WPOL is plan oriented and addresses challenges novices face with plan integration and object design.

A plan representation of programming knowledge facilitates novices in developing their skill in applying a particular programming concept or task. Programming concepts are represented visually: for example, by plan representation or abstract icons or images. Visualization helps novices retain a mental representation of an object, concept, or algorithm, compared to a textual representation.

WPOL focuses on plan integration and introduces a Plan-Object learning paradigm that sets a relationship between plans and objects. The plan framework provides a context and meaning for objects. A plan will direct object creation by determining the proper data members and functions, as well as the object's relationships and incorporation into the problem solution.

4.2 Phases of Learning

WPOL consists of three phases of learning: Plan Observation, Integration, and Creation. Each phase is desribed below.

4.2.1 Plan Observation

The goal of the Observation Phase is to demonstrate, step by step, the solution to a problem in terms of Plans and Objects. Plan design and integration are visualized, with Plan Properties explicitly defined. WPOL consists of sample “Master Plans” (e.g. Student, Employee, Inventory, Accounts) that the student can observe. The Integration View shows how Plans are integrated and their Integration Mode: Appended, Embedded, Interleaved, or Branched. In Code View, Plans are implemented according to their Plan Properties.

The Observation Phase of the Student Average Plan is illustrated in Figures 4.1-4.15 and is described below.

The Observation Phase begins with the problem description and identification of the major components. The Student Average Plan, which is a Level 1 (Beginner) Master Plan, will be used as an introductory example.

Student Average Plan Description:

Compute the average of a student's assignment, midterm, and final scores. A student's id, assignment, midterm, and final scores will be input. The student's id, along with the computed average, will be displayed.

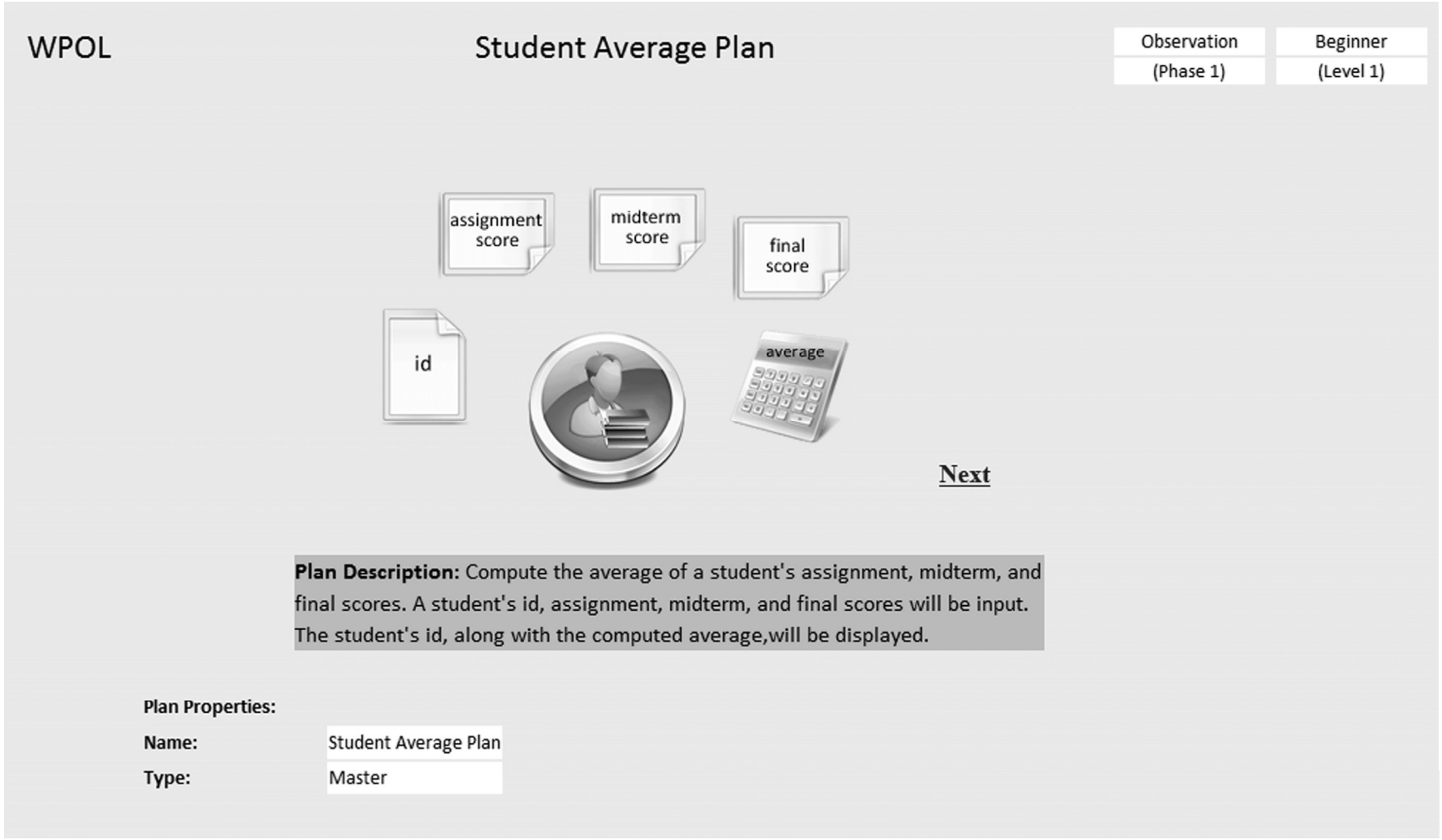

Figure 4.1 – Student Average Plan Description

The observation then identifies the objects that are needed in the program. For each object, an Object Plan is defined. An Object Plan consists of embedded Data Member and Member Function Plans, as well as Object Utilities. The Object Plan is then customized for the particular object at hand.

The Data Member Plan is of type “Class Component” and consists of Variable sub-plans. The Observation Phase visualizes the process of identifying data members and integrating the variable plans (Integration Mode: Embedded) into the Data Member Plan. Variable Plan Properties consist of Name, Data Type, and Accessibility. This example includes a Student Object Plan, whose Data Member Plan consists of embedded plans id (Type: Variable, Data Type: Integer, Accessibility: private) and assignmentScore, midtermScore, finalScore (Type: Variable, Data Type: Double, Accessibility: private).

The Member Function Plan, also of type “Class Component,” has Member Functions as embedded sub-plans. Member Function Properties are Name, Return Type, Parameter(s), and Accessibility. In the Student Average example, the Member Function Plan consists of embedded plan computeAverage (Type: Function, Return Type: double, Parameter(s): none, Accessibility: public).

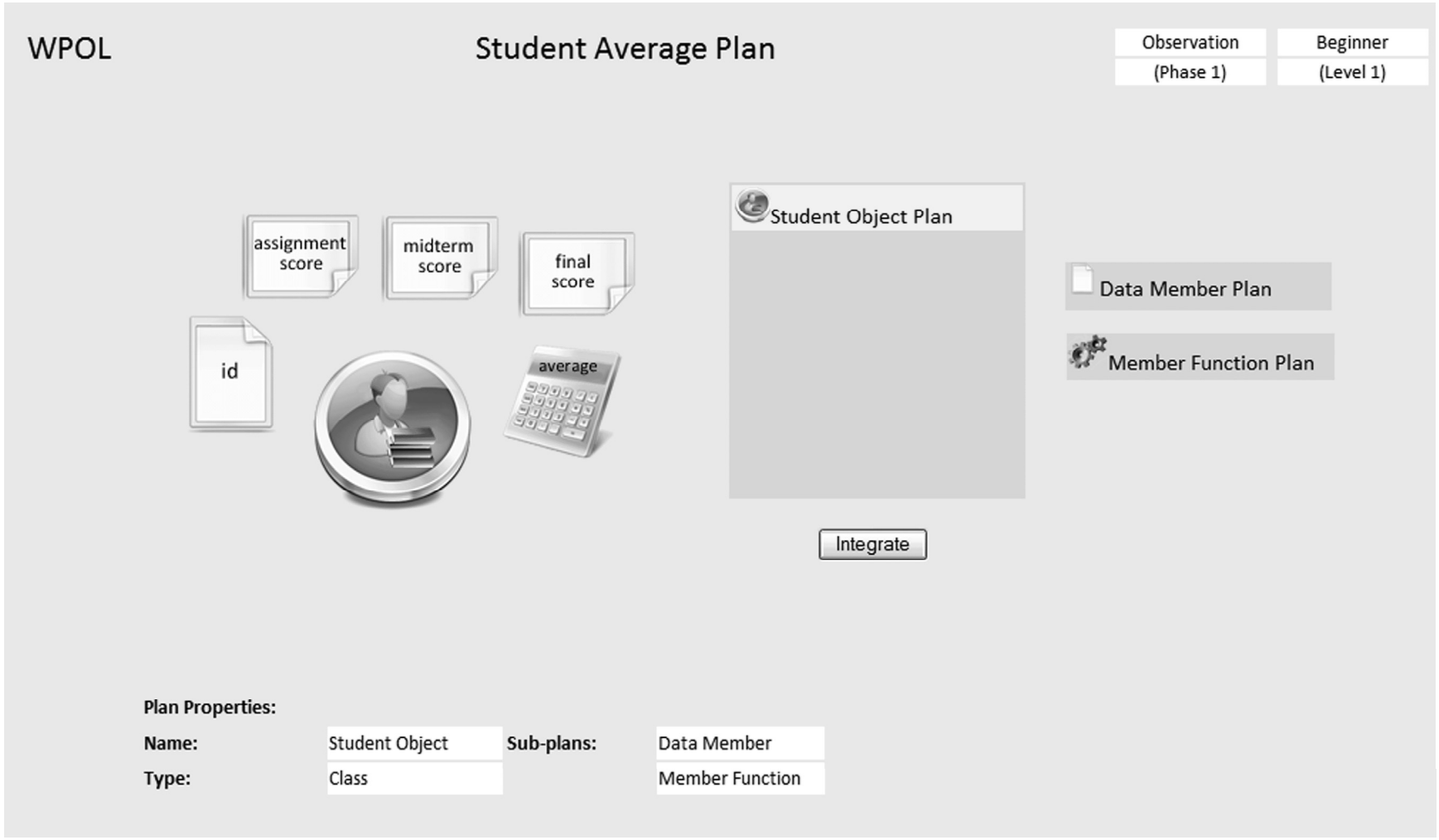

Figure 4.2 – Integration View for Student Object Plan

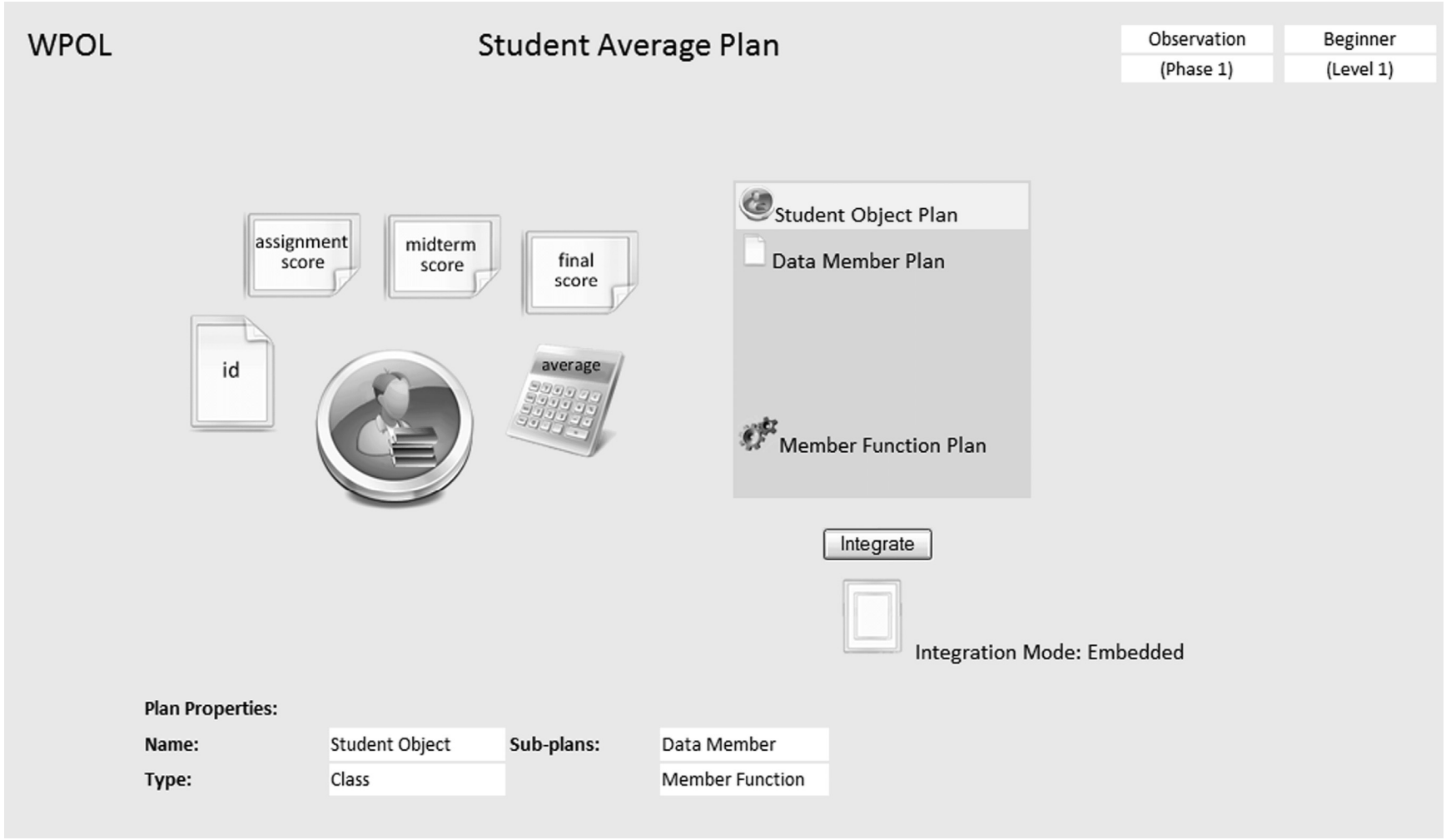

Figure 4.3 – Integration of Data Member and Member Function Plans (Embedded)

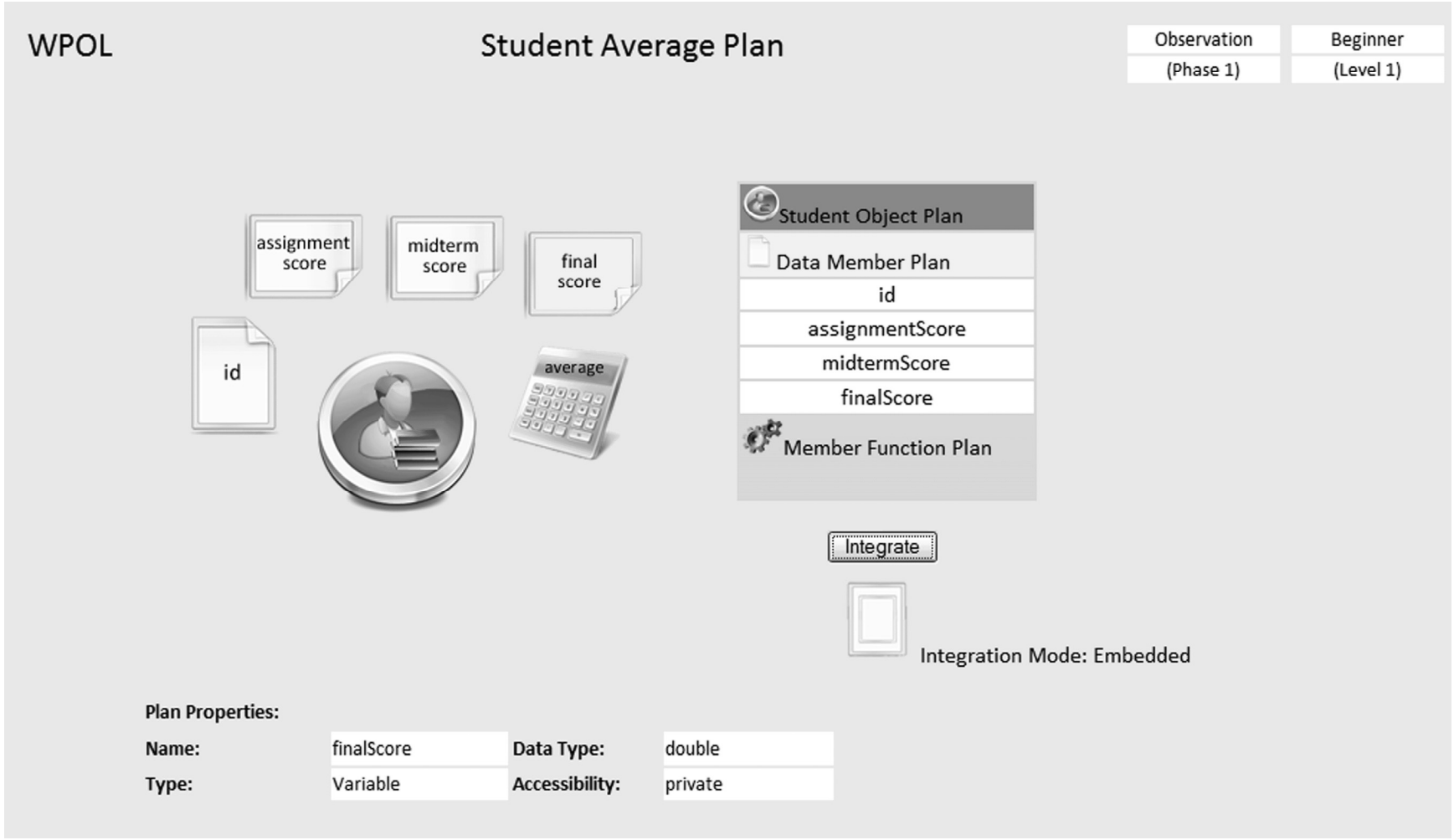

Figure 4.4 – Integration of Variable Sub-plans into Data Member Plan (Embedded)

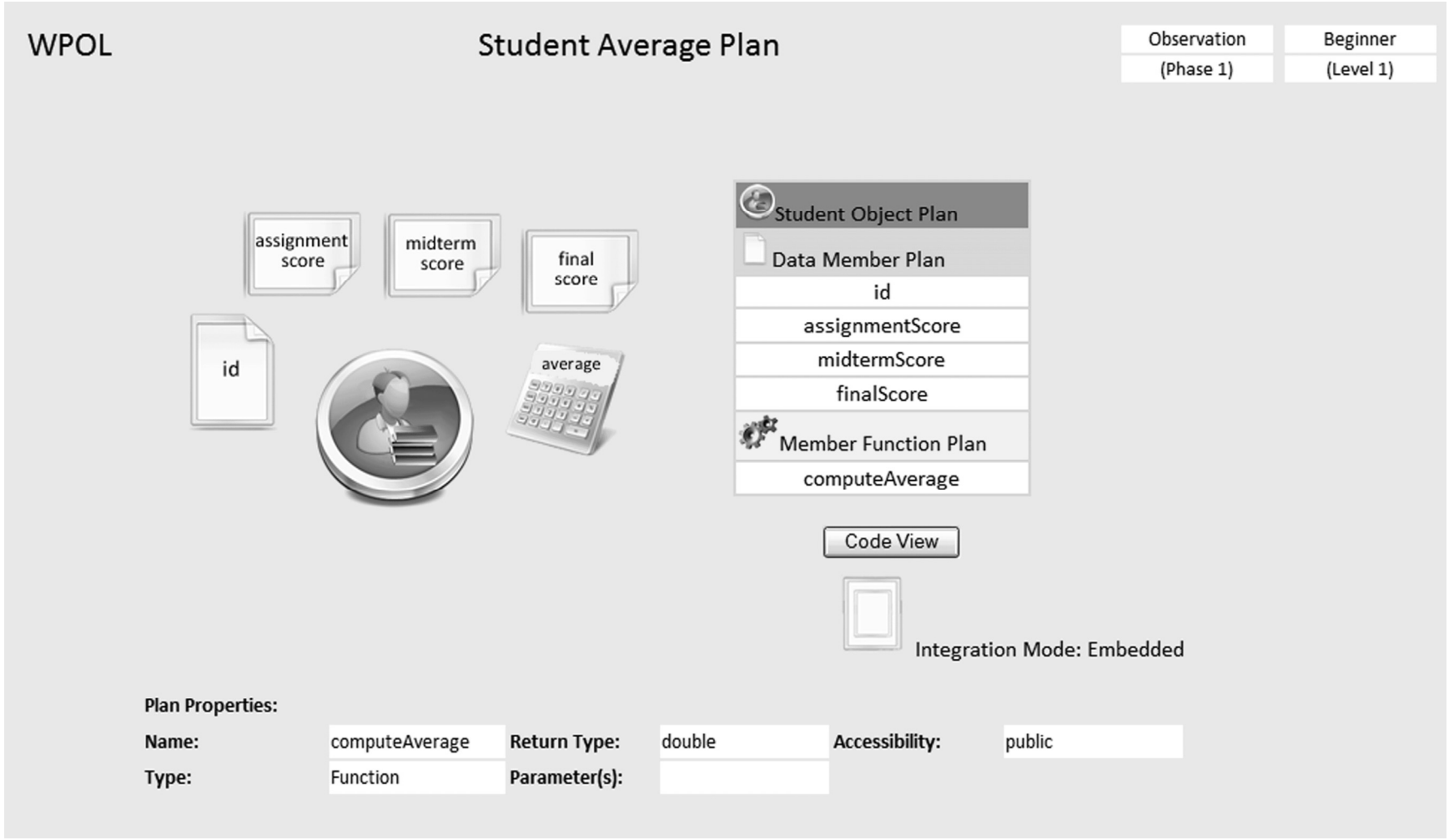

Figure 4.5 – Integration of computeAverage into Member Function Plan (Embedded)

When design of the Object Plan is complete, its implementation is demonstrated. Each sub-plan will be implemented one at a time, while its properties are displayed. This enables the observer to clearly see how each sub-plan and each plan property is translated into code. Plan Properties identify and define the necessary information needed to implement the plans. The association between each property and the code needed to implement it makes the translation from plan to code thoroughly understandable. The translation of the Student Object Plan into its class implementation is shown in Fig 4.6.

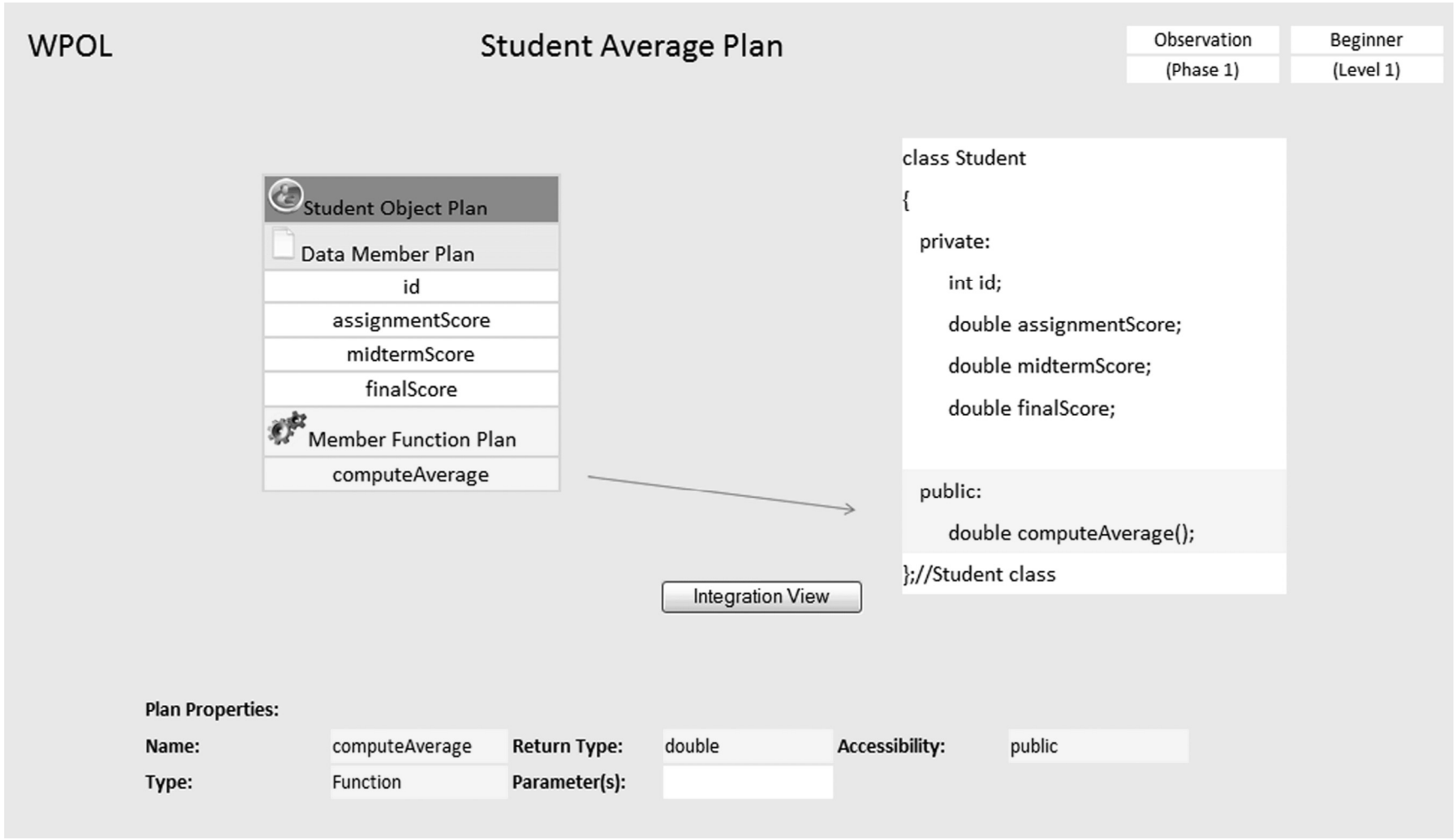

Figure 4.6 – Code View for Student Object Plan

The Functions of an Object's Member Function Plan are individually examined as to their properties and necessary sub-plans. To complete the implementation of the Student Object, the observation zooms into computeAverage. For computeAverage, Compute Average from the Plan Library provides a template for a simple average plan's sub-plans and relationships. computeAverage has its own sub-plans: Sum Scores and Count Scores. The Sum Scores Plan is integrated into computeAverage, Integration Mode: Embedded. The Count Scores Plan is appended to Sum Scores with the divide operation. computeAverage is then implemented as C++ code. The Properties of computeAverage provide information as to the type of plan (Function), scope of the function (member of Student Object), Return Type: double, and Accessibility: public. Compute Average contributes the declaration of the average variable, and average computation. The Sum Scores Plan includes declaration of sum variable, initialization to zero, and computation of sum. In this case, the Sum Scores Plan will compute the sum of scores contained in the Student Object. The Count Scores Plan (referring to Count [Simple]) is appended to the Sum Scores Plan with the division operator and will contain the number of scores. The generation of the code to implement the computeAverage Plan is shown in Figure 4.10.

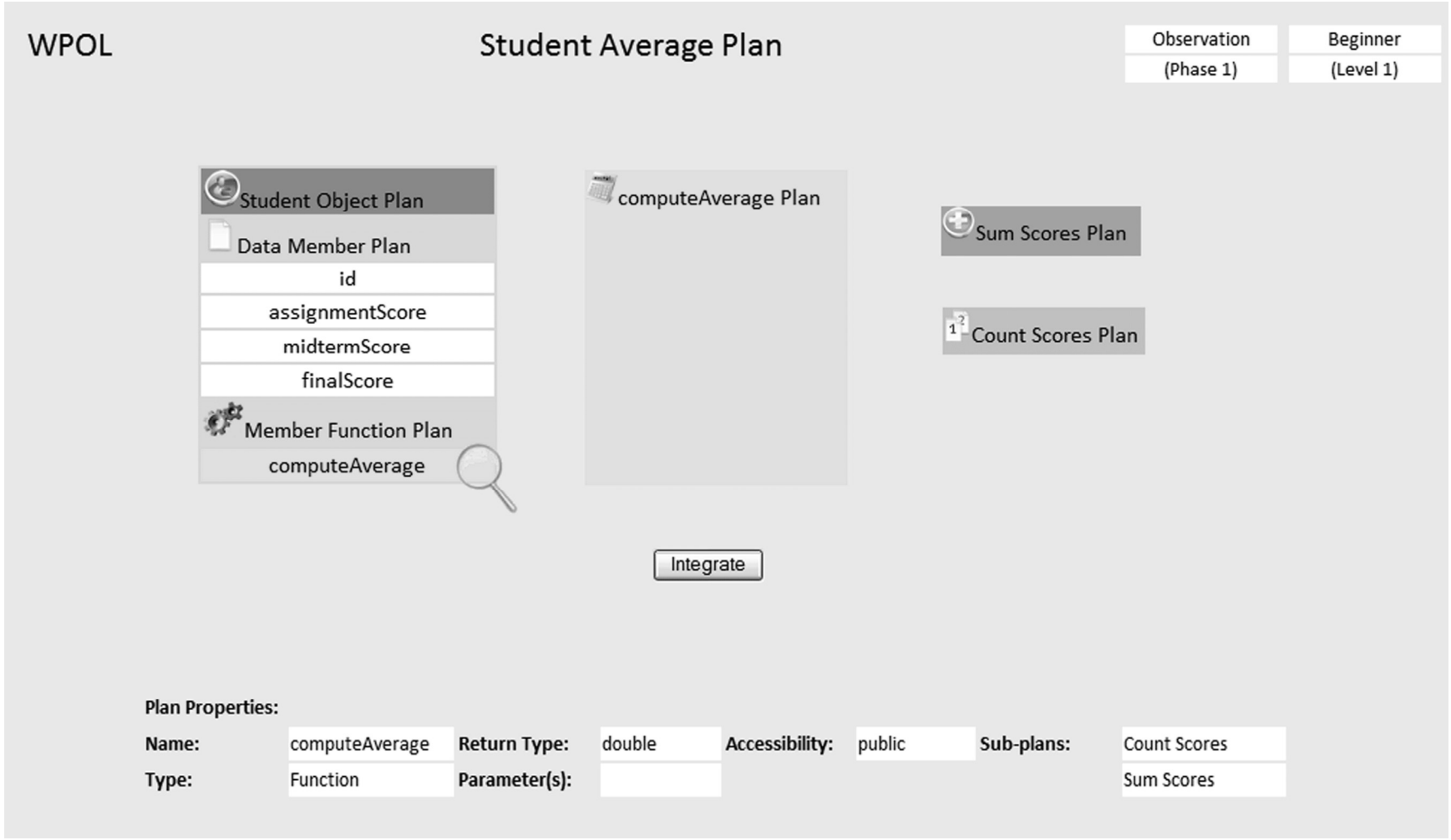

Figure 4.7 – Integration View for computeAverage Plan

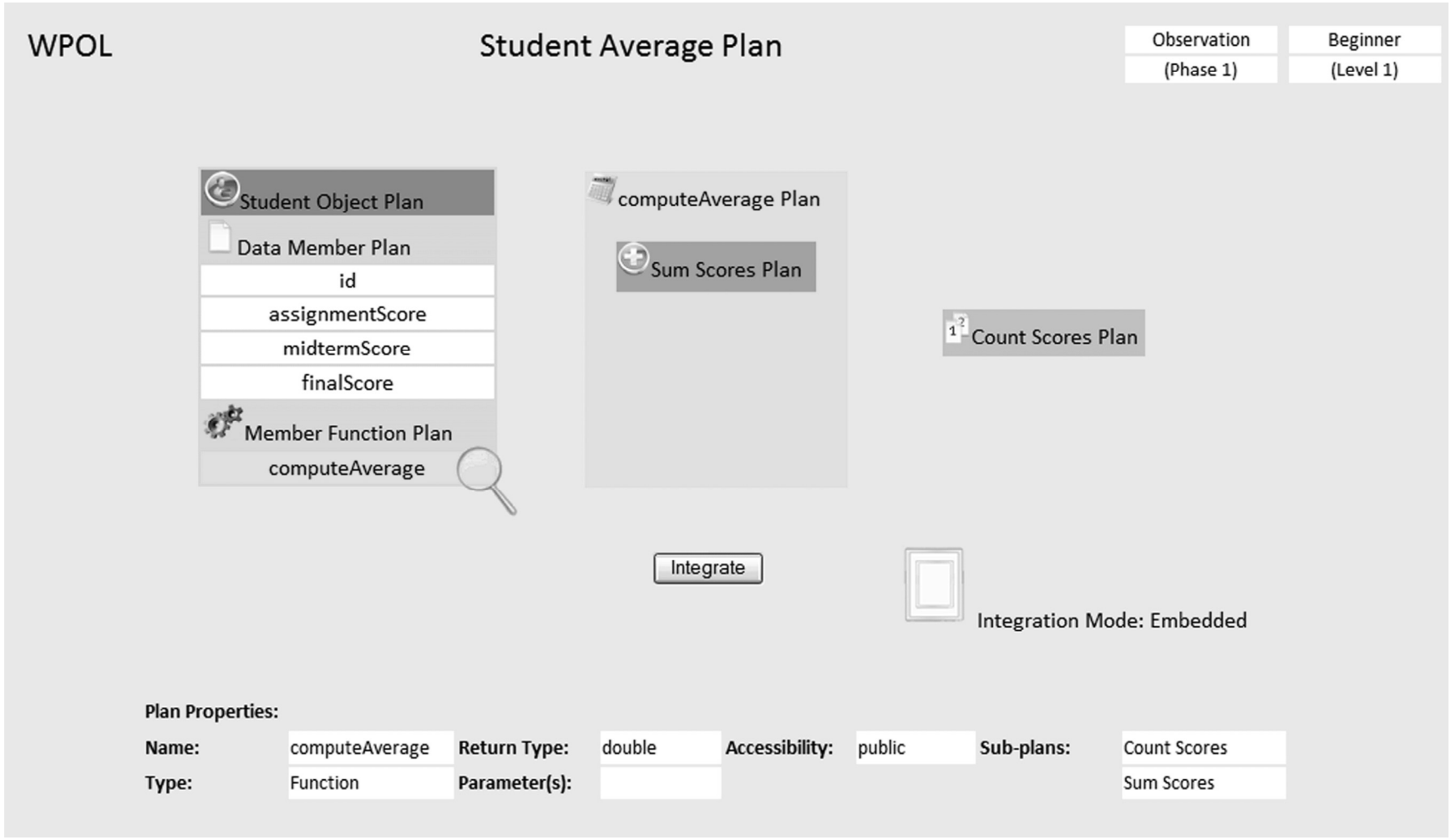

Figure 4.8 – Integration of Sum Scores Plan (Embedded)

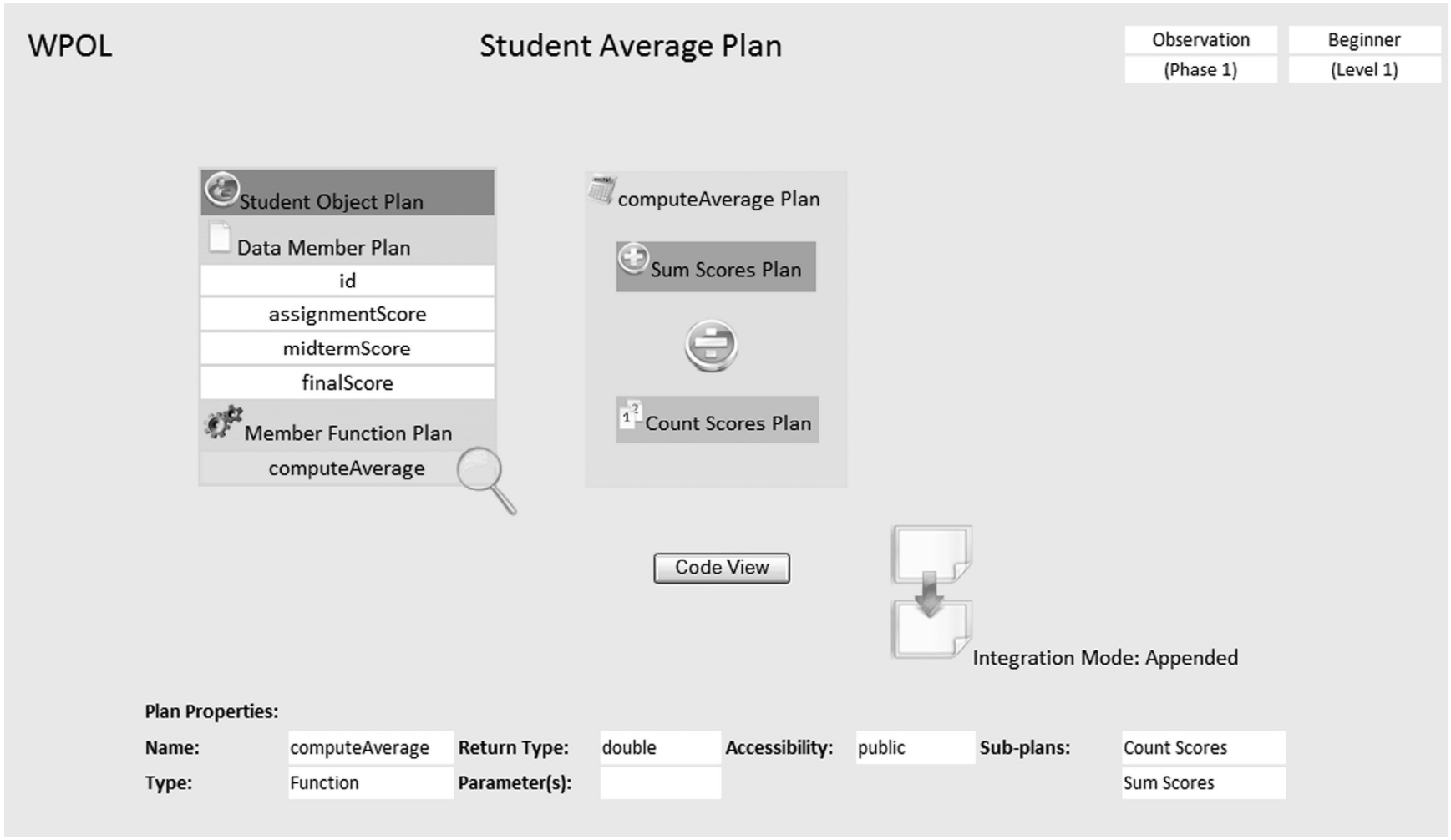

Figure 4.9 – Integration of Count Scores Plan (Appended)

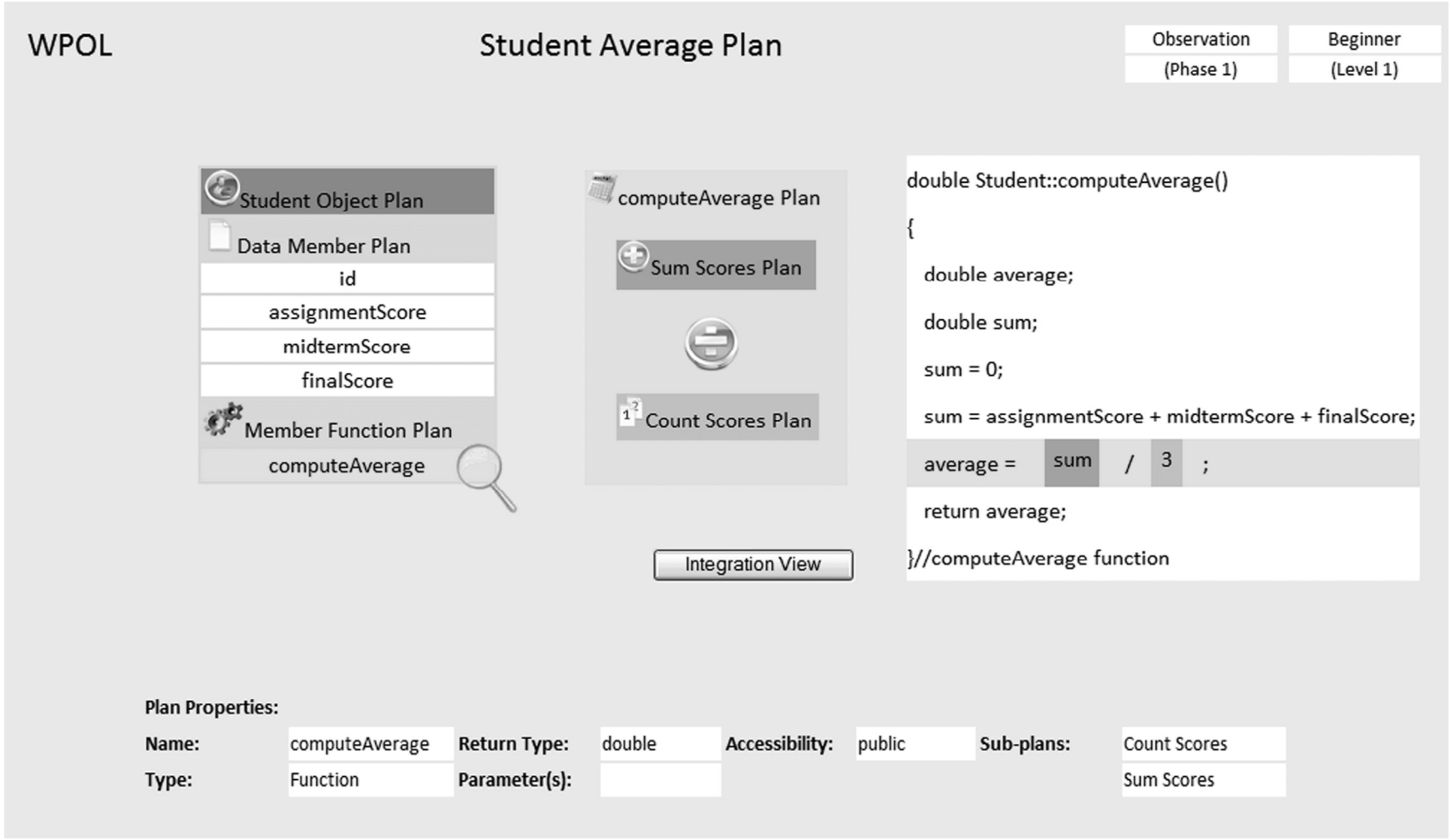

Figure 4.10 – Code View for computeAverage Plan

Fig 4.11 and 4.12 demonstrate the Integration View of the Student Average Main Plan. The Student Object Plan is considered Interleaved with the Student Average Main Plan since Student Average Main will be utilizing Student at different points within main. Creating an instance of the Student Object within Student Average Main connects these two plans. The Input Student Plan is embedded into Student Average Main. Input Student Plan [Mode: Console] gets required input from the user and utilizes the Set Plan from the Object Utilities to initialize private Data Members: assignmentScore, midtermScore, and finalScore. The Display Student Plan [Mode: Console] is appended to Input Student and utilizes the Get Plan to access the private Data Member: id, as well as computeAverage to compute display the student's average. Fig 4.13 illustrates the implementation of the Student Average Main Plan.

The "Run Plan" portion of the observation enables the student to run the program as it would run when executed. As the program is running, the plans that are being utilized at a specific time are highlighted and data members are initialized. (Fig 4.14)

The final step in the Observation Phase is to view the complete Student Average Plan program code. (Fig 4.15)

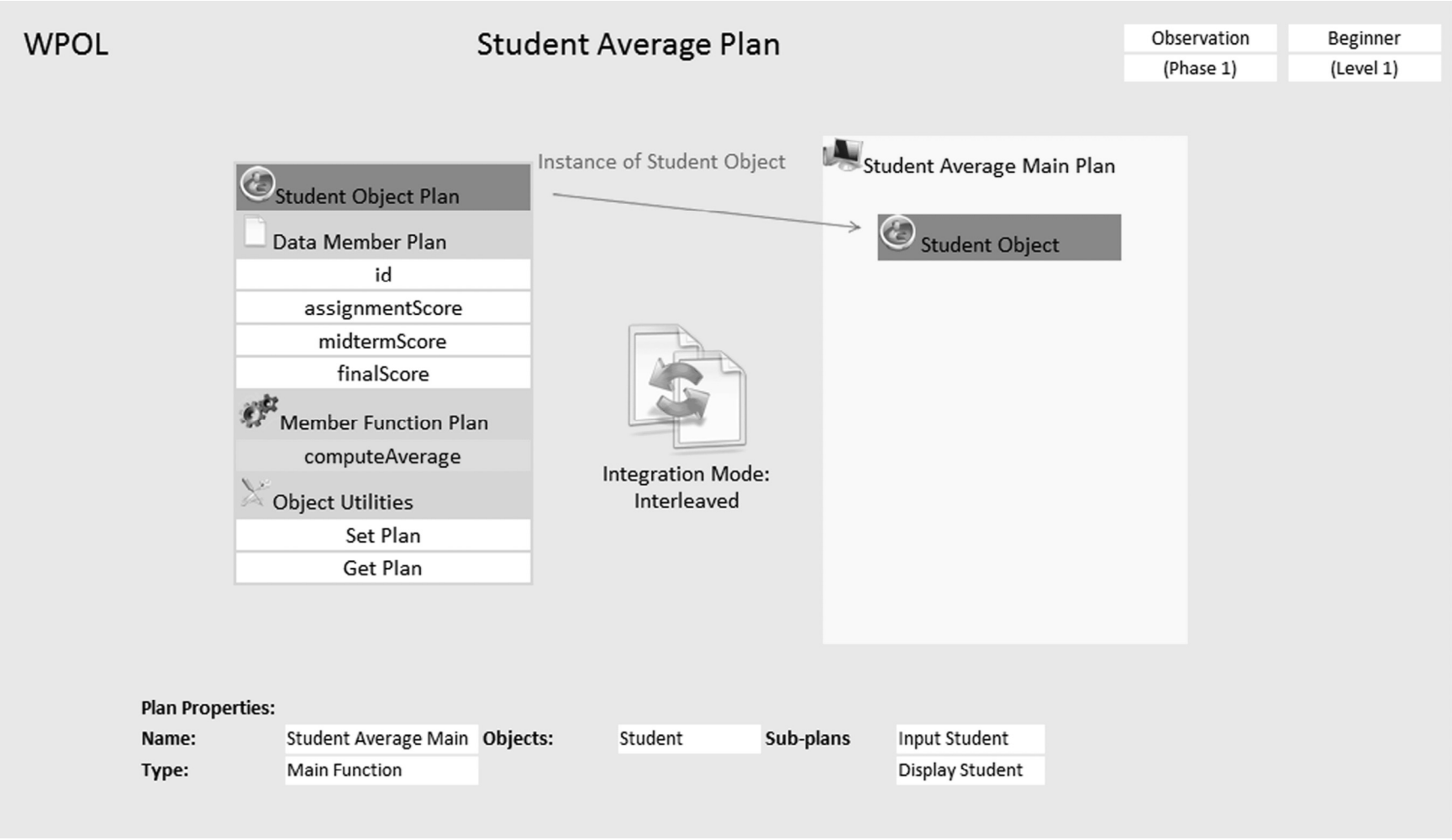

Figure 4.11 – Instance of Student Object in Student Average Main Plan

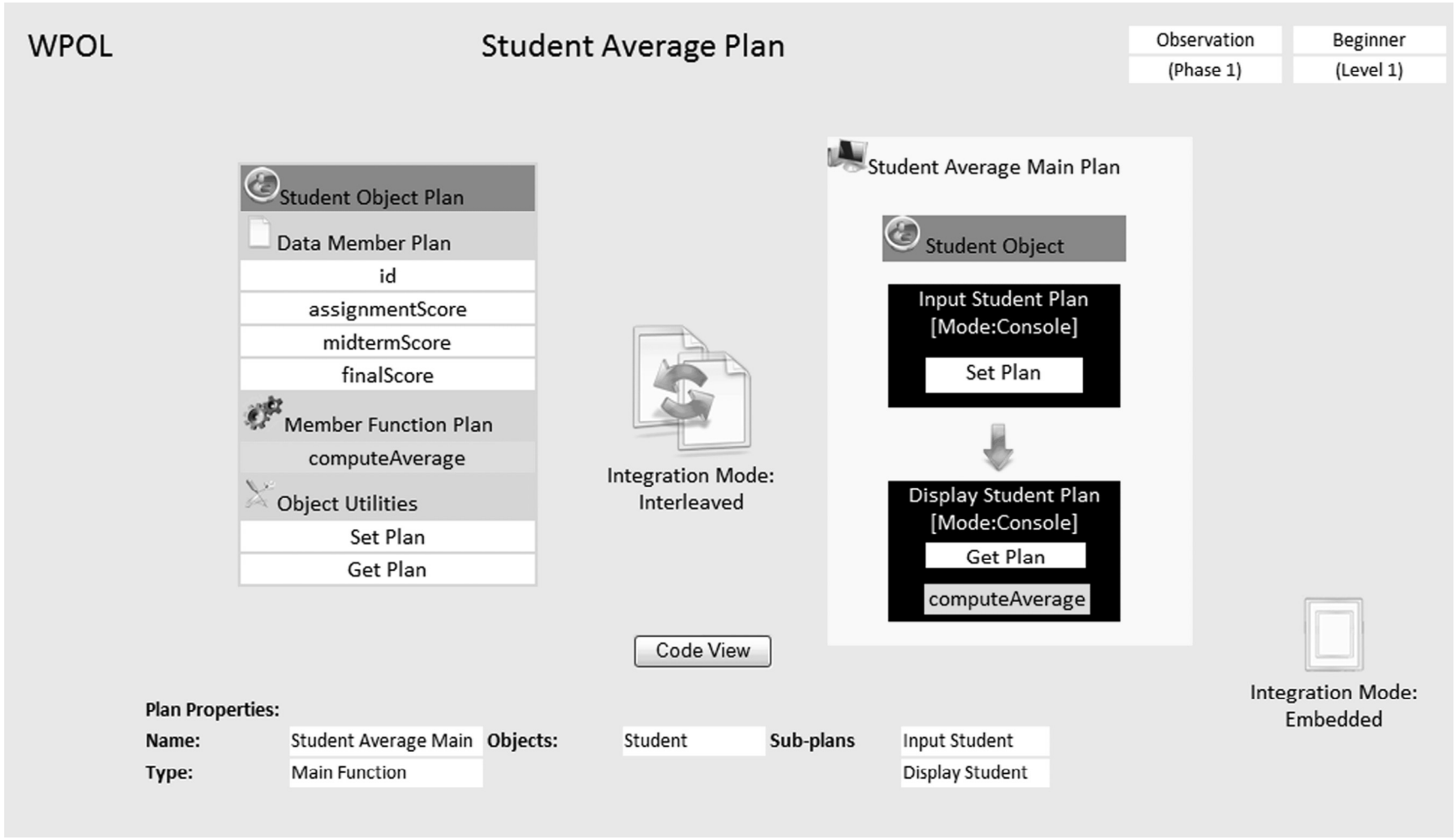

Figure 4.12 – Integration View for Student Average Main Plan

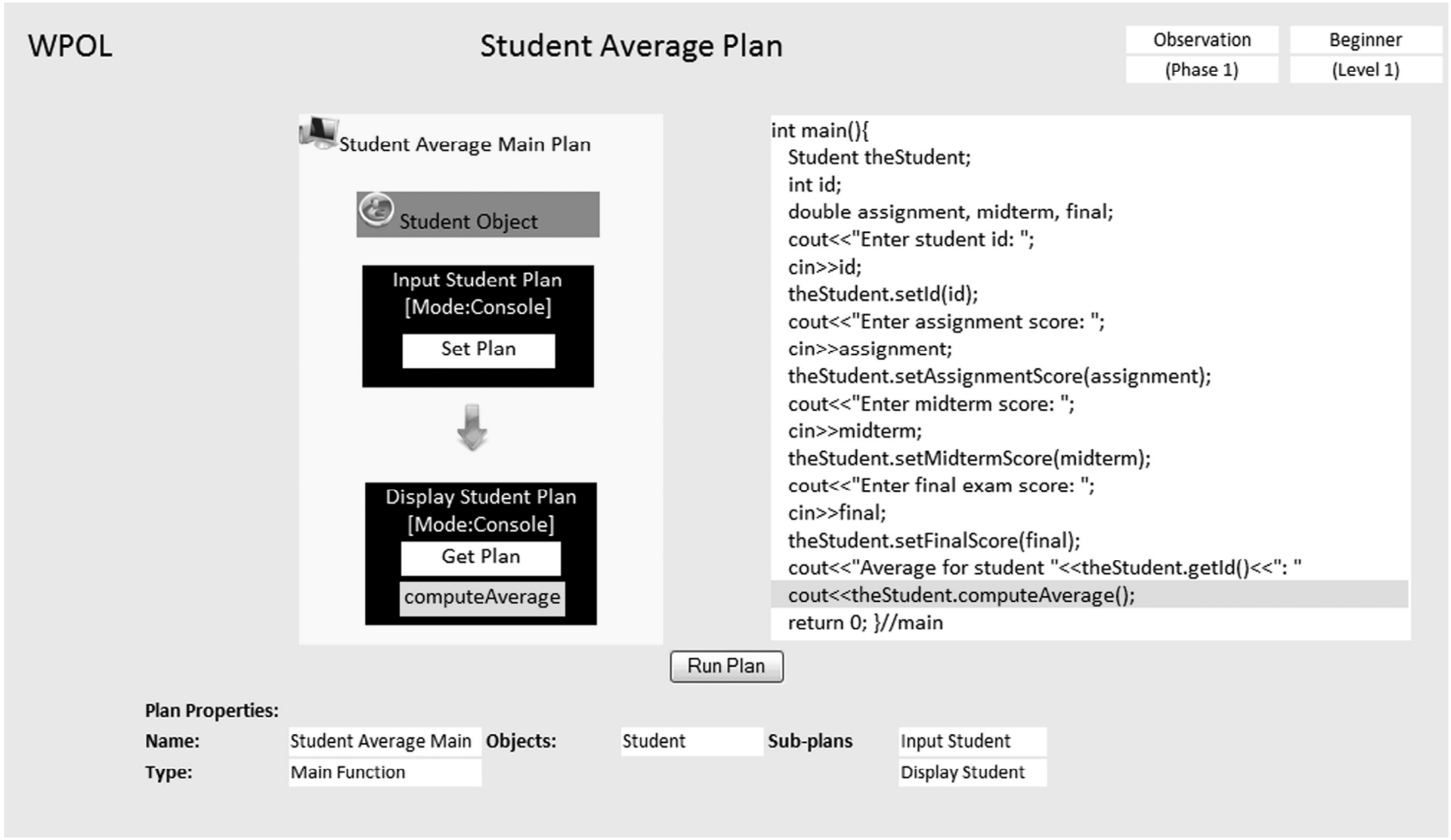

Figure 4.13 – Code View for Student Average Main Plan

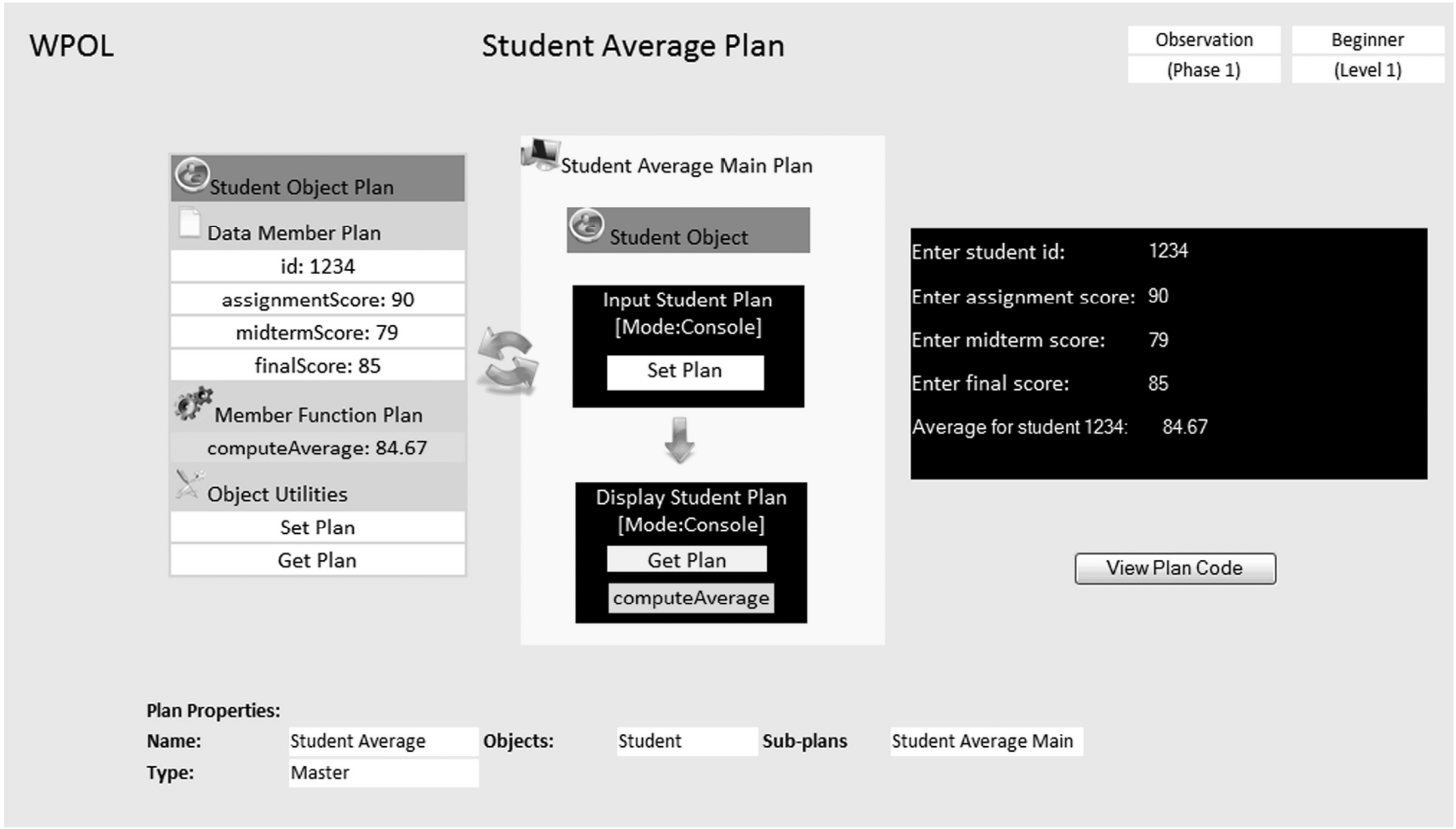

Figure 4.14 – Run View of Student Average Plan

WPOL

Student Average Plan

Observation	Beginner
(Phase 1)	(Level 1)

```
#include<iostream>
using namespace std;

class Student
{
 private:
         int id;
         double assignmentScore;
         double midtermScore;
         double finalScore;

 public:
        Student();
        Student(int i, double a, double m, double f);
        ~Student();
        void setId(int i);
        void setAssignmentScore(double a);
        void setMidtermScore(double m);
        void setFinalScore(double f);
        double computeAverage();
        int getId();
};//Student class

Student::Student()
{
    id=0;
    assignmentScore=0;
    midtermScore=0;
    finalScore=0;
}//Student class default constructor

Student::Student(int i, double a, double m, double f)
{
     id = i;
     assignmentScore = a;
     midtermScore = m;
     finalScore = f;
}//Student class constructor
```

```
Student::~Student(){}

void Student::setId(int i)
{
     id = i;
}//setId function

void Student::setAssignmentScore(double a)
{
     assignmentScore = a;
}//setAssignmentScore function

void Student::setMidtermScore(double m)

{
     midtermScore = m;
}//setMidtermScore funtion

void Student::setFinalScore(double f)
{
     finalScore = f;
}//setFinalScore function

double Student::computeAverage()
{
       double average;
       double sum;
       sum = 0;
       sum = assignmentScore + midtermScore + finalScore;
       average = sum / 3;
       return average;
}//computeAverage function

int Student::getId()
{
    return id;
}//getId function
```

```
int main()
{
    Student theStudent;
    int id;
    double assignment;
    double midterm;
    double final;

    cout<<"Enter student id: ";
    cin>>id;
    theStudent.setId(id);

    cout<<"Enter assignment score: ";
    cin>>assignment;
    theStudent.setAssignmentScore(assignment);

    cout<<"Enter midterm score: ";
    cin>>midterm;
    theStudent.setMidtermScore(midterm);

    cout<<"Enter final exam score: ";
    cin>>final;
    theStudent.setFinalScore(final);

    cout<<"Average for student "<<theStudent.getId()<<": ";
    cout<<theStudent.computeAverage();

    return 0;
}//main
```

Figure 4.15 – C++ Implementation of Student Average Plan

Sort Students Plan Observation Example

Sort Students Plan is a Level 2 (Intermediate) Master Plan. This example builds on the previous Student Average Plan and extends it by incorporating concepts of inheritance, array, loop, decision-making, and sorting. It also introduces the Branched type of plan integration. Inheritance is used to create new classes by extending existing classes. These new classes (derived classes) inherit data members and member functions from an existing class (base class). The concept of inheritance is essential when designing a software and programming in the object oriented paradigm and promotes software extendibility and reuse. The observation begins with the plan description (Fig 4.16) and depicts the necessary objects with their attributes and functions.

Sort Students Plan Description:

Build on the Student Average Plan to create a program that will input five students. In addition to the id, assignmentScore, midtermScore, and finalScore, the program will also include personal information about the student, such as their first name, last name, and email. The program will then display all students' information, sorted according to their average (ascending).

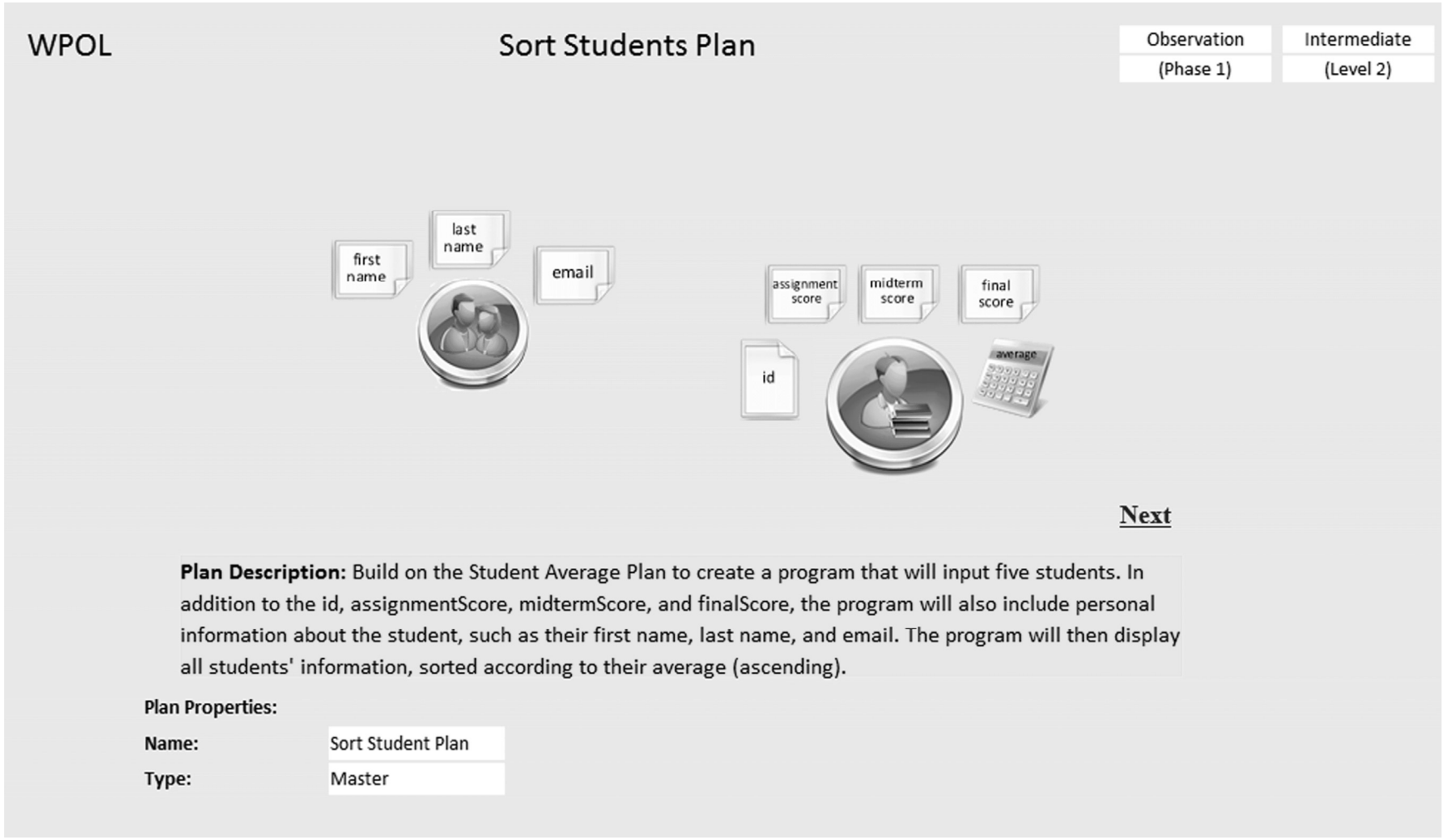

Figure 4.16 - Sort Students Plan Description

Since the Sort Students Plan builds on the Student Average Plan, Plans already created, such as the Student Object Plan, can be used. The Sort Students Plan Description asks to include personal information about the student, such as their first name, last name, and email. To do this, a Person Object will be designed with these attributes that the Student Object Plan will inherit.

The next step is to define the Person Object Plan. The Person Object Plan consists of Data Member and Object Utilities as embedded sub-plans. Data Member is of type “Class Component” and consists of Variable sub-plans. In the case of the Person Object Plan, Data Member contains the following as embedded sub-plans: firstName (Type: Variable, Data Type: string, Accessibility: private), lastName (Type: Variable, Data Type: string, Accessibility: private), and email (Type: Variable, Data Type: string, Accessibility: private). (Figures 4.17-4.20)

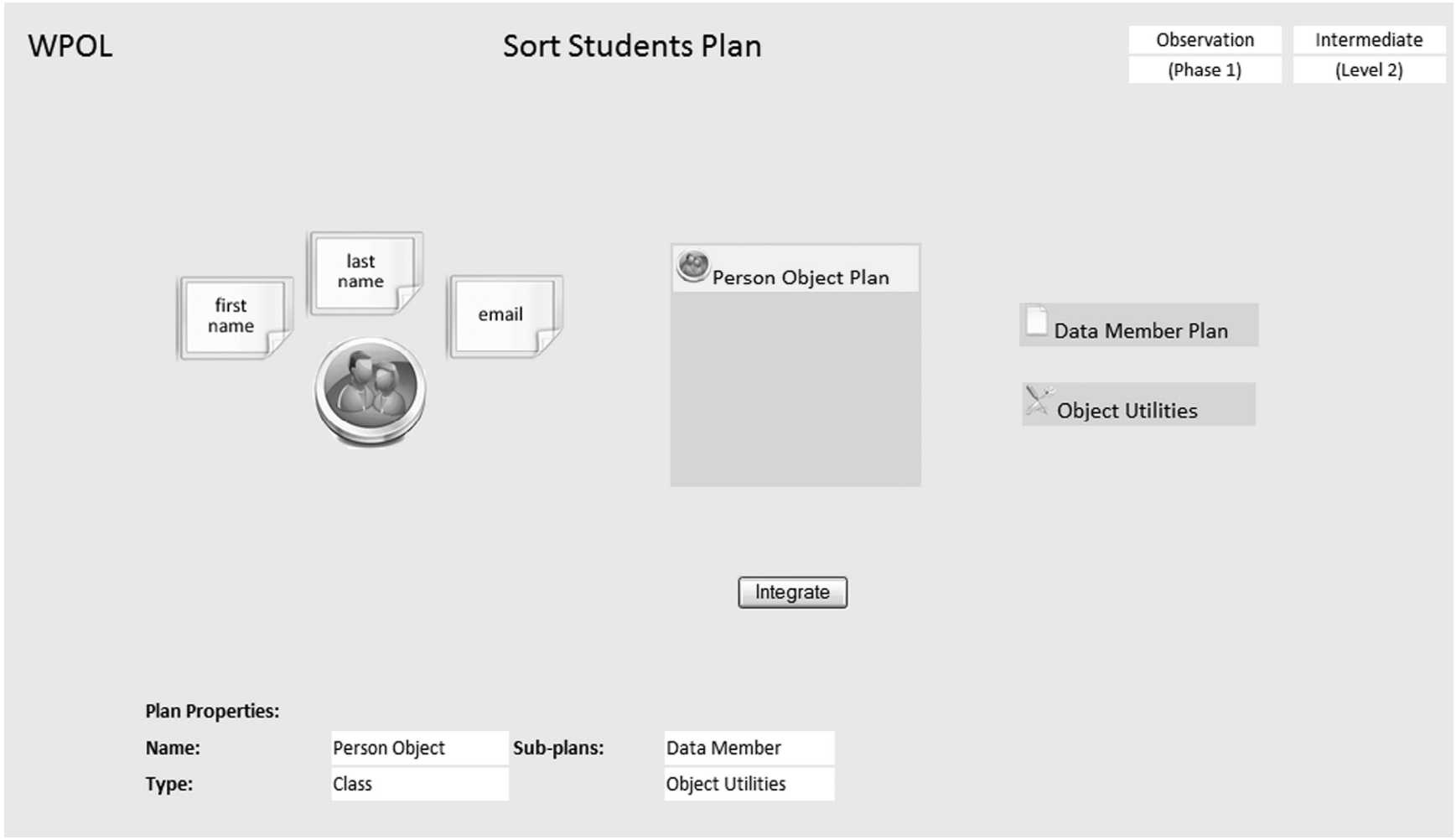

Figure 4.17 - Person Object Plan Integration View

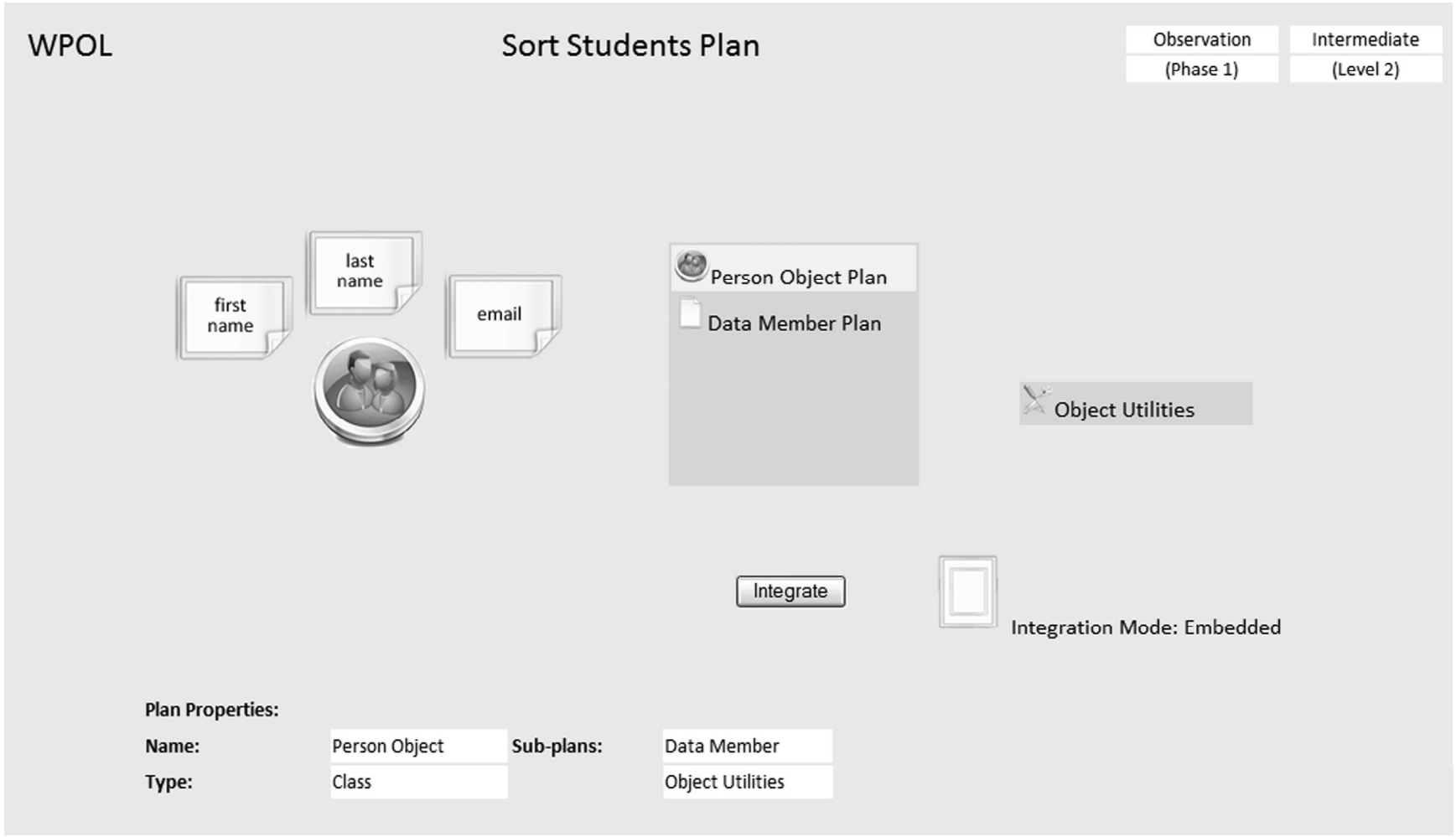

Figure 4.18 – Data Member Plan Embedded into Person Object

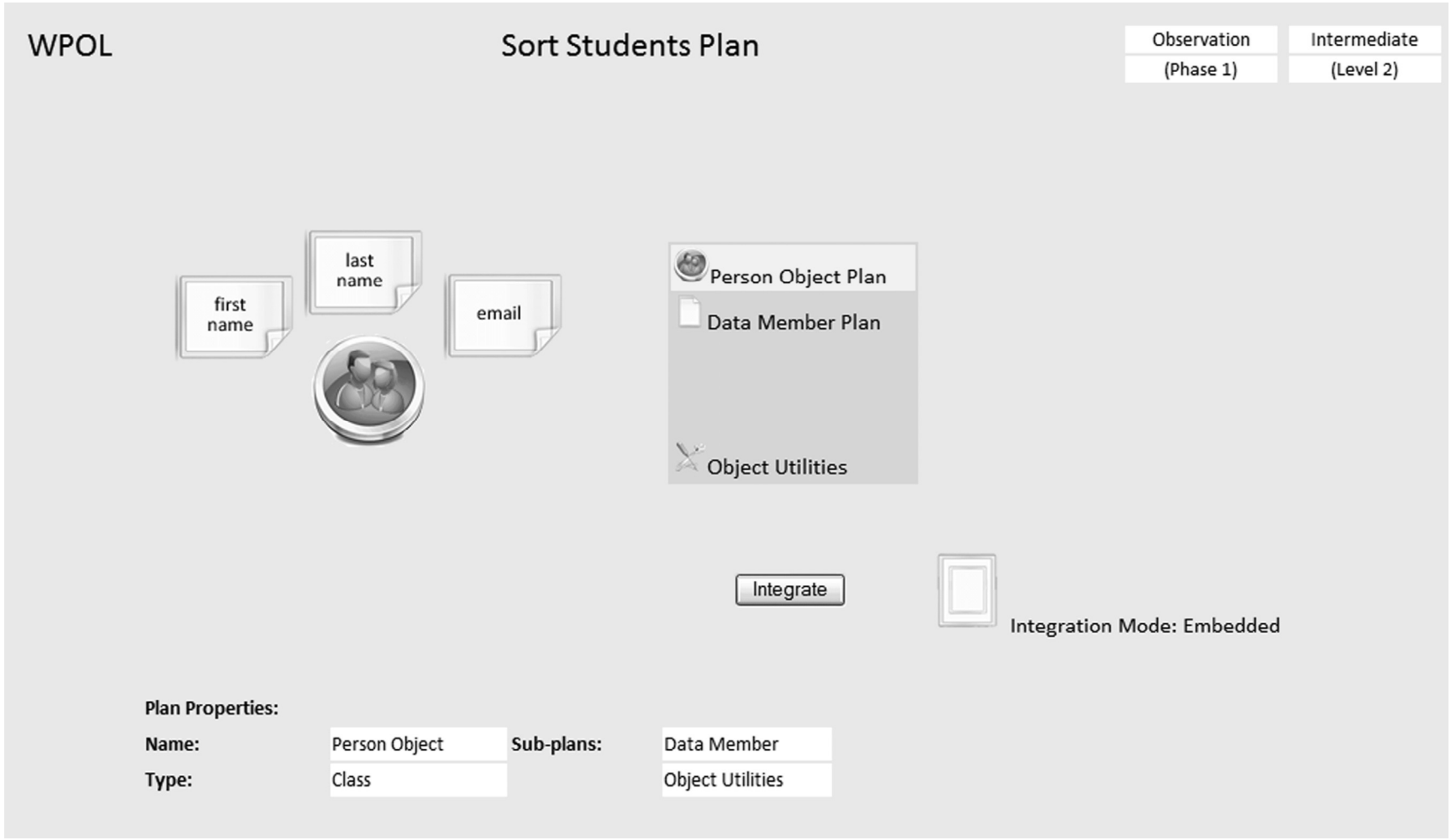

Figure 4.19 - Person Object with Data Member and Object Utilities Sub-plans

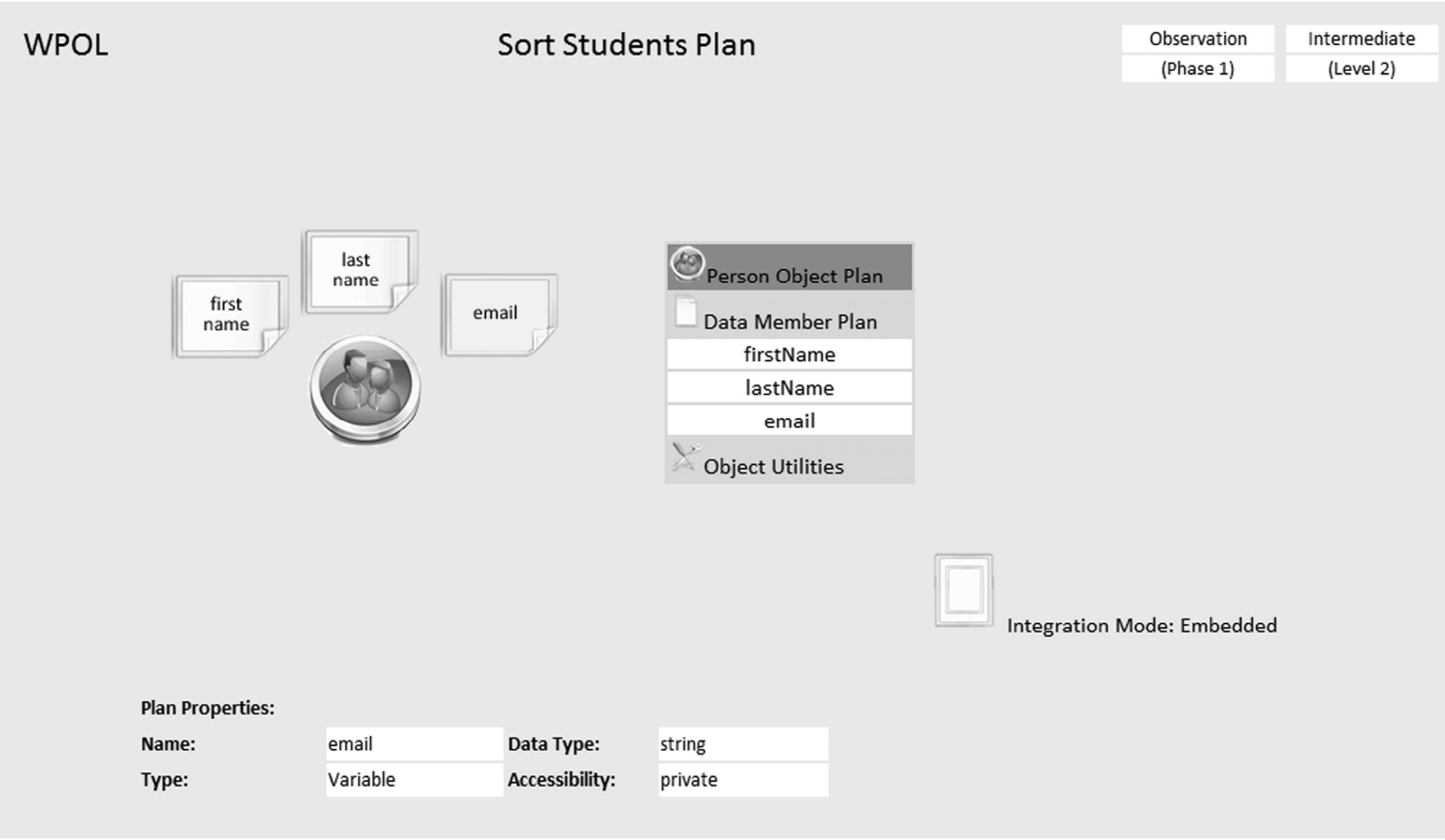

Figure 4.20 - Integration of Variable Sub-Plans into Data Member Plan

Now that the Person Object Plan has been designed, the Student Object Plan is loaded and the inheritance (IS-A) relationship is established. (Figure 4.21) The Student Object Plan, by inheriting the Person Object Plan, now includes the required attributes it provides.

Figures 4.22-4.24 demonstrate the implementation of the Person Object Plan as a Person class. Each Variable in Data Member is implemented according to its Plan Properties. The Object Utilities (Type: Class Component, Sub-plans: Constructor, Destructor, Set Plan, Get Plan) are also implemented. The Set Plan includes a set function for each Variable in Data Member to enable setting a value. The Get Plan consists of get functions for each Variable in Data Member to provide access to the variable's value. The code for establishing the inheritance (IS-A) relation between Student and Person is then added, public inheritance by default.

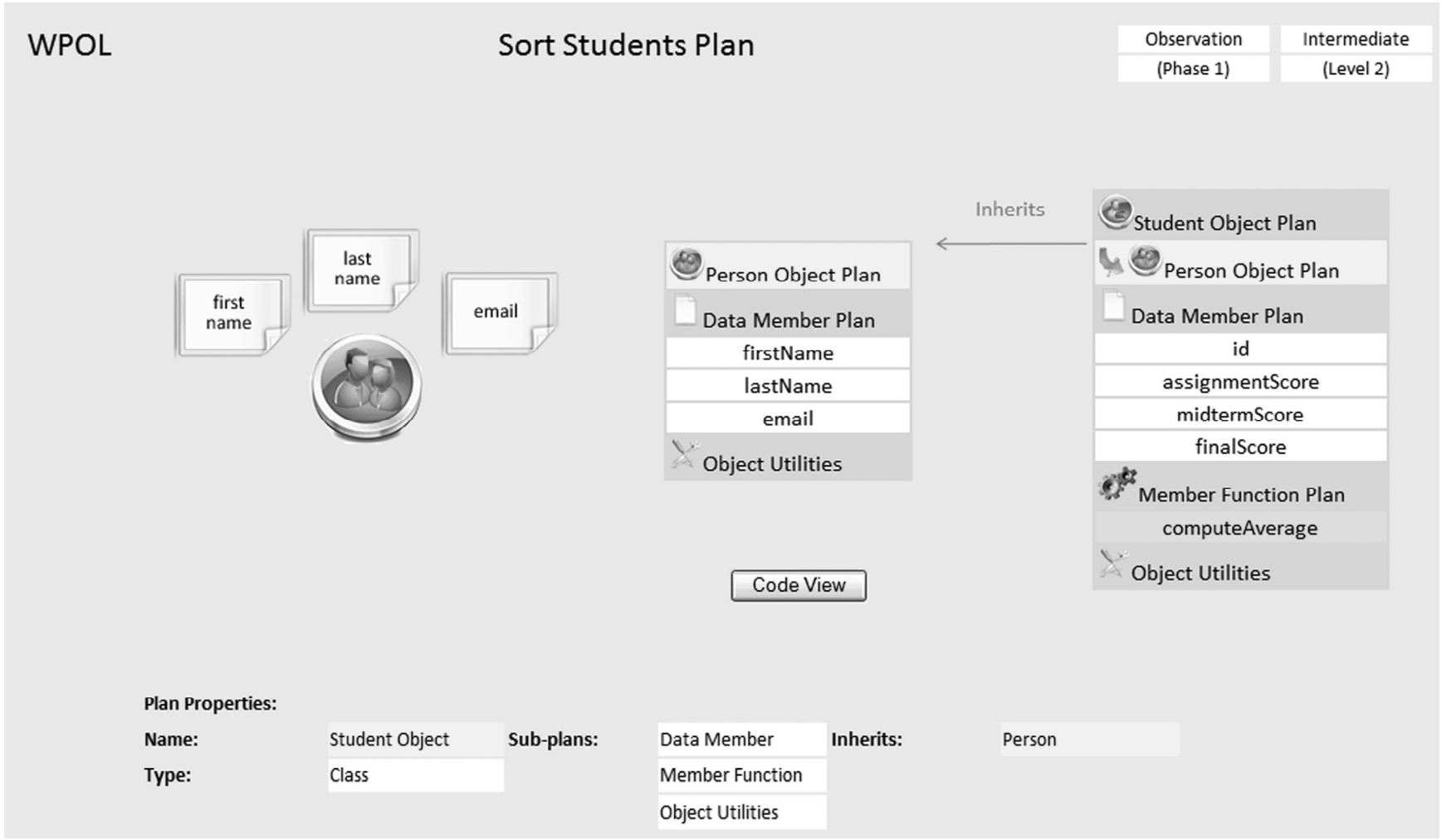

Figure 4.21 – Inheritance (IS-A) Relationship between Student and Person

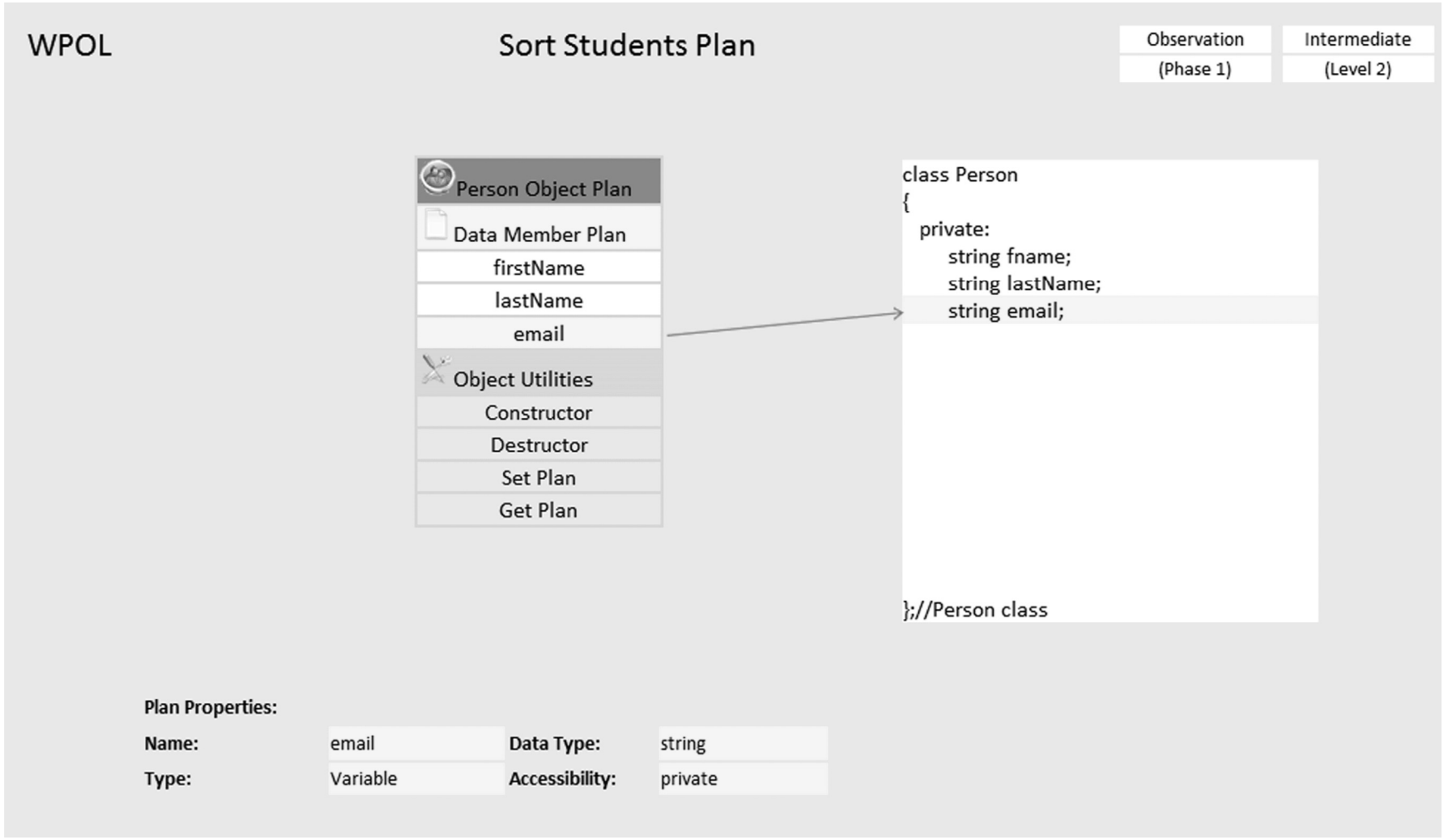

Figure 4.22 - Implementation of Variable Plans According to Plan Properties

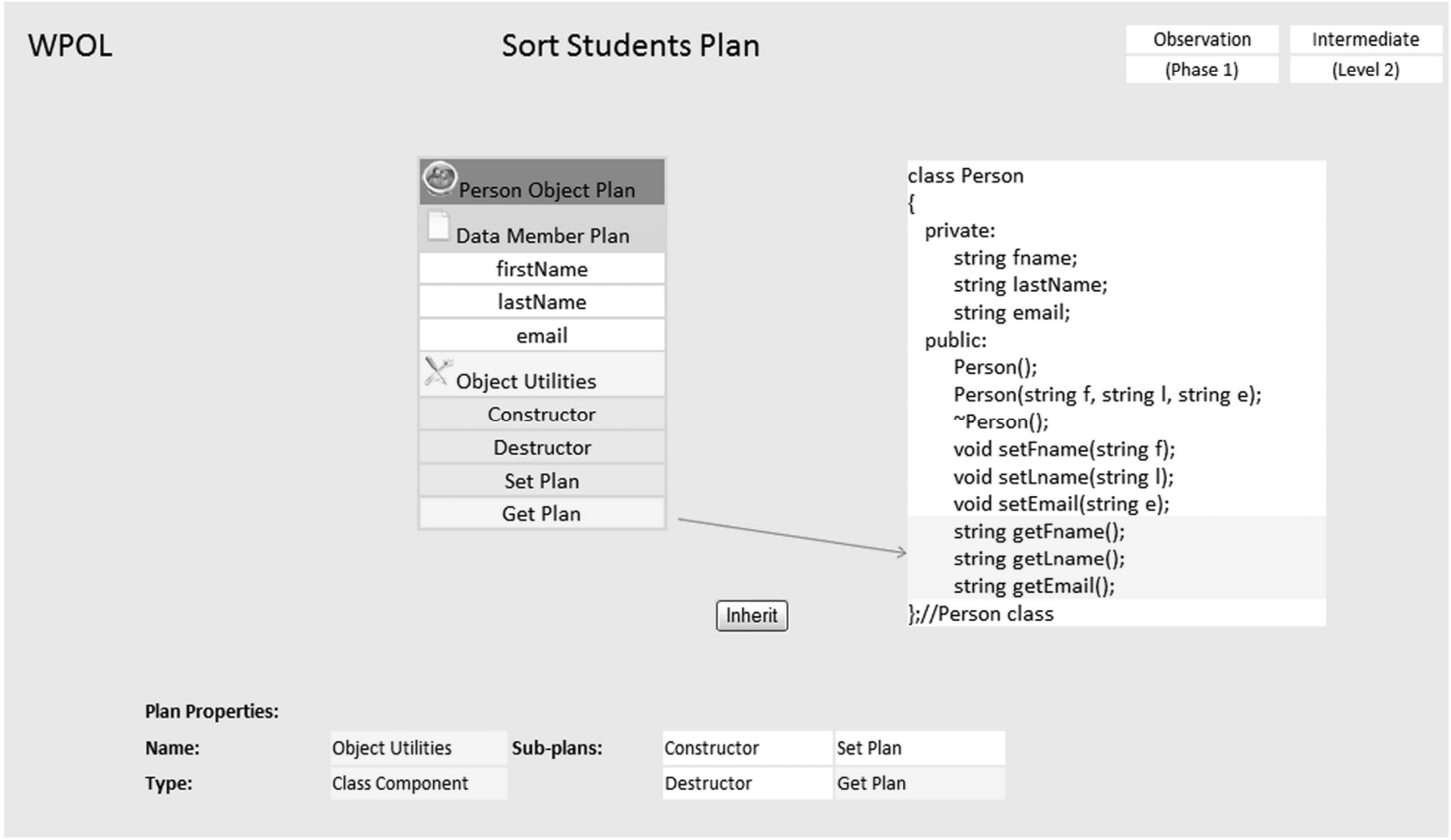

Figure 4.23 - Implementation of Object Utilities

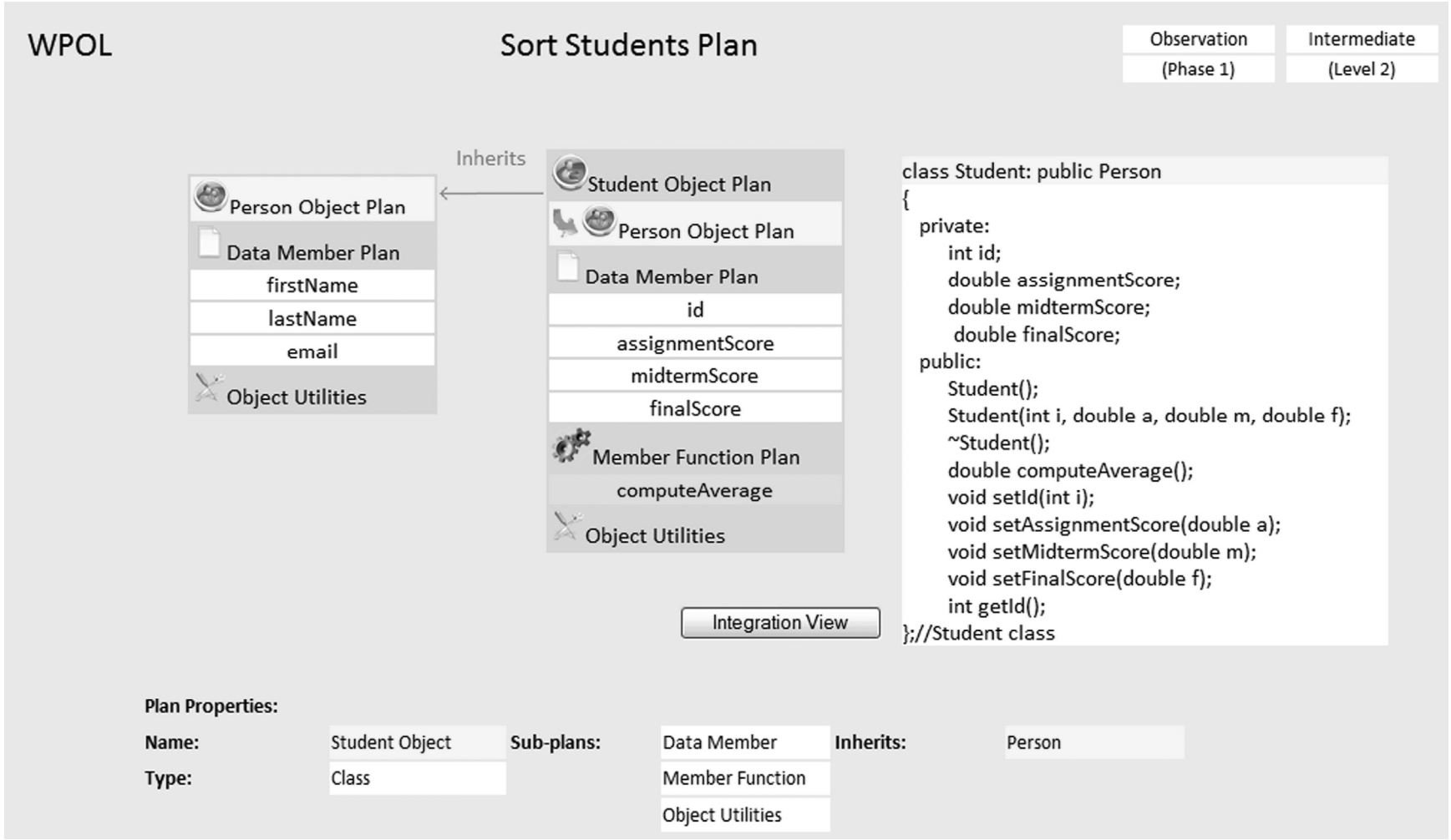

Figure 4.24 - Implementation of Inheritance (IS-A) Relationship

The next step is the Integration View for the Sort Students Main Plan. The Student Object Plan and the Sort Students Main Plan are considered Interleaved since the Student Object Plan will be used at several points throughout main. An array of objects, Student Array is included since this program requires several students to be sorted. (Figure 4.25) The Sort Students Main Plan is of type Main Function and has sub-plans: Loop, Bubble Sort, Input Student, and Display Student. The program necessitates a loop to input the Student Array; therefore, a Loop Plan is embedded into the Sort Students Main Plan. (Figure 4.26) This Loop Plan contains Input Student Plan [Mode: Console] as an embedded sub-plan. Input Student utilizes the Set Plan from Student Object. Since the program description specifies that the students are to be sorted according to their average, the Bubble Sort Plan is appended to the input loop. (Figure 4.27) The final task of main is to display the sorted student information. A second Loop Plan is then appended to the Bubble Sort Plan. The Display Student Plan [Mode: Console], which utilizes Get Plan and computeAverage from the Student Object Plan, is embedded into this Loop Plan. (Figure 4.28)

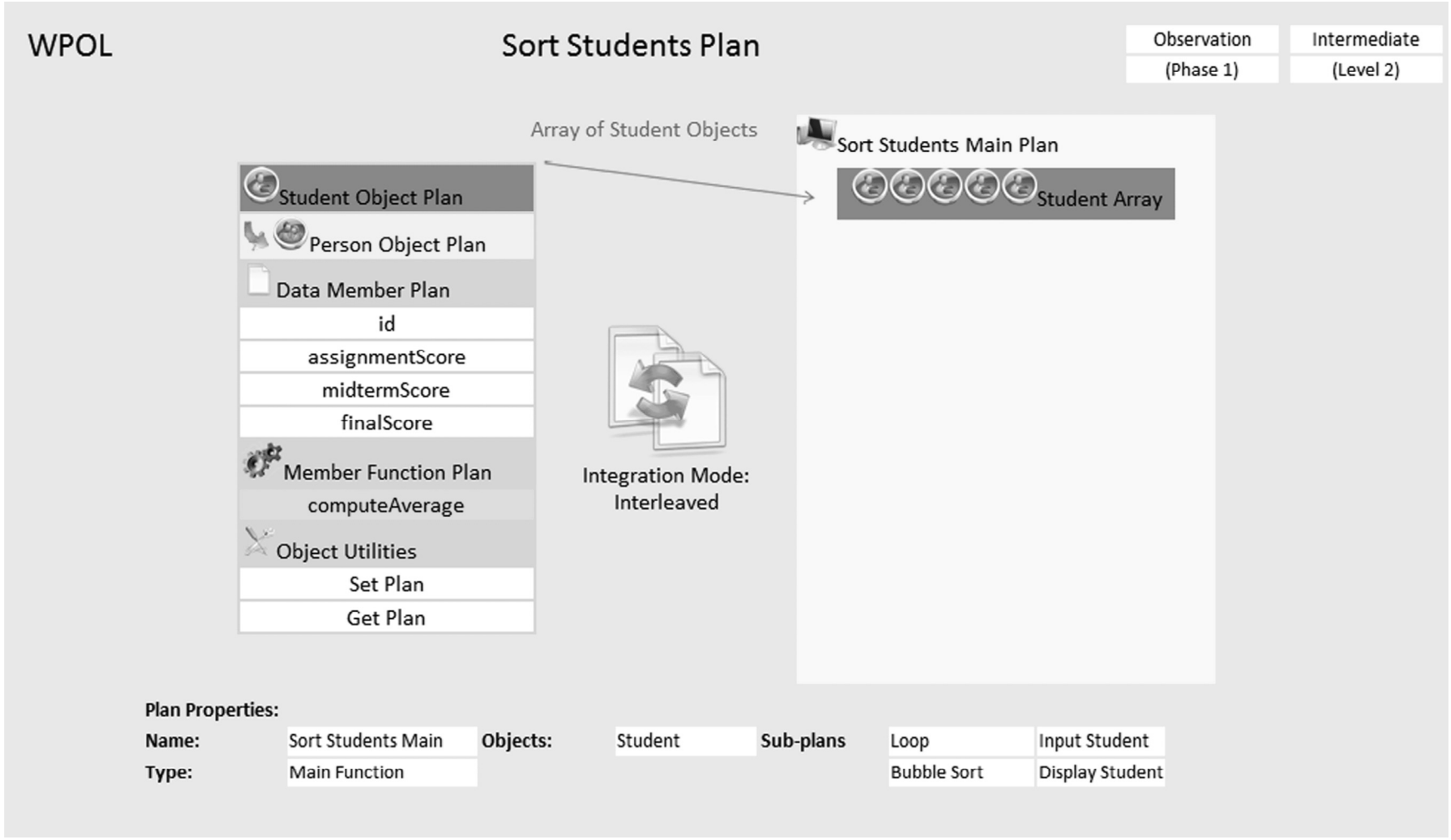

Figure 4.25 - Sort Students Main Plan and Student Array

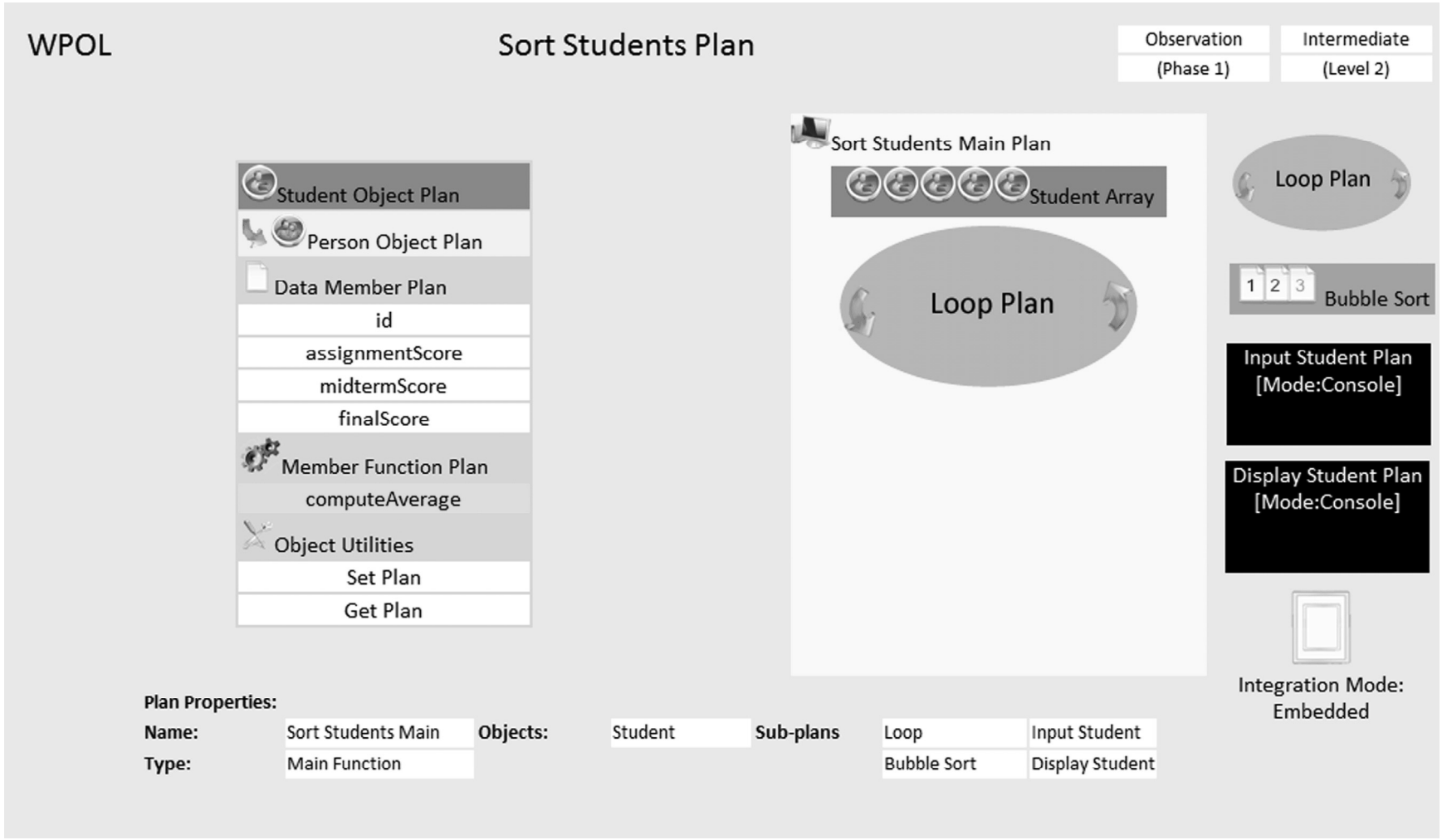

Figure 4.26 – Integration of Loop Plan (Embedded)

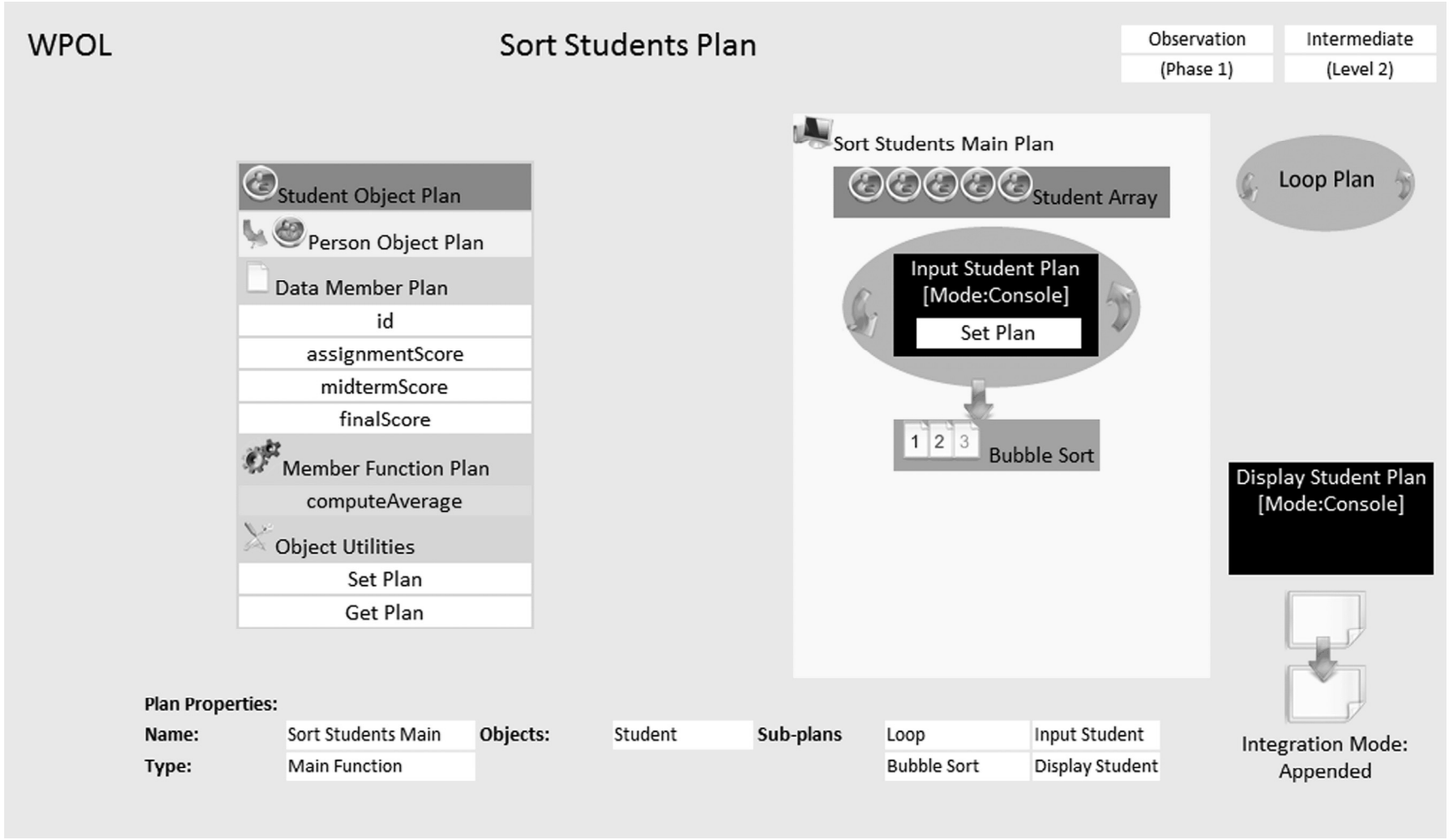

Figure 4.27 - Integration of Bubble Sort Plan (Appended)

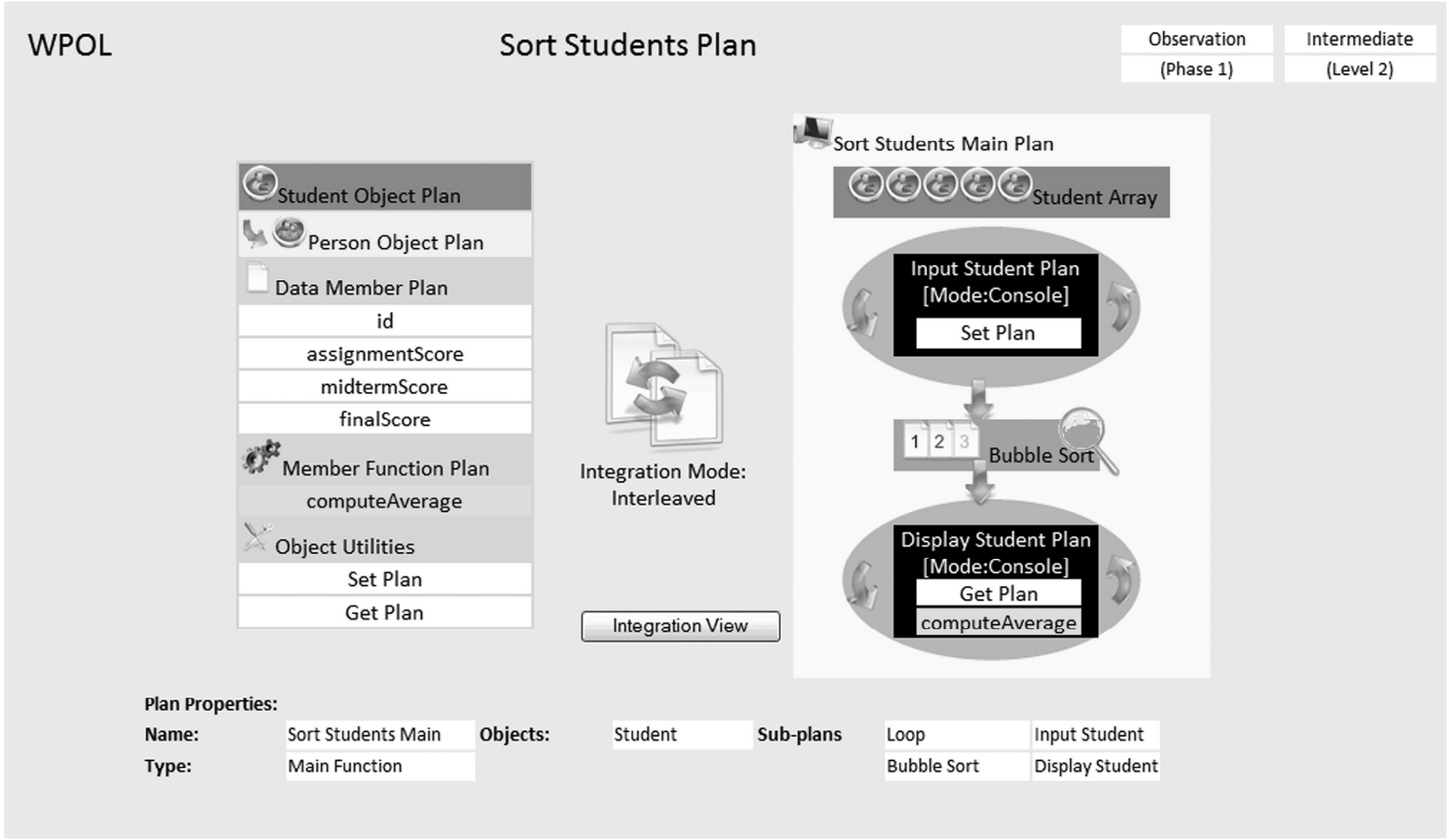

Figure 4.28 – Sort Students Main Plan

The observation will then zoom into the Integration View for the Bubble Sort Plan. Bubble sort is a sorting algorithm in which smaller values gradually “bubble” up toward the top of the array and larger values sink to the bottom. The algorithm scans through an array of numbers and compares each adjacent pair. If the adjacent numbers are not in order, they are swapped. This is repeated for n passes through the data; n being the number of items to be sorted. Bubble sort has a worst case complexity of $O(n^2)$ [Knuth 97]. The Bubble Sort Plan is of type Algorithm / Problem Solving and consists of sub-plans: Loop, Compare Adjacent, Swap, and Null. The Pass Loop is embedded into the Bubble Sort Plan. The Scan Loop is then embedded into the Pass Loop. Compare Adjacent is embedded into the Scan Loop and is then branched into Swap and Null. Figures 4.29-4.35 show the Integration View and demonstration of the Bubble Sort Plan.

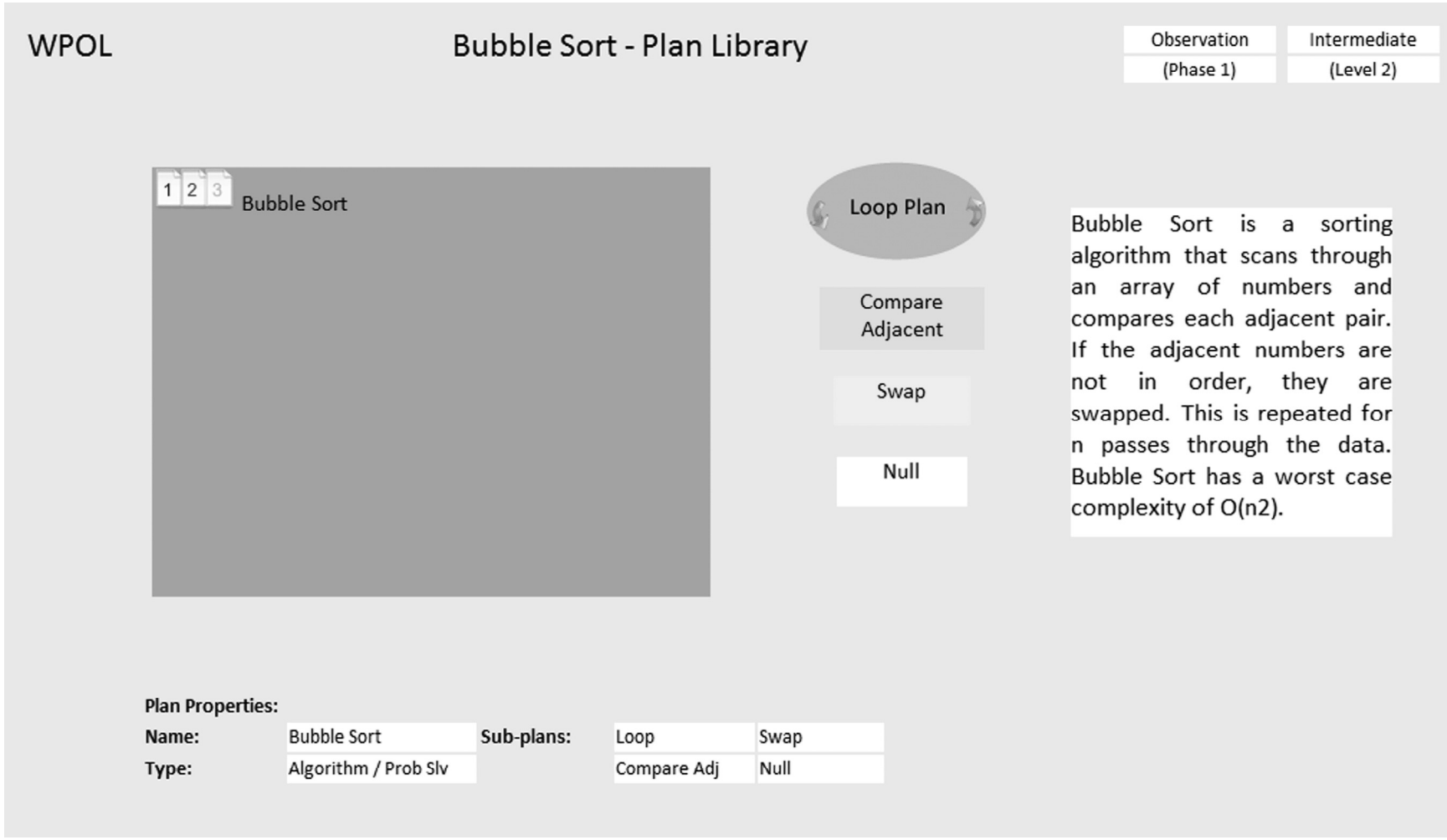

Figure 4.29 – Bubble Sort – Plan Description

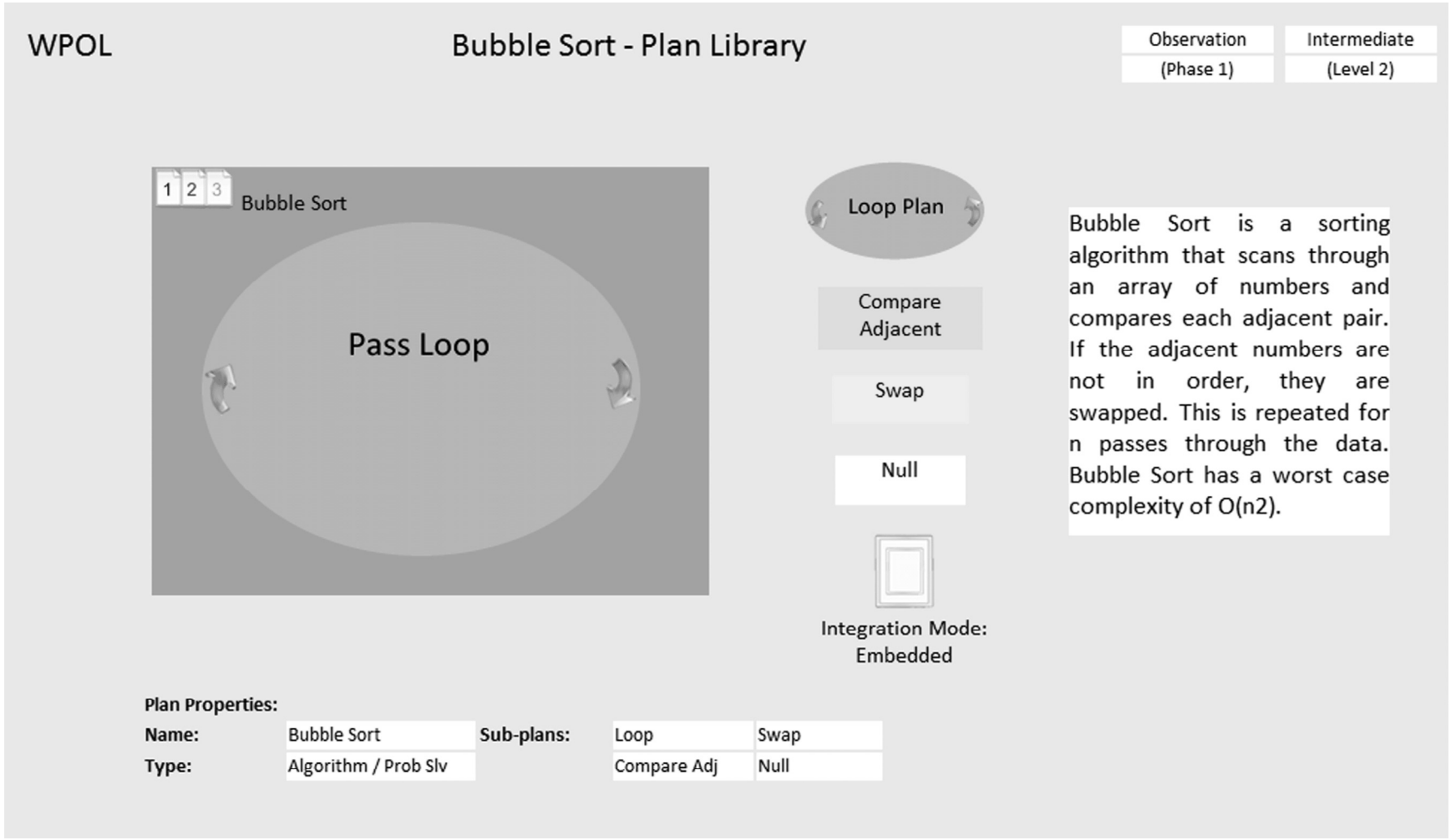

Figure 4.30 – Bubble Sort - Integration of Pass Loop (Embedded)

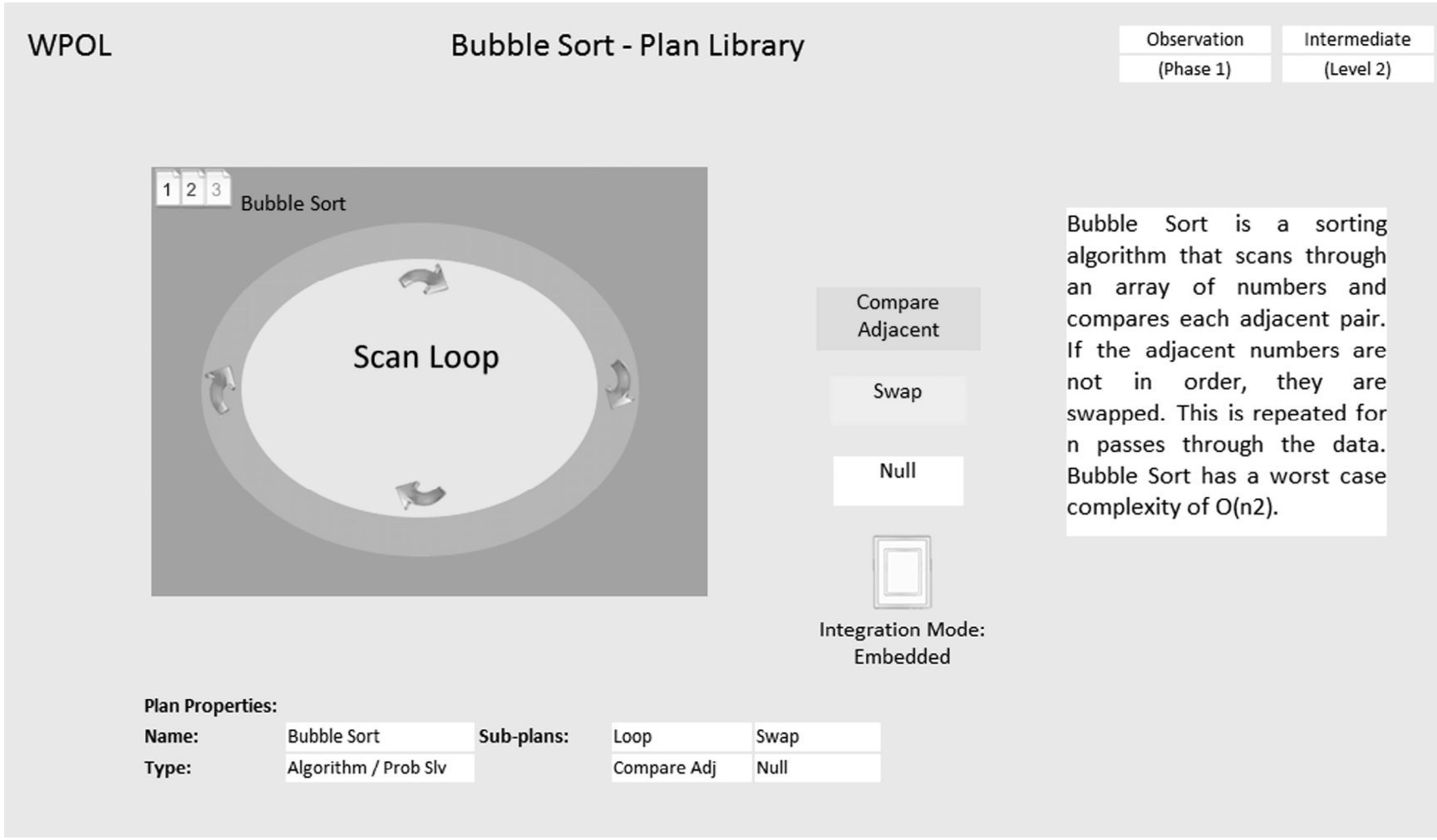

Figure 4.31 – Integration of Scan Loop (Embedded)

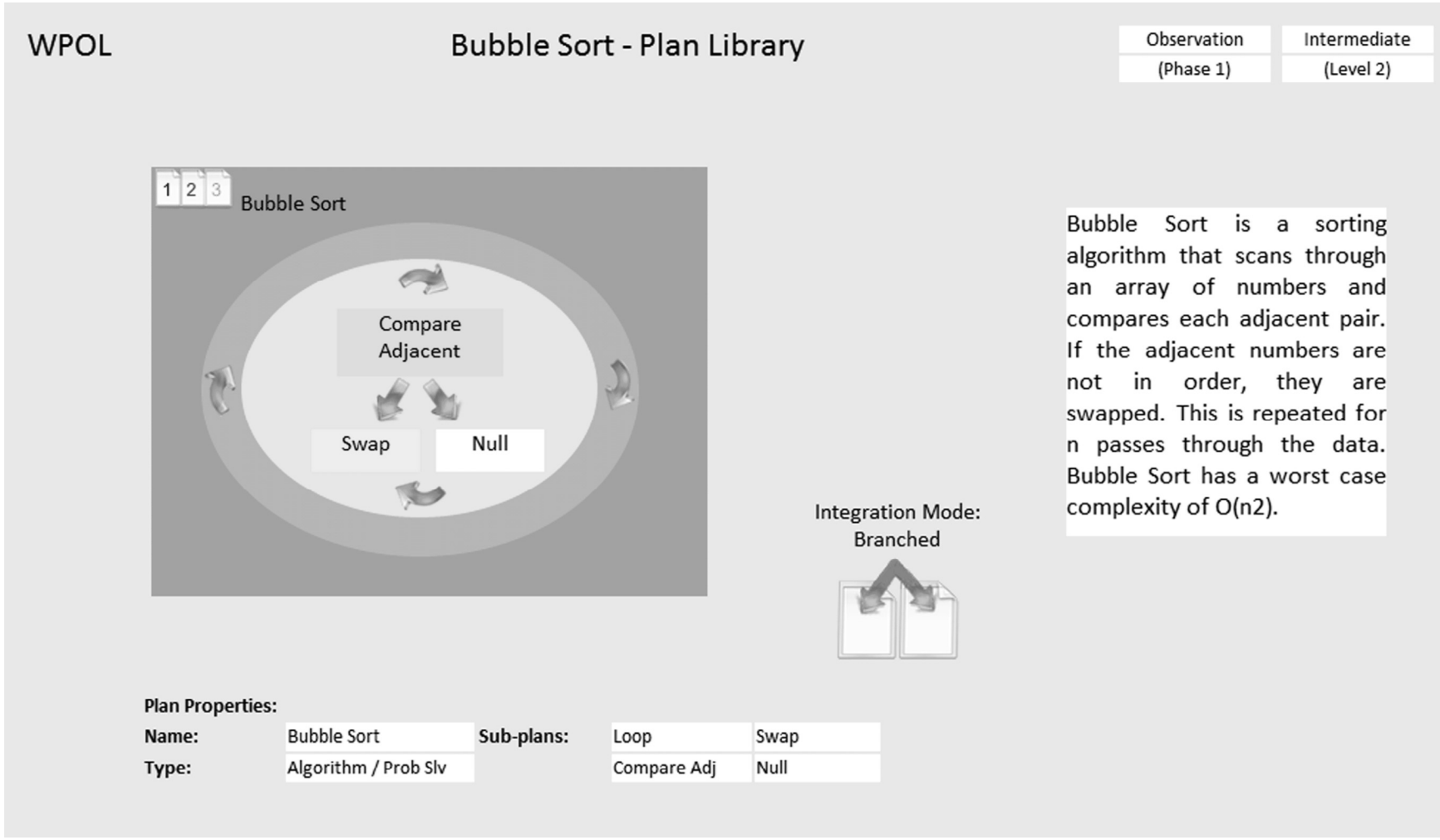

Figure 4.32 – Integration of Swap and Null (Branched)

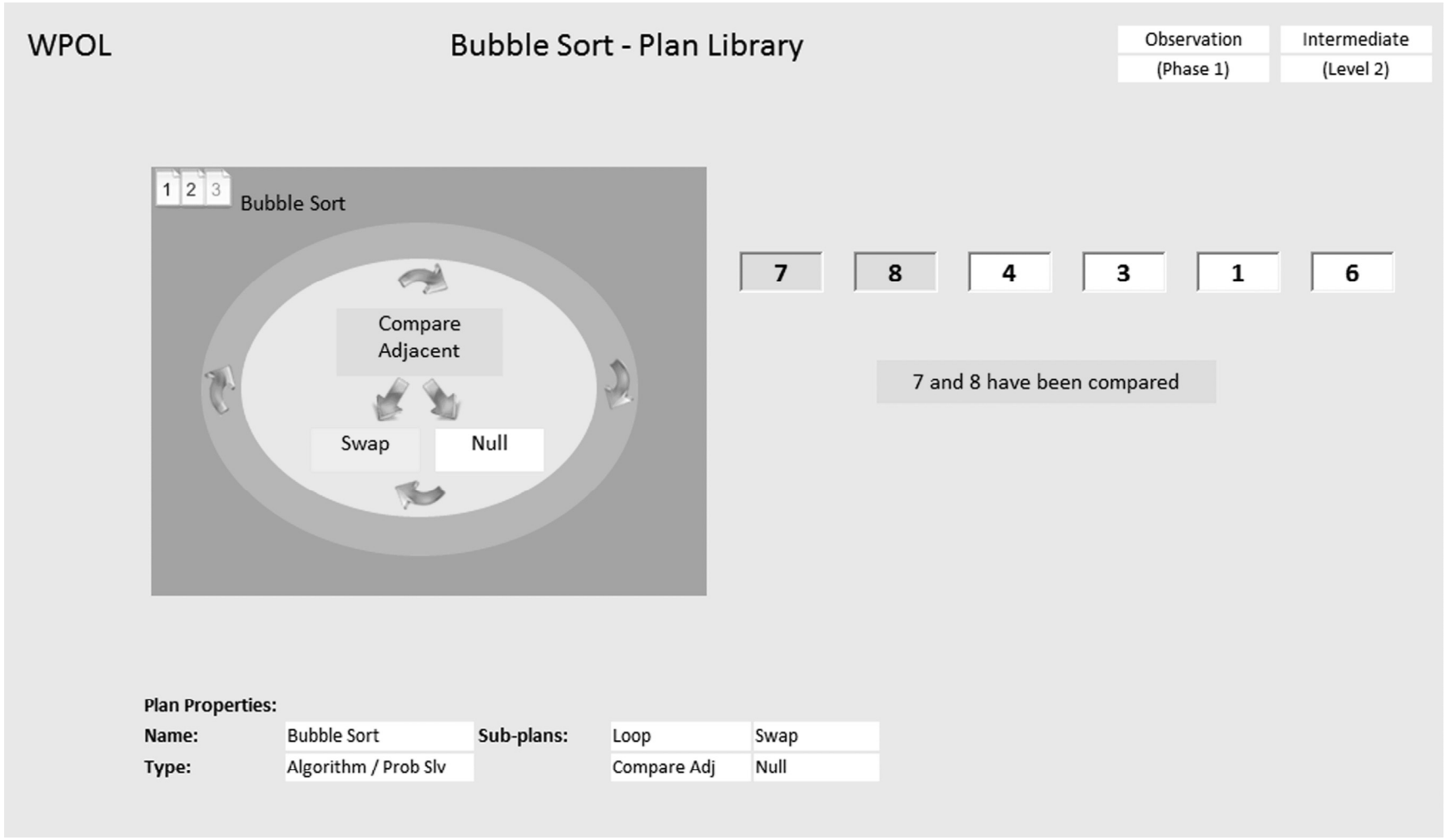

Figure 4.33 – Compare Adjacent Plan Demonstration

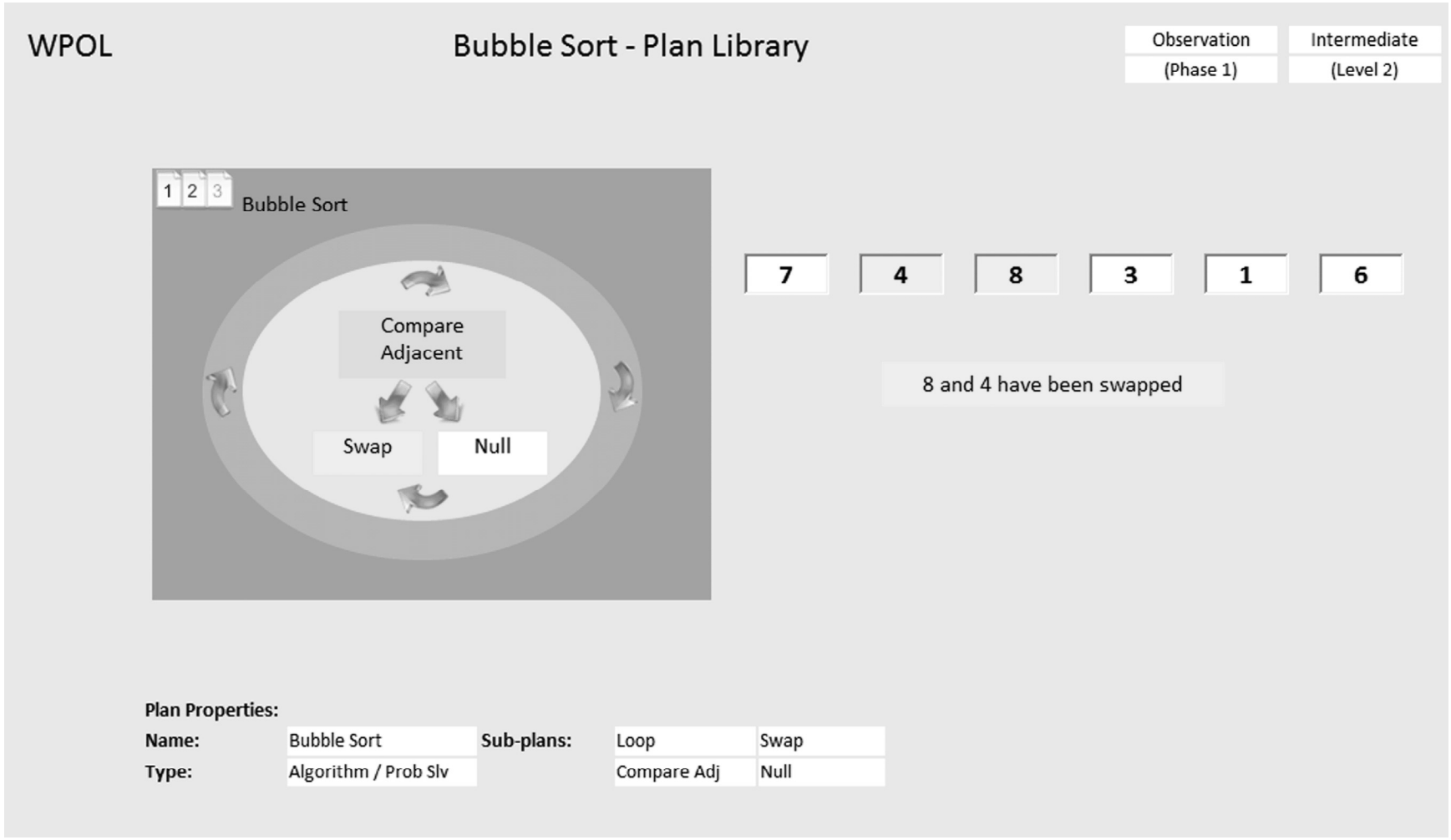

Figure 4.34 – Swap Plan Demonstration

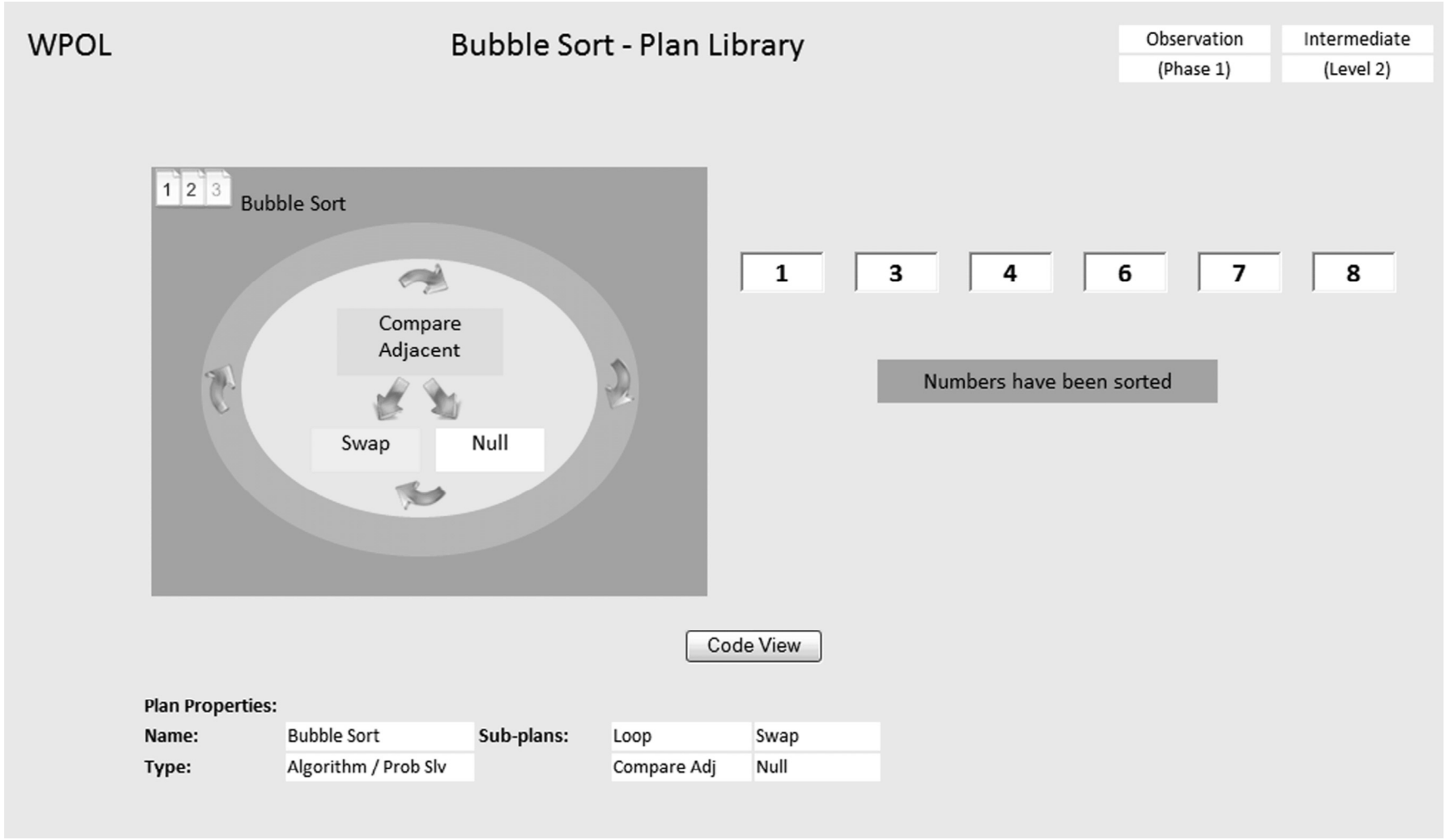

Figure 4.35 – Bubble Sort Demonstration

The next step in the observation is generation of the code for the Sort Students Main Plan. (Figures 4.36-4.38) Each plan is implemented according to their properties. Student Array (Type: Language Construct, Elements: Students, Size: 5) is made up of 5 Student Objects. The input loop is then implemented according to: Loop Plan (Type: Language Construct, Loop Type: For, Iterations: 5, Sub-plans: Input Student). This Loop Plan is implemented as a for loop with 5 iterations. The Input Student Plan is embedded into the loop and includes the code for inputting the information for each Student Object. Since each element of Student Array is a Student Object, its Set Plan is used to initialize data members.

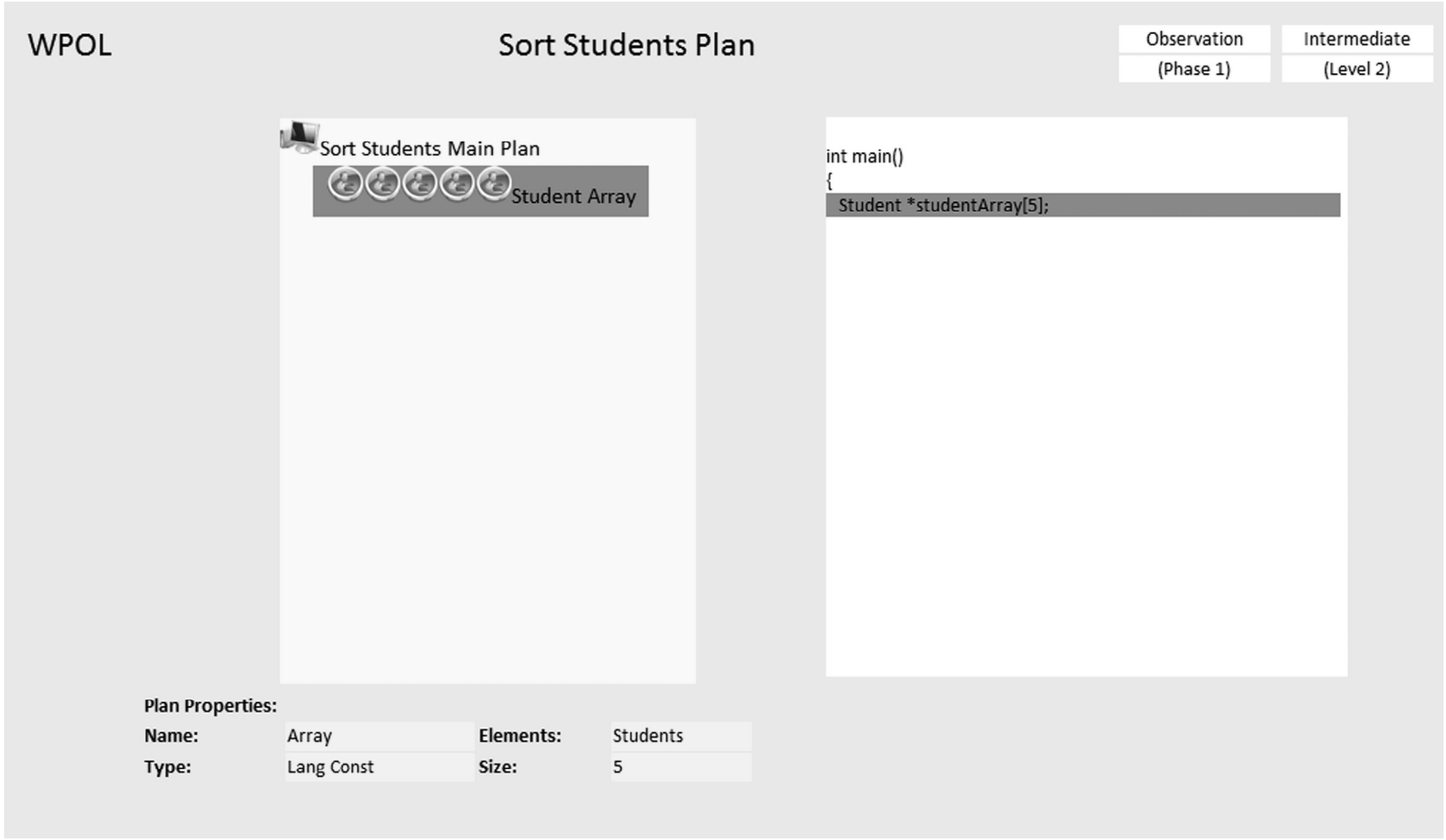

Figure 4.36 – Code View for Sort Students Main Plan

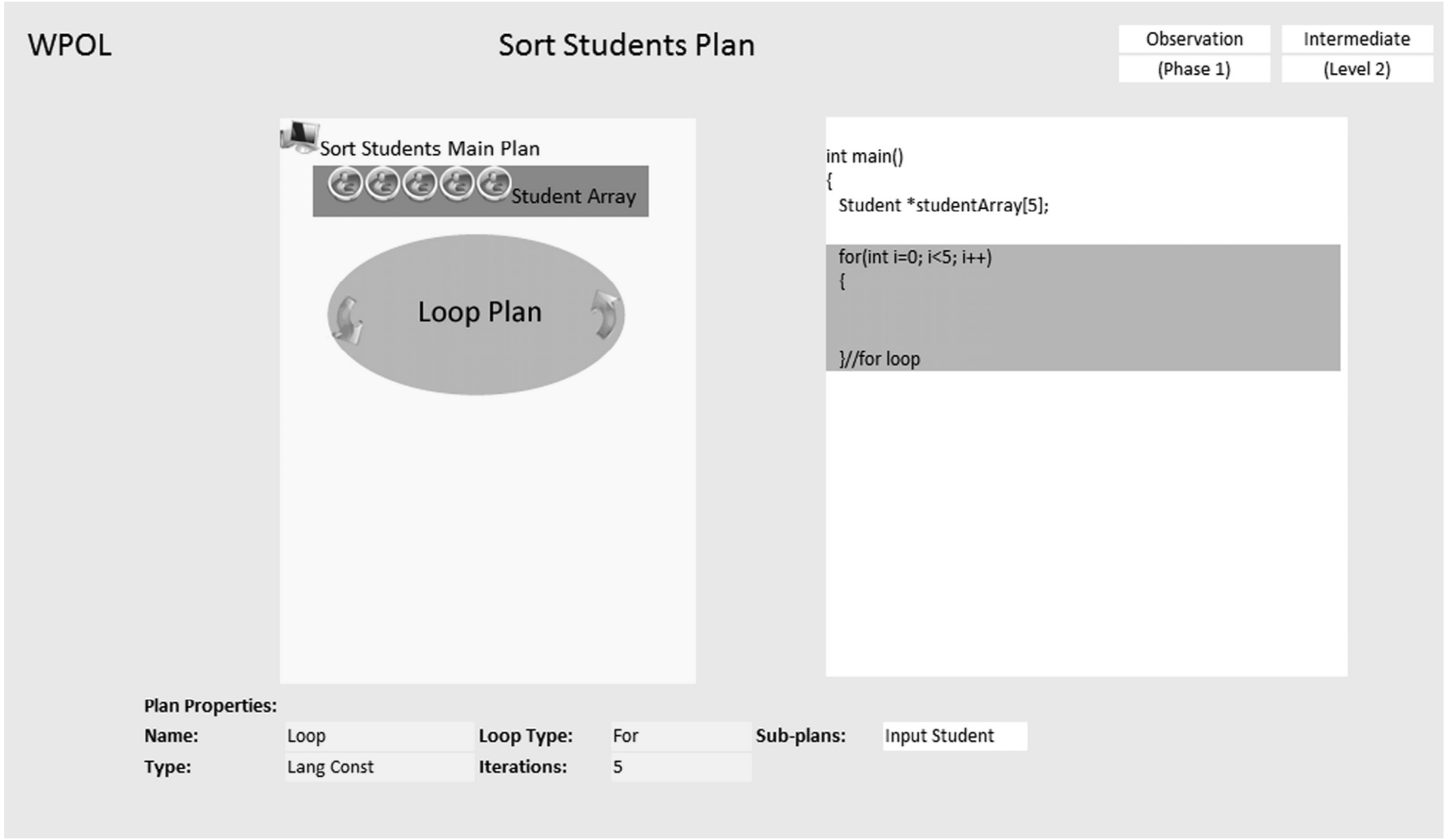

Figure 4.37 – Implementation of Loop Plan

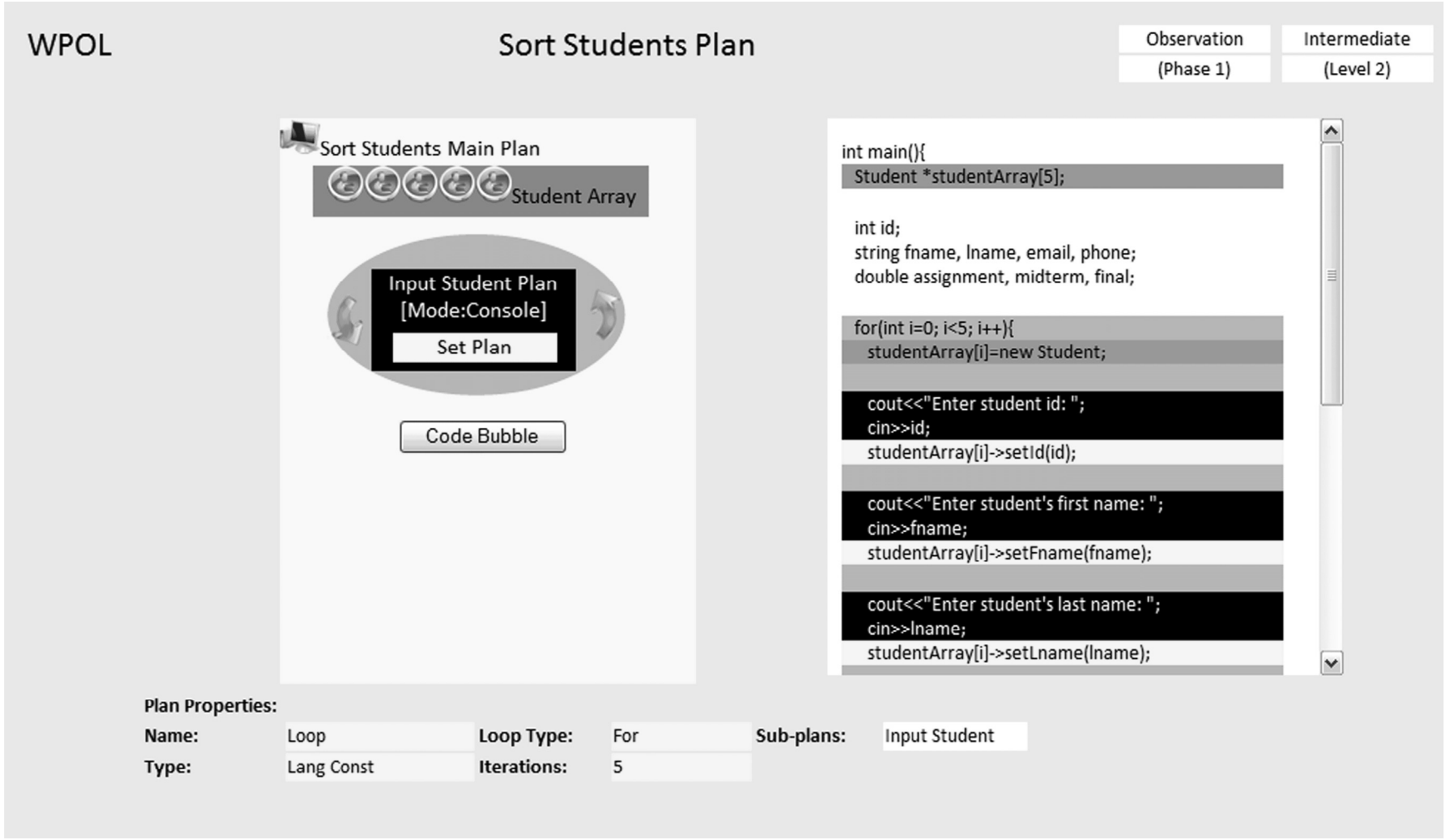

Figure 4.38 – Implementation of Input Student Plan

The Code View for the Bubble Sort Plan (Figure 4.39) demonstrates how its implementation is assembled based on the plan design. Pass Loop and Scan Loop are implemented as nested for loops. The outer loop (Pass Loop) will iterate 5 times (size of Student Array) and the inner loop (Scan Loop) will traverse the unsorted Students in Student Array upon each pass. Since the Students are to be sorted according to their average, Compare Adjacent will compare each pair of Students by the result of computeAverage. Since Compare Adjacent is Branched into Swap and Null, an if statement is used to implement the Branched relationship. Based on the result of the comparison of their averages, if the Students are not in order, they will be swapped; otherwise, no action will be taken.

After the Student Array is sorted, the Student information is to be displayed. The second Loop Plan traverses the Student Array and the embedded Display Student Plan [Mode: Console] utilizes Student's Get Plan and computeAverage to display the information of each Student. (Figure 4.40)

Figure 4.41 contains the complete program for the Sort Students Plan.

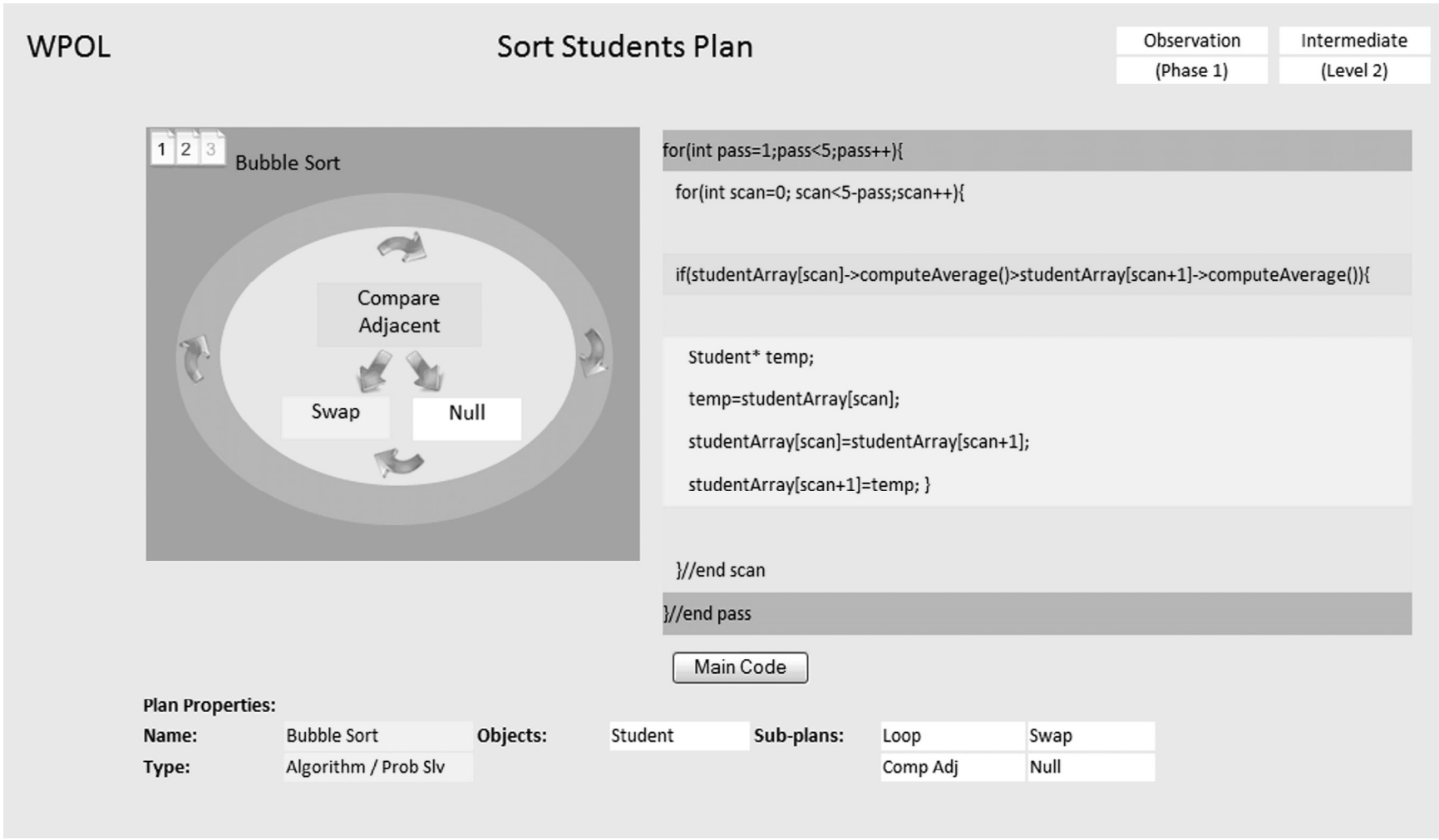

Figure 4.39 – Code View for Bubble Sort Plan

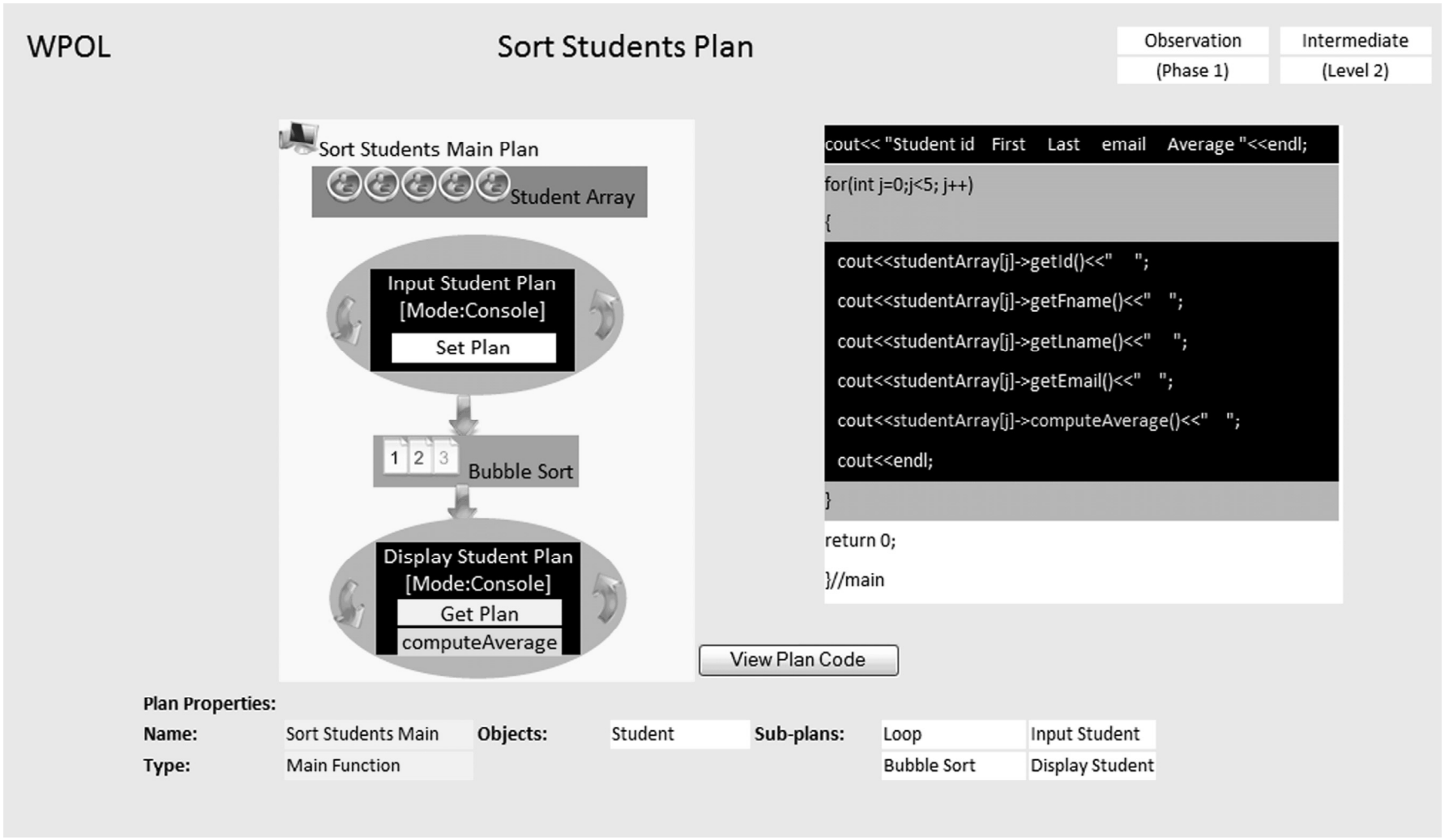

Figure 4.40 – Implementation of Loop Plan and Display Student Sub-plan

WPOL

Sort Students Plan

Observation	Intermediate
(Phase 1)	(Level 2)

```
#include<iostream>
#include<string>
using namespace std;

class Person
{
 private:
                string fname;
                string lname;
                string email;
 public:
                Person();
                Person(string f, string l, string a);
                ~Person();
                void setFname(string f);
                void setLname(string l);
                void setEmail(string e);
                string getFname();
                string getLname();
                string getEmail();
};//Person class

Person::Person()
{
        fname="";
        lname="";
        email="";
}//Person class default constructor

Person::Person(string f, string l, string e)
{
        fname=f;
        lname=l;
        email=e;
}//Person class contructor
```

```
Person::~Person(){}

void Person::setFname(string f)
{
        fname=f;
}//setFname function

void Person::setLname(string l)
{
        lname=l;
}//setLname function

void Person::setEmail(string e)
{
        email=e;
}//setEmail function

string Person::getFname()
{
        return fname;
}//getFname function

string Person::getLname()
{
        return lname;
}//getLname function

string Person::getEmail()
{
        return email;
}//getEmail function
```

```
class Student: public Person
{
 private:
          int id;
          double assignmentScore;
          double midtermScore;
          double finalScore;

 public:
        Student();
        Student(int i, double a, double m, double f);
        ~Student();
        void setId(int i);
        void setAssignmentScore(double a);
        void setMidtermScore(double m);
        void setFinalScore(double f);
        double computeAverage();
        int getId();
};//Student class

Student::Student()
{
    id=0;
    assignmentScore=0;
    midtermScore=0;
    finalScore=0;
}//Student class default constructor

Student::Student(int i, double a, double m, double f)
{
     id = i;
     assignmentScore = a;
     midtermScore = m;
     finalScore = f;
}//Student class constructor

Student::~Student(){}
```

```
void Student::setId(int i)
{
     id = i;
}//setId function

void Student::setAssignmentScore(double a)
{
     assignmentScore = a;
}//setAssignmentScore function

void Student::setMidtermScore(double m)
{
     midtermScore = m;
}//setMidtermScore funtion

void Student::setFinalScore(double f)
{
     finalScore = f;
}//setFinalScore function

double Student::computeAverage()
{
       double average;
       double sum;
       sum = 0;
       sum = assignmentScore + midtermScore + finalScore;
       average = sum / 3;
       return average;
}//computeAverage function

int Student::getId()
{
    return id;
}//getId function
```

```
int main()
{
        Student *studentArray[5];

        int id;
        string fname, lname, email, phone;
        double assignment, midterm, final;

        for(int i=0; i<5; i++)
        {
                studentArray[i]=new Student;

                cout<<"Enter student id: ";
                cin>>id;
                studentArray[i]->setId(id);

                cout<<"Enter student's first name: ";
                cin>>fname;
                studentArray[i]->setFname(fname);

                cout<<"Enter student's last name: ";
                cin>>lname;
                studentArray[i]->setLname(lname);

                cout<<"Enter student's email: ";
                cin>>email;
                studentArray[i]->setEmail(email);

                cout<<"Enter assignment score: ";
                cin>>assignment;
                studentArray[i]->setAssignmentScore(assignment);

                cout<<"Enter midterm score: ";
                cin>>midterm;
                studentArray[i]->setMidtermScore(midterm);

                cout<<"Enter final exam score: ";
                cin>>final;
                studentArray[i]->setFinalScore(final);
        }//for loop
```

```
    for(int pass=1;pass<5;pass++){

                for(int scan=0; scan<5-pass;scan++){

                        if(studentArray[scan]->computeAverage()>studentArray[scan+1]->computeAverage()){
                                Student* temp;
                                temp=studentArray[scan];
                                studentArray[scan]=studentArray[scan+1];
                                studentArray[scan+1]=temp; }
                            }//end scan}//end pass

      cout<< "Student id   First    Last    email    Average "<<endl;
      for(int j=0;j<5; j++){

                cout<<studentArray[j]->getId()<<"  ";
                cout<<studentArray[j]->getFname()<<"  ";
                cout<<studentArray[j]->getLname()<<"  ";
                cout<<studentArray[j]->getEmail()<<"  ";
                cout<<studentArray[j]->computeAverage()<<"  ";
                cout<<endl;}
    return 0;
}//main
```

Figure 4.41 - C++ Implementation for Sort Students Plan

4.2.2 Plan Integration

The Integration Phase tests a novice's ability to properly integrate plans to form a solution. The purpose of this phase is to reinforce concepts of plan integration and object design. The student is provided with a Plan description of a program to be completed. In this phase, the student is presented with plans from the Plan Library and asked to select which plan(s) should be integrated. The correct Integration Mode (Appended, Branched, Embedded, or Interleaved) must also be selected. This reinforces the students' understanding since they are taking an active role in creating the solution and incorrect solutions are explained.

The Payroll Plan is used to demonstrate a sample integration phase. This example tests the design of an Employee Object to be used in this plan and is shown in Figures 4.42-4.45.

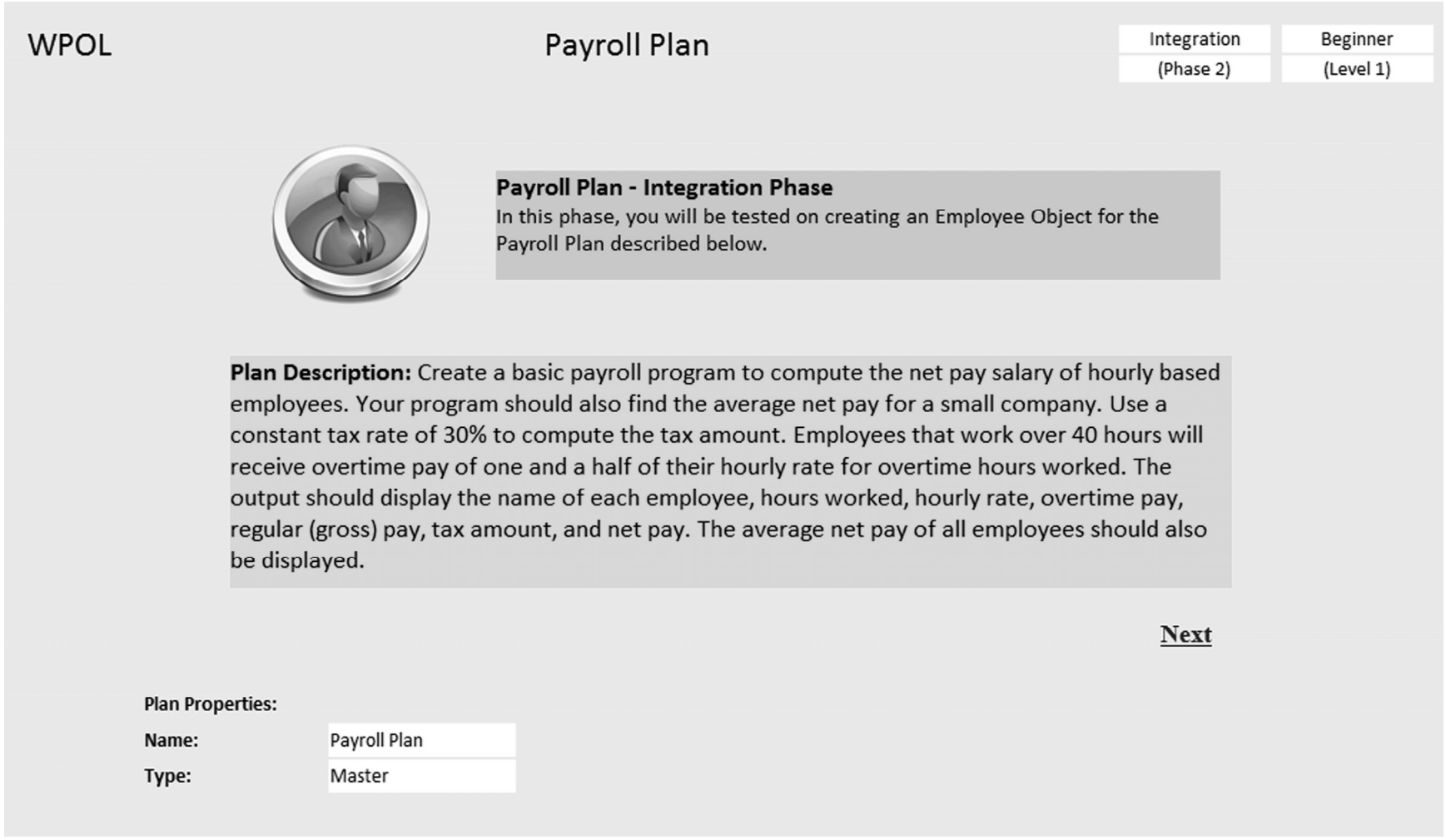

Figure 4.42 – Integration Phase – Payroll Plan Description

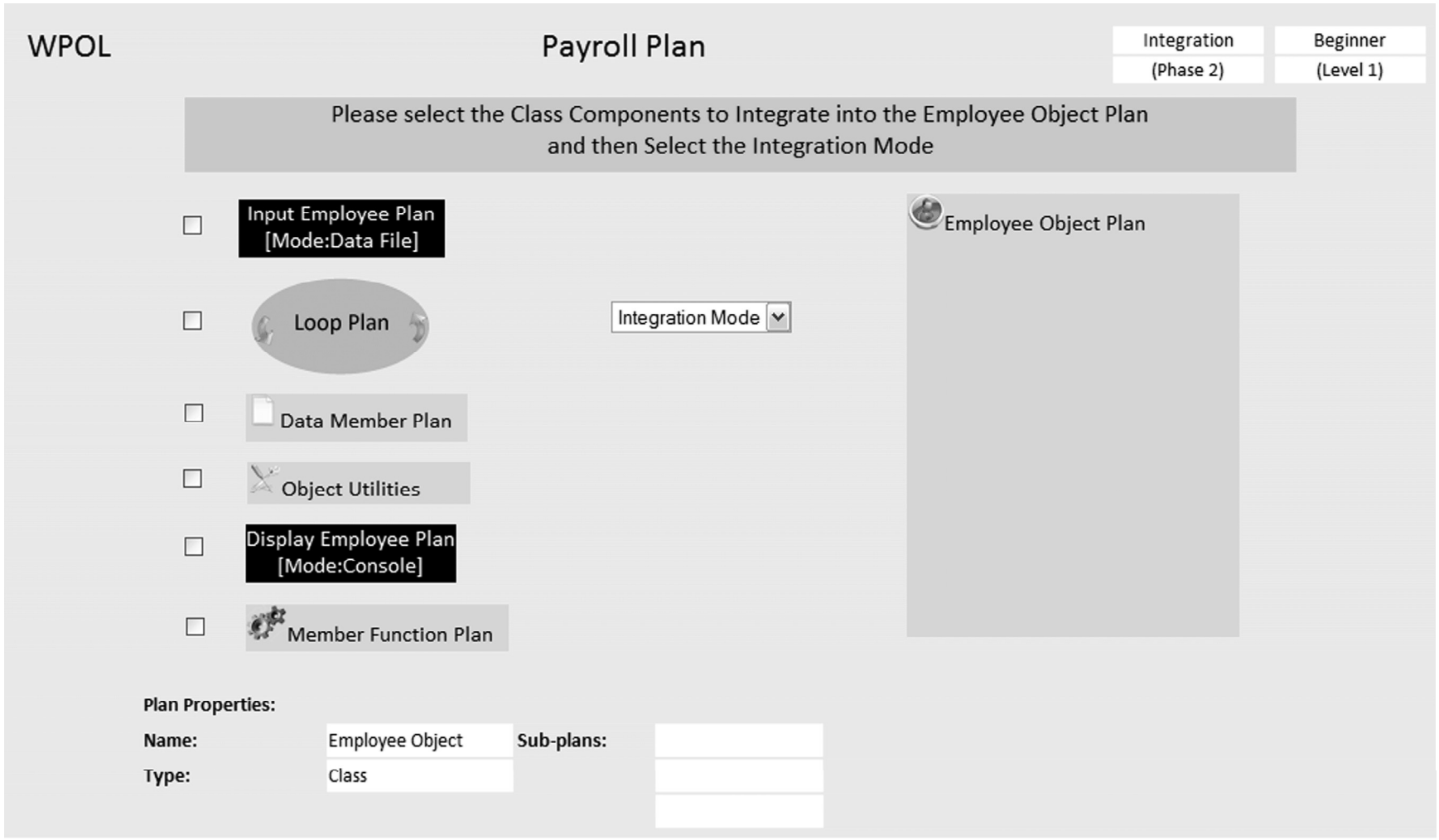

Figure 4.43 – Integration View for Employee Object Plan

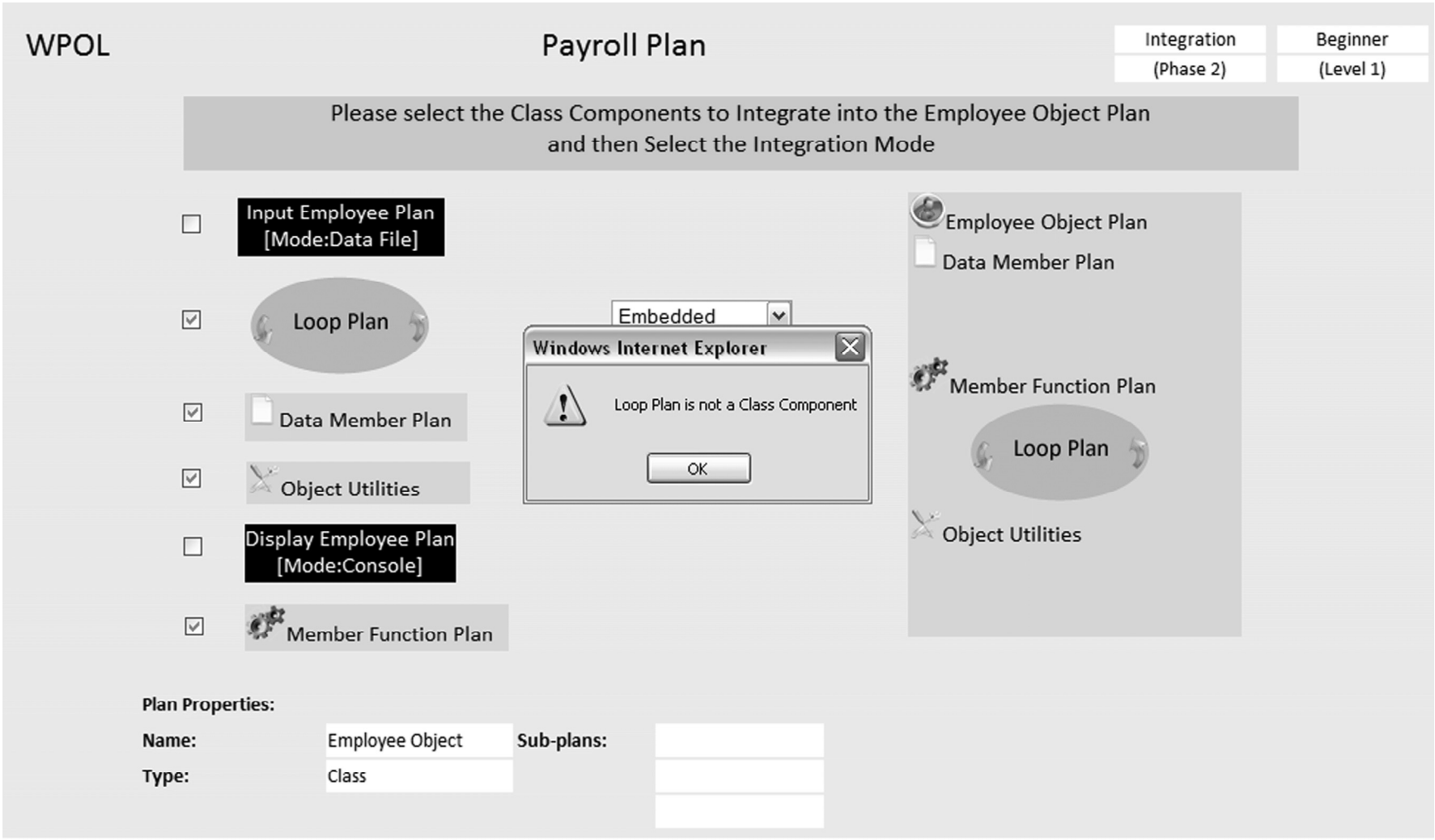

Figure 4.44 – Selection of Incorrect Plan

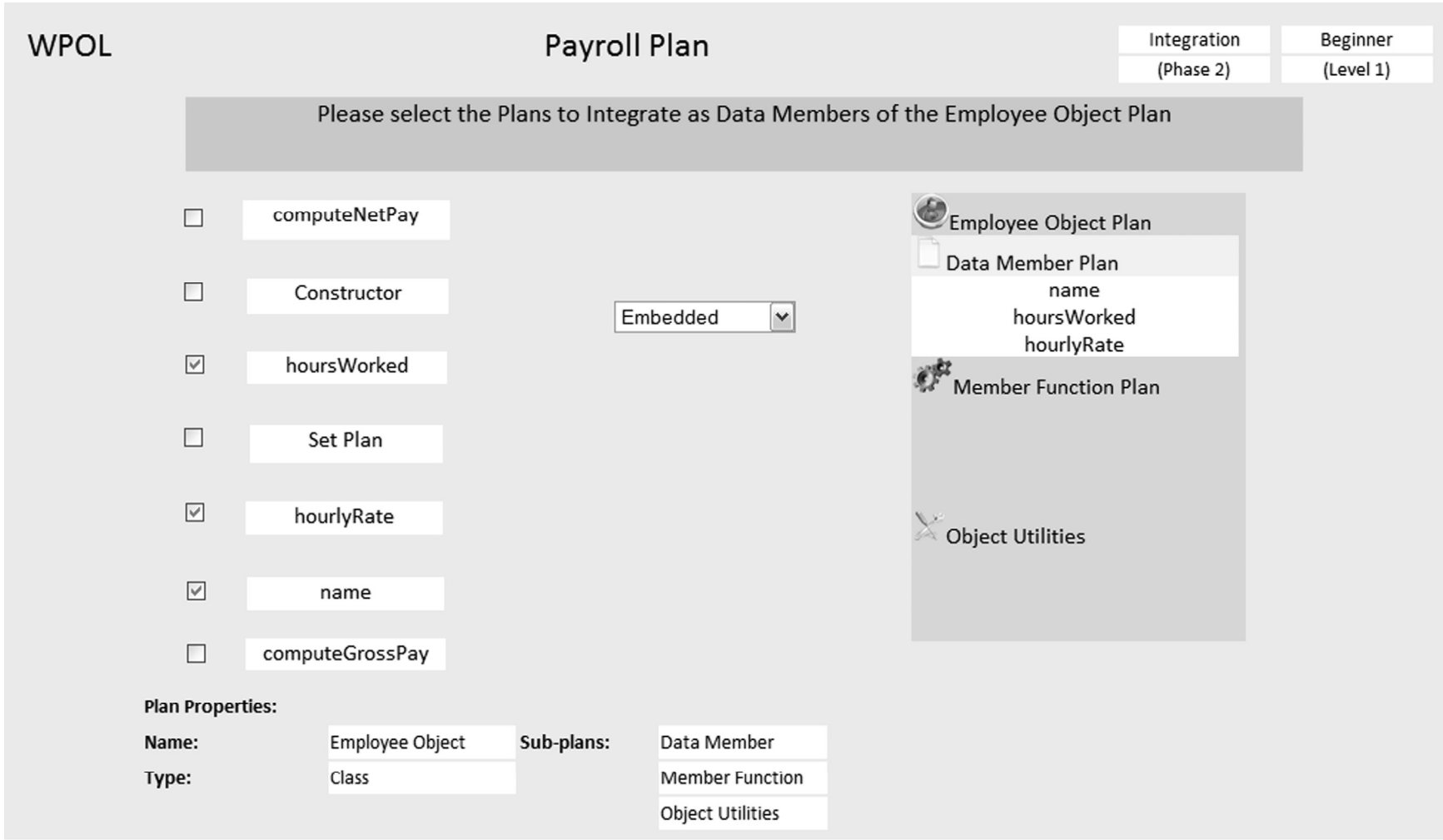

Figure 4.45 – Integration View for Data Member Plan

4.2.3 Plan Creation

In the Creation Phase, the student can customize Plans and design new Objects. Plans are customized by setting plan properties. This phase is beneficial to students to aide them in creating their own programs. For example, an Object can be created by setting the properties of an Object Plan, as well as the properties of its sub-plans. The creation of an Object Plan is demonstrated in Figures 4.46-4.49. In Figure 4.46, the name property of the Object Plan is set; in this example, a Book class is used.

The Book Object Plan consists of embedded Data Member and Member Function sub-plans. Since the Data Member Plan contains sub-plans of type Variable, the next step is to create and customize these Variable Plans. A Variable Plan has properties: Name, Type (integer, double, float, char, string, etc), and Accessibility (private, public, protected). In this example, title (string), ISBN (string), and price (double) are created as private data members. Setting of Variable Plan properties is shown in Figure 4.47. The Member Function Plan is made up of sub-plans of type Function. The Function Plans are created by setting the properties of Name, Return Type, Accessibility, and Parameter(s). To include a function such as double totalPrice(int quantity), the properties are set as shown in Figure 4.48. After all of the plans necessary for the Book Object are created, the Book class is implemented to include data members, member function prototypes. This is done by translating the plan properties into their code representation. The code for the Book class is shown in Figure 4.49.

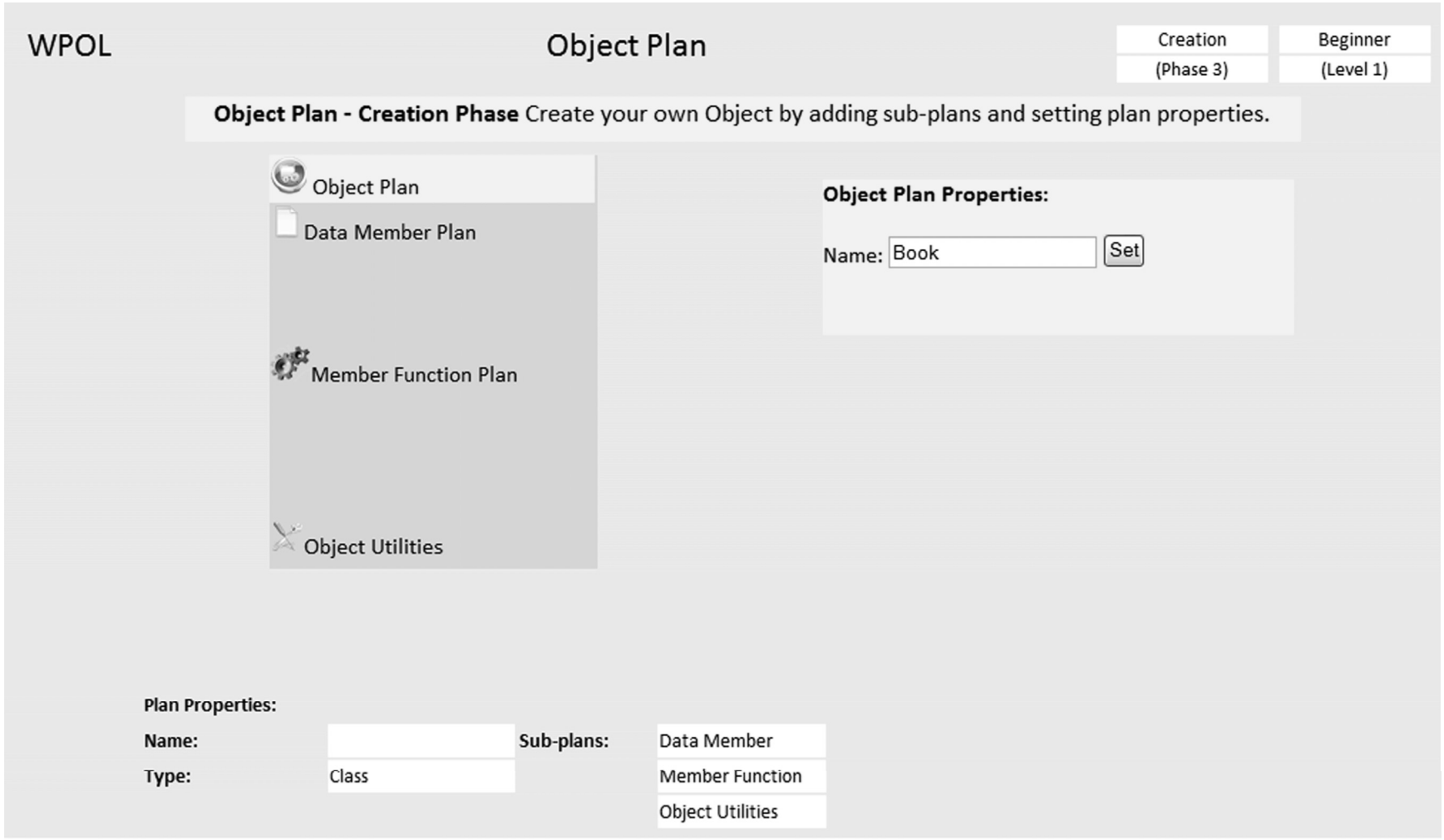

Figure 4.46 – Creation Phase – Object Plan

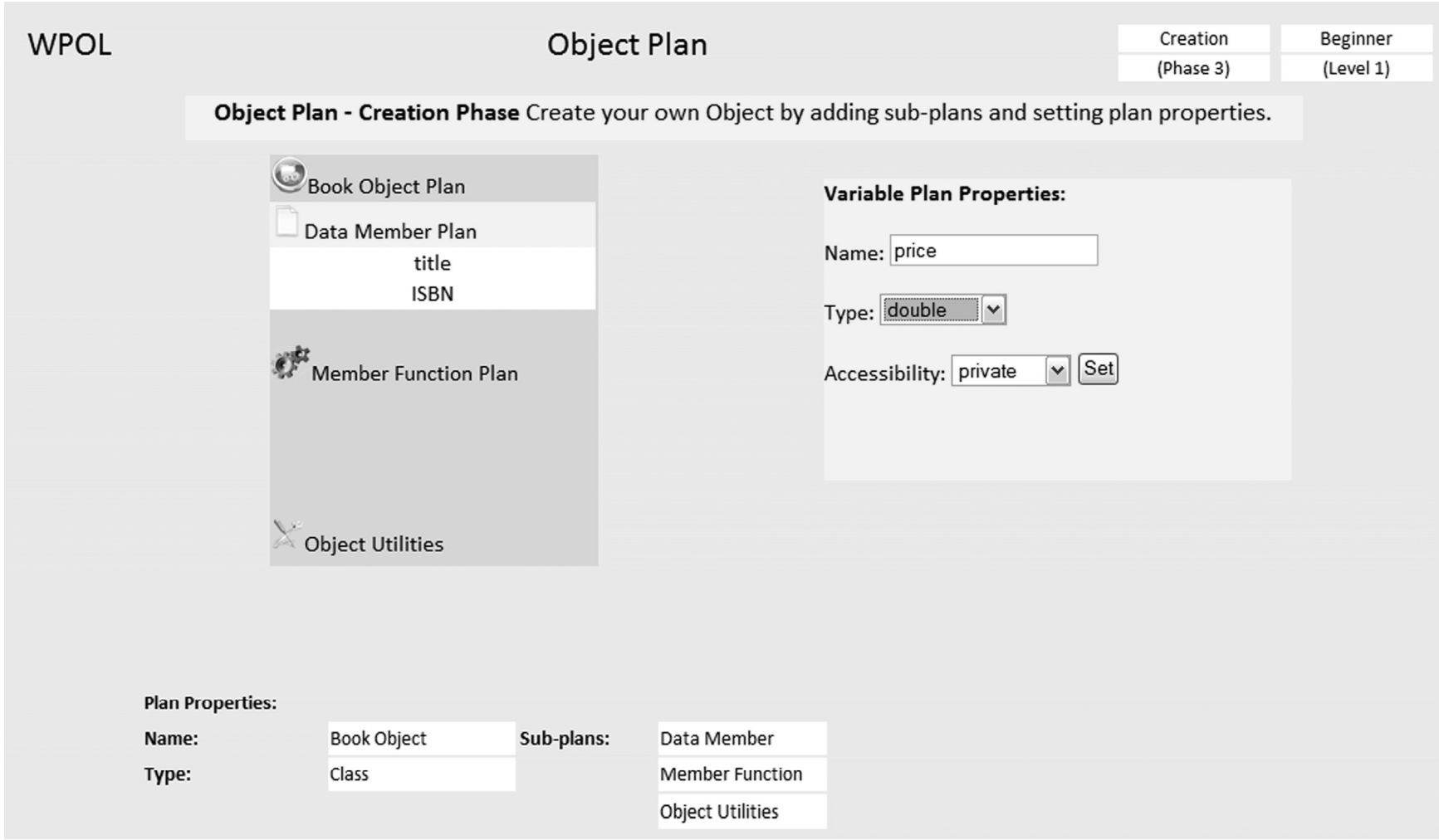

Figure 4.47 – Variable Plan Properties

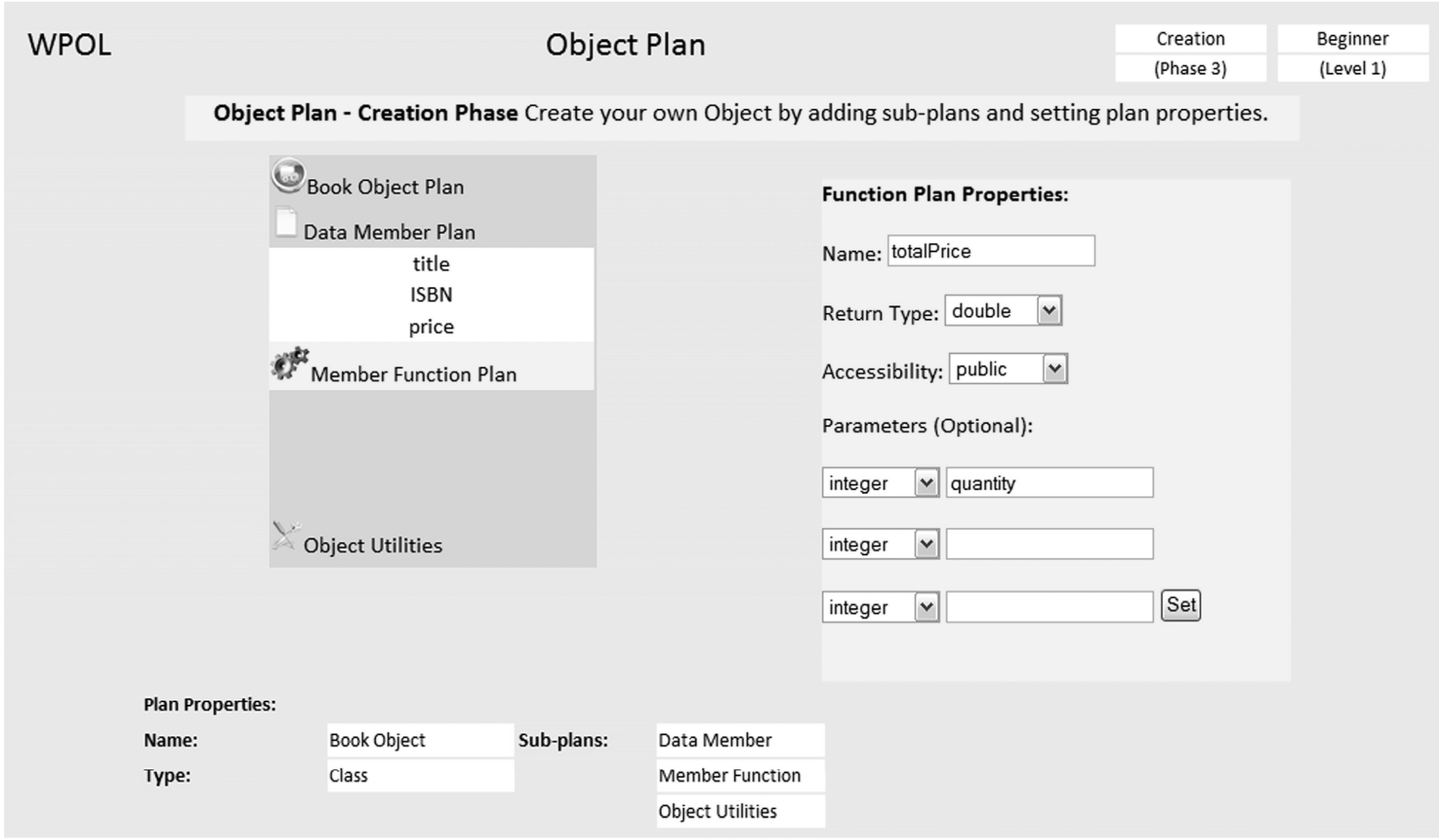

Figure 4.48 – Function Plan Properties

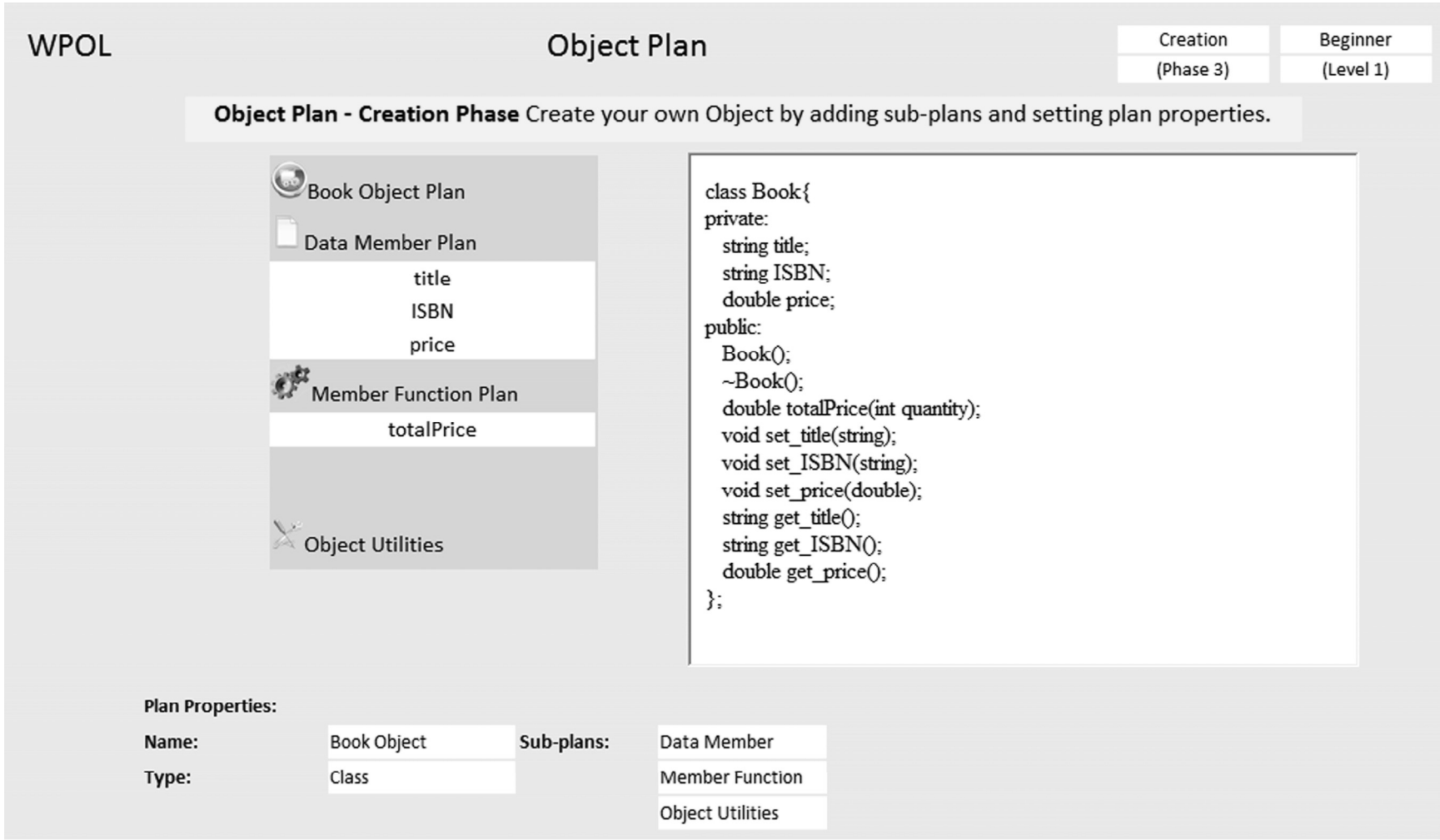

Figure 4.49 – Object Plan Code

5 Data Analysis and Results

5.1 New Error Categories to Include OOP Misconceptions and Errors

There have been several classifications developed for novice programming errors [Ko *et al*. 03]. In order to focus on plan and object related errors, a revised classification has been developed for this empirical study. The categories are broadened to include errors in object oriented design, such as Misrepresentation of objects, data members, and member functions, as well as Non-referenced objects. The errors are classified according to error type (Missing, Misplaced, Malformed, Miscellaneous, Misused, Misrepresentation, Non-referenced). Each error type is categorized whether it is referring to a Plan (Algorithm/Problem Solving), Object (Class), Member Data, Member Function, or Object Utilities (Constructor, Destructor, Set, Get). Among the most common novice errors related to object oriented class design are inability to incorporate objects into the solution and problems with attributes and class cohesion [Thomasson *et al*. 06].

This study is focused on plan and object design and implementation errors, as well as correct incorporation of plans and objects into problem solving. Since the students had ample time to debug their programs in order to produce some output, this study is not focusing on syntax errors.

5.2 Empirical Study of Novice Programmers Comparing Traditional Teaching Methods vs. WPOL

An empirical study was conducted on online programming courses at the State University of New York Empire State College. The course is titled “Introduction to C++ and OOP” and students taking this course are beginner programmers. The course is broken down into seven modules, two weeks each. The topic sequence is as follows: Introduction and Input, Process, Output, Repetition and Decision Making, Arrays and Functions, Functions and Pointers, Class and Objects, Inheritance and Sorting, Polymorphism and Algorithm Analysis. For this study, four sections of this course were examined. The first two sections followed the traditional online course materials (Spring 2007 and Summer 2007) and the next two sections followed the course materials along with exposure to the Plan Object Paradigm and WPOL (Summer 2007 and Fall 2007). The students in these courses were given the same Case Study to complete. The Case Study programs of these groups are analyzed and compared according to the above error classification. The following is the Case Study, which was administered at the end of the module in which objects were introduced:

Case Study:

Create an Employee class for a basic payroll program to compute the net pay salary of hourly based employees. Your program should also find the average net pay for a small company. To define the class, include the appropriate data members, member functions, and access modifiers. For simplicity, use a constant tax rate of 30% to

compute the tax amount. Employees that work over 40 hours will receive overtime pay of one and a half of their hourly rate for overtime hours worked. The output should display the name of each employee, hours worked, hourly rate, overtime pay, regular (gross) pay, tax amount, and net pay. The average net pay of all employees should also be displayed.

Tables 5.1-5.4 categorize the errors of a total of 44 students over 4 semesters.

Errors Observed in Traditional Online Course (22 students over 2 semesters)

	Plans Related to Alg/Prob Solv
Missing	17
Misplaced	21
Malformed	12
Misc	
Total	50
(Related to OOP)	37

Table 5.1 – Traditional Online Group Plan Errors Related to Alg/Problem Solving

	Object (Class)	Data Member	Member Function	Object Utilities
Misrepresentation	11	18	9	
Non-referenced	5	3		
Misused	4	13	11	
Missing			13	23
Misplaced		5	9	
Malformed	6	8	15	2
Misc		5		
Total	26	52	57	25

Table 5.2 – Traditional Online Group Object Errors

Errors Observed in Online Course along with WPOL (Web Plan Object Language) and Plan Object Paradigm (22 students over 2 semesters)

	Plans Related to Alg/Prob Solv
Missing	5
Misplaced	7
Malformed	16
Misc	
Total	28
(Related to OOP)	16

Table 5.3 – WPOL Group Plan Errors Related to Alg/Problem Solving

	Object (Class)	Data Member	Member Function	Object Utilities
Misrepresentation	3	4	4	
Non-referenced	2			
Misused	2		1	1
Missing		1		10
Misplaced		4	12	
Malformed	5		7	
Misc	1	1		
Total	13	10	24	11

Table 5.4 – WPOL Group Object Errors

Tables 5.1 and 5.3 include the errors of the traditional group and WPOL group in regard to Plans related to algorithms and problem solving which are categorized according to error type (Missing, Misplaced, Malformed, Misc.). The students' programs in the WPOL group had 54% fewer total errors related to algorithm and problem solving plans. In particular, the number of missing and misplaced plans was significantly improved. Since many plans were included that were missing in the first group, the number of malformed errors is spread over a larger number of plans. Since one of the goals of this study is to address novices' ability to utilize objects in problem solving, particularly of interest is the 56.7% reduction in problem solving errors related to objects.

Tables 5.2 and 5.4 include the errors in object design, implementation, and use. Object (class), data member, member function, and object utilities errors are categorized by error type (Misrepresentation, Non-referenced, Misused, Missing, Misplaced, Malformed, Misc.). Another goal of this study is to improve novice representation, design, and use of objects. Programs of students in the WPOL group exhibited 50% fewer total errors related to objects. Notable improvements in object, data member, and member function representation and use were observed.

The following are some code snippets from various student solutions to the Case Study that demonstrate errors of interest. It is useful to examine a range of programs to illustrate how students' interpretations of how an object should be created and used in a program.

The Case Study specifications ask the students to compute the net pay of each hourly based employee, requiring input of hours worked and hourly rate and computation of gross pay, overtime pay, and tax amount. The program also should compute the average net pay of all employees. One of the most common errors among students writing this program is misplacement of the Compute Average Net Pay Plan. Many students misplaced this plan inside the Employee Object. The Compute Average Net Pay Plan includes variables for sum and average net pay. Misplacing these variables as data members of the Employee Object demonstrates a misconception about how objects are used in problem solving. Students also misplaced a computeAverageNetPay() member function in the Student Object.

The following student uses parallel arrays inside the employee class. This does not take advantage of objects and constitutes a misrepresentation of the employee object. The student does not recognize that the employee class represents the data and functionality for one employee object. These data members would be considered malformed since they are incorrectly implemented as arrays. This student also includes a misplaced int counter; data member and misplaced float getAverageNetpay(); member function.

Misrepresentation of Employee Object Example

```
class employee{

        ifstream fin;
        string firstname[100], lastname[100];          //MALFORMED
        int  empid[100];
        int counter;                                   //MISPLACED

        float  hoursworked[100], hourlyrate[100], grosspay[100], taxamount[100],
        netpay[100], regularhours[100], overtimehours[100], regularpay[100],
        overtimepay[100];                              //MALFORMED

        void calculateOvertimePays();
        void calculateGrossPays();
        void calculateNetPays();
        void printColumnNames();
        void printArrays();

public: employee();

         ~employee();
         void printreport();
         float getAverageNetpay();                     //MISPLACED

};
```

Figure 5.1 – Program Excerpt Demonstrating a Misrepresentation of the Employee Object, Malformed Data Members, and Misplaced Data Member and Member Function

In the following example, the student creates a separate class for each employee, indicating a clear misconception about objects. Rather than understanding that by designing an employee class, we are actually defining a type that will be used and reused to create instances of this class (employee objects), the student is assumes that each employee necessitates its own class.

Object Misconception

```
class Employee
{
        private:

                string name;
                double hourlyrate;
                double hoursworked;
                double overtimeh;
                double otpay;
                double gp;

        public:

                Employee();
                Employee(string empName, double hoursW, double rate);
                void getinfo();
                double grosspay();
                double overtime();
                double netpay();
                double taxamount();
                void display();

};
class Employee1

{

        private:

                string name;
                double hourlyrate;
                double hoursworked;
                double overtimeh;
                double otpay;
                double gp;
```

```
        public:

                Employee();
                Employee(string empName, double hoursW, double rate);
                void getinfo();
                double grosspay();
                double overtime();
                double netpay();
                double taxamount();
                void display();

};


class Employee2

{

        private:

                string name;
                double hourlyrate;
                double hoursworked;
                double overtimeh;
                double otpay;
                double gp;

        public:

                Employee();
                Employee(string empName, double hoursW, double rate);
                void getinfo();
                double grosspay();
                double overtime();
                double netpay();
                double taxamount();
                void display();

};

class Employee3

{

        private:

                string name;
                double hourlyrate;
                double hoursworked;
                double overtimeh;
                double otpay;
                double gp;
```

```
    public:

        Employee();
        Employee(string empName, double hoursW, double rate);
        void getinfo();
        double grosspay();
        double overtime();
        double netpay();
        double taxamount();
        void display();

};
```

Figure 5.2- Program Demonstrates a Misconception by Erroneously Implementing a Class for Each Employee Object

The next two program excerpts are examples of inability to incorporate the Employee Object into the rest of the program – difficulty utilizing an object to solve a problem. The following student program includes functions for finding the sum and average; however, the net pay information needed from the Employee Object is not passed to the functions nor otherwise accessed. This is also an example of a non-referenced object error. The Plans for computing the sum and average are also malformed since they contain other errors such as incorrect function parameters and incorrect counter.

Non-referenced Employee Object

```
float findsum( int firstnumber, int secondnumber, int thirdnumber ){
    float sum;
    sum = netpay1 + netpay2 + netpay3 + netpay4 + netpay5;
    return sum;
}//Find Sum

float findaverage( float sum ){
    float average;
    average = sum / 5.0;
    return average;
}//Find Average
```

Figure 5.3 – Netpay of Employee Object is not Referenced, Sum and Average Net Pay Plans are Malformed

The following lines of code are also an example of inability to incorporate the Employee Object into solving a problem, as well as a non-referenced object error. Rather than accessing the net pay from the Employee Object, where it is contained, the student incorrectly attempts to compute the average net pay of all employees using an array of manually entered values.

Non-referenced Employee Object

```
int avenet[5]={(1405+868+1638+735+1023)/5};
int avegross[5]={(2008+1240+2340+1050+1462)/5};
cout<<"AVERAGE NETPAY: "; cout<<avenet[0]<<"  "<<"AVERAGE GROSSPAY: "; cout<<avegross[0]<<endl;
```

Figure 5.4 – Employee Object is Not Incorporated into the Program, Average Netpay Plan is Malformed

The following excerpts are from an exemplary program in which the student correctly designs the Employee Object and uses a separate function to compute the average net pay that appropriately utilizes the Employee Object.

Correct Representation and Use of Employee Object

```
class Employee {

public:
        //Employee class constructor with default values
        Employee(string = "", string = "", int = 0, int = 0, float = 0.0);

        //public function called by "main" program
        void calculateNetPay();

        //public functions that return the values of private data members
        int getEmployeeID();
        string getFirstName();
        string getLastName();
        int getHoursWorked();
        float getHourlyRate();
        float getOvertimePay();
        float getBasePay();
        float getGrossPay();
        float getTaxAmount();
        float getNetPay();

private:
        float TAXRATE;

        int employeeID;
        string firstName;
        string lastName;
        int hoursWorked;
        float hourlyRate;
        int overtimeHours;
        float basePay;
        float grossPay;
        float overtimePay;
        float taxAmount;
        float netPay;

        //private functions that handle payroll calculations
        int findOvertimeHours(int);
        float findOvertimePay(int, float);
        float findBasePay(int, int, float);
        float findGrossPay(float, float);
        float findTaxAmount(float);
        float findNetPay(float, float);
};

…
```

```
float findNetpayAvg(Employee employee[], int total) {

        float netpayAvg = 0.0; //stores the final averaging result
        float sum = 0.0; //stores the tally of all netpays

        //get the net pay from each Employee object and add it to the sum variable
        for(int i = 0; i < total; i++)
                sum += employee[i].getNetPay();

        //divide the sum by the total number of employees;
        //assign result to netpayAvg
        netpayAvg = sum / total;

        //return the result
        return netpayAvg;

}
```

Figure 5.5 – Employee Object is Correctly Implemented and Incorporated into the Program

The programs of the group of students exposed to the Plan Object Paradigm and WPOL demonstrated fewer errors related to plans and objects. Due to the reinforcement of Plan and Object design, integration, and implementation, the students' programs reflected better understanding of how plans and objects are created and incorporated into a solution. A visual representation of data member and member function integration helps students identify appropriate data and functions to include in an object's design.

5.3 Teaching Programming in an Online Course vs. Traditional Classroom Setting

Based on observations from teaching the same programming courses online and in a traditional classroom setting, it is interesting to notice that students in an online programming course tend to write more unique programs that exhibit interesting errors. Their work is more reflective of their own ideas and intentions as

to how they think a problem should be solved. Since students in online courses have limited interaction with the instructor and classmates, their work is less influenced by others and as a result more individualized. In the empirical study on online courses in this work, there were several unique solutions, correct and incorrect, that give insight into the understanding and thought process of the student.

6 Conclusion and Future Work

6.1 Dissertation Contributions

Current concerns in computer science education and industry inspire us to find ways to improve a novice's experience of learning programming to foster better assimilation of programming concepts, such as object oriented programming. In essence, this project seeks to capture the way expert programmers represent programming knowledge and visualize this knowledge for novices to enhance their learning of programming in the object oriented paradigm. This work also seeks to address concerns raised by educators in the Objects-First vs. Objects-Next debate by presenting a method for introducing objects with a visual environment and plan representation that reinforces object design and object oriented concepts.

A contribution of this research project is development of a new learning paradigm for object oriented languages. The Plan-Object Paradigm applies the concept of plans to object oriented programming. The Plan-Object approach enhances novice programmers' ability to design, implement, and integrate objects into their programs. Another contribution of this work is a proposed design of an online learning environment for novice programmers. Web Plan Object Language (WPOL) is an environment that utilizes the Plan-Object approach with three phases

of learning: plan observation, plan integration, and plan creation. In order to better characterize novice difficulties with problem solving and object design, new error categories were developed. An empirical study was conducted to measure novice's performance on a sample case study involving objects and problem solving. The online course exposed to the Plan Object Paradigm and WPOL demonstrated fewer errors related to plans and objects. Due to the visual experience of Plan and Object design, integration, and implementation, the students' programs exhibited fewer errors related to plans and OOP. Student programs also reflected better understanding of correct object representation and incorporation of plans and objects into a solution.

6.2 Future Work

There are many avenues for further research and development of this project. The goal is to implement a comprehensive Web Plan Object Language online learning environment and disseminate it for educational use and collaboration. As for specific enhancements, it would be useful to customize WPOL by creating a profile for each novice programmer in order to track their error patterns and learning progress. This environment will also be easily transferred to any object oriented language such as Java and C#. Another future work is to build a Plan and Object Database. This would be a collaborative effort to create a programming resource by collecting a variety of Plans and Objects that can be accessed by others to use as a learning tool or for reuse in building a program.

Appendix

Thesis Related Publication/Presentations

EBRAHIMI, A., & SCHWEIKERT, C. "Empirical Study of Novice Programming with Plans and Objects", ACM Inroads 38, 4 (Dec. 2006), 52-54.

SCHWEIKERT, C., & EBRAHIMI, A. “A Plan Oriented Object Learning Approach” Fourteenth International Conference on Learning, Johannesburg, South Africa, Jun. 2007

SCHWEIKERT, C., & EBRAHIMI, A. “Teaching Approaches for Object Oriented Programming Online: An Empirical Study of an SLN Course” Conference on Instructional Technologies, SUNY Plattsburgh, May 29-June 1, 2007

SCHWEIKERT, C., & EBRAHIMI, A. “Web Learning via Plan and Assessment” Thirteenth International Conference on Learning, Montego Bay, Jamaica, Jun. 2006

YARMISH, G., EBRAHIMI, A., KOPEC, D., SCHWEIKERT, C., SOKOL, D. “Programmer Errors: Formal and Informal Methods of Reducing Error Rate” Second International Conference on Technology, Knowledge, and Society – Dec. 2005.

SCHWEIKERT, C. “Study of Common Student Errors and Learning Patterns”, SUNY New Paltz TLC Conference “Enhancing Teaching and Learning Across the Disciplines”, October 22, 2004

Source Code Extracts

Observation Phase Sample 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>WPOL</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<script language="JavaScript" type="text/JavaScript">

function fbuild()
{
	var nbuild = document.getElementById("build").value;

	if(nbuild==0)
	{
		document.getElementById("imode").style.visibility="visible";
		document.getElementById("pdata").style.visibility="visible";
		document.getElementById("datadiv").style.visibility="hidden";

		nbuild++; document.getElementById("build").value=nbuild;
	}
	else if(nbuild==1)
	{
		document.getElementById("funcdiv").style.visibility="hidden";
		document.getElementById("pfunc").style.visibility="visible";

		nbuild++; document.getElementById("build").value=nbuild;
	}
	else if(nbuild==2)
	{
		document.getElementById("pstud").style.backgroundColor="#898989";
		document.getElementById("pdata").style.backgroundColor="#FFFF99";
		document.getElementById("pname").innerHTML="Data Member";
		document.getElementById("ptype").innerHTML="Class Component";
		document.getElementById("sub1").innerHTML="Variable(s)";
		document.getElementById("sub2").innerHTML="";
		document.getElementById("sub2").style.backgroundColor="EBEBEB";
		document.getElementById("imode").style.visibility="hidden";

		nbuild++; document.getElementById("build").value=nbuild;
	}
	else if(nbuild==3)
	{
		document.getElementById("iid").src="pics/yid.png";
		document.getElementById("pid").style.visibility="visible";
		document.getElementById("imode").style.visibility="visible";
		document.getElementById("pname").innerHTML="id";
		document.getElementById("ptype").innerHTML="Variable";
		document.getElementById("prop3").innerHTML="<strong>Data Type:</strong>";
		document.getElementById("sub1").innerHTML="integer";
```

```
	document.getElementById("prop4").innerHTML="<strong>Accessibility:</strong>";
		document.getElementById("sub2").innerHTML="private";
		document.getElementById("sub2").style.backgroundColor="#FFFFFF";

		nbuild++; document.getElementById("build").value=nbuild;
	}
	else if(nbuild==4)
	{
		document.getElementById("iid").src="pics/id.png";
		document.getElementById("iass").src="pics/yass.png";
		document.getElementById("pass").style.visibility="visible";
		document.getElementById("pname").innerHTML="assignmentScore";
		document.getElementById("sub1").innerHTML="double";

		nbuild++; document.getElementById("build").value=nbuild;
	}
	else if(nbuild==5)
	{
		document.getElementById("iass").src="pics/ass.png";
		document.getElementById("imid").src="pics/ymid.png";
		document.getElementById("pmid").style.visibility="visible";
		document.getElementById("pname").innerHTML="midtermScore";

		nbuild++; document.getElementById("build").value=nbuild;
	}
	else if(nbuild==6)
	{
		document.getElementById("imid").src="pics/mid.png";
		document.getElementById("ifin").src="pics/yfin.png";
		document.getElementById("pfin").style.visibility="visible";
		document.getElementById("pname").innerHTML="finalScore";

		nbuild++; document.getElementById("build").value=nbuild;
	}
	else if(nbuild==7)
	{
			document.getElementById("ifin").src="pics/fin.png";
			document.getElementById("pdata").style.backgroundColor="#D6D6D6";
			document.getElementById("pfunc").style.backgroundColor="#FFFF99";
			document.getElementById("pname").innerHTML="Member Function";
			document.getElementById("ptype").innerHTML="Class Component";
			document.getElementById("prop3").innerHTML="<strong>Sub-
plans:</strong>";
			document.getElementById("sub1").innerHTML="Function(s)";
			document.getElementById("imode").style.visibility="hidden";
			document.getElementById("prop4").innerHTML="";
			document.getElementById("sub2").innerHTML="";
			document.getElementById("sub2").style.backgroundColor="#EBEBEB";

			nbuild++; document.getElementById("build").value=nbuild;
	}
	else if(nbuild==8)
	{
			document.getElementById("imode").style.visibility="visible";
			document.getElementById("iavg").src="pics/yavg.png";
```

```
                                document.getElementById("pavg").style.visibility="visible";
                                document.getElementById("pname").innerHTML="computeAverage";
                                document.getElementById("ptype").innerHTML="Function";
                                document.getElementById("prop3").innerHTML="<strong>Return
Type:</strong>";
                                document.getElementById("sub1").innerHTML="double";

        document.getElementById("prop4").innerHTML="<strong>Parameter(s):</strong>";
                                document.getElementById("sub2").style.backgroundColor="#FFFFFF";

        document.getElementById("prop5").innerHTML="<strong>Accessibility:</strong>";
                                document.getElementById("prop5i").innerHTML="public";
                                document.getElementById("prop5i").style.backgroundColor="#FFFFFF";

                                document.getElementById("bbuild").value="Code View";

                                nbuild++; document.getElementById("build").value=nbuild;

        }
        else if(nbuild==9)
        {
                document.location="p2.html";
        }

}//end fbuild function

<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if ((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.reload();
}
MM_reloadPage(true);
//-->
</head>

<body onload="document.getElementById('build').value=0;">
<div class="style2" id="Layer1" style="position:absolute; width:1009px; height:577px; z-index:1">
  <div class="style2" id="phase" style="position:absolute; width:115px; height:76px; z-index:1; left:
781px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000;
top: 0px;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Observation</div></td>
      </tr>
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Phase 1) </div></td>
      </tr>
    </table>
  </div>
  <div class="style2" id="level" style="position:absolute; width:115px; height:76px; z-index:2; left:
895px; top: 0px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Beginner</div></td>
```

```
    </tr>
    <tr>
      <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Level 1) </div></td>
    </tr>
   </table>
  </div>
  <div id="Layer6" style="position:absolute; width:98px; height:407px; z-index:3; top: 76px;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000; visibility:
visible;">
  </div>
  <div id="properties" style="position:absolute; width:912px; height:105px; z-index:6; left: 98px; top:
483px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000;">
    <table style="table-layout:fixed" width="730" height="101" border="0">
     <tr>
      <td width="126"><strong>Plan Properties:</strong></td>
     </tr>
     <tr>
      <td><strong>Name:</strong></td>
      <td bgcolor="#FFFFFF" id="pname">Student Object</td>
      <td width="116" id="prop3"><strong>Sub-plans:</strong></td>
      <td bgcolor="#FFFFFF" id="sub1">Data Member</td>
      <td id="prop5"> </td>
      <td id="prop5i"> </td>
     </tr>
     <tr>
      <td><strong>Type:</strong></td>
      <td bgcolor="#FFFFFF" id="ptype">Class</td>
      <td id="prop4"> </td>
      <td bgcolor="#FFFFFF" id="sub2">Member Function</td>
      <td> </td>
      <td> </td>
     </tr>
    </table>
</div>
</div>
<div class="style2" id="Layer2" style="position:absolute; width:98px; height:76px; z-index:2;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 0px none #000000;">
  <div align="center"><br>
    <span class="style5">WPOL</span></div>
</div>
<div class="style2" id="Layer3" style="position:absolute; width:683px; height:76px; z-index:3; left:
108px; top: 15px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000;">
<div align="center"><br>
     <span class="style5">Student Average Plan</span></div>
</div>
<div id="apDiv1"></div>
<div id="apDiv2">
  <input name="build" type="hidden" id="build" value="0">
</div>
<div id="apDiv4"><img src="pics/student1.png" width="115" height="115"></div>
<div id="apDiv5"><img src="pics/id.png" name="iid" width="86" height="86" id="iid"></div>
<div id="apDiv6"><img src="pics/ass.png" name="iass" width="84" height="74" id="ass"></div>
<div id="apDiv7"><img src="pics/mid.png" name="imid" width="84" height="74" id="mid"></div>
<div id="apDiv8"><img src="pics/fin.png" name="ifin" width="84" height="74" id="fin"></div>
<div id="apDiv9"><img src="pics/avg.png" name="iavg" width="86" height="88" id="avg"></div>
```

```
<div id="apDiv10">
  <table style="table-layout:fixed" width="208" height="222" border="0" bgcolor="#D6D6D6">
    <tr>
      <td valign="middle" bgcolor="#FFFF99" id="pstud"><span class="style6"><img
src="pics/sstud.png" width="26" height="26">Student Object Plan</span></td>
    </tr>
    <tr>
      <td id="pdata" style="visibility:hidden;" valign="middle">    <img src="pics/data.png" width="24"
height="24">Data Member Plan</td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pid" style="visibility:hidden;"><div
align="center">id</div></td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pass" style="visibility:hidden;"><div
align="center">assignmentScore</div></td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pmid" style="visibility:hidden;"><div
align="center">midtermScore</div></td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pfin" style="visibility:hidden;"><div
align="center">finalScore</div></td>
    </tr>
    <tr>
      <td id="pfunc" style="visibility:hidden;" valign="middle"><img src="pics/func.png" width="29"
height="26">Member Function Plan</td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pavg" style="visibility:hidden;"><div
align="center">computeAverage</div></td>
    </tr>
  </table>
</div>
<div id="apDiv12">
  <input type="submit" name="bbuild" id="bbuild" value="Integrate" onClick="fbuild()">
</div>
<div id="datadiv"><img src="pics/data.png" alt="" width="24" height="24">Data Member
Plan</div>
<div id="imode"><img src="pics/emb.png" alt="" width="60" height="60">Integration Mode:
Embedded</div>
<div id="funcdiv"><img src="pics/func.png" width="29" height="26">Member Function Plan</div>
</body>
</html>
```

Observation Phase Sample 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>WPOL</title>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript" type="text/JavaScript">

function id()
{
                document.getElementById("sarr").style.visibility="hidden";
                document.getElementById("pstud").style.backgroundColor="#898989";
                document.getElementById("pdata").style.backgroundColor="#FFFF99";
                document.getElementById("pid").style.backgroundColor="#FFFF99";
                document.getElementById("iarr").style.visibility="visible";

                document.getElementById("classd").style.backgroundColor="#FFFFFF";
                document.getElementById("classb").style.backgroundColor="#FFFFFF";
                document.getElementById("classe").style.backgroundColor="#FFFFFF";
                document.getElementById("priv").innerHTML="   private:";

        document.getElementById("idd").innerHTML="      
  int id;";
                document.getElementById("idd").style.backgroundColor="#FFFF99";

                document.getElementById("pname").innerHTML="id";
                document.getElementById("ptype").innerHTML="Variable";
                document.getElementById("prop3").innerHTML="<strong>Data Type:</strong>";
                document.getElementById("sub1").innerHTML="integer";
                document.getElementById("sub1").style.backgroundColor="#FFFF99";

        document.getElementById("prop4").innerHTML="<strong>Accessibility:</strong>";
                document.getElementById("sub2").innerHTML="private";
                document.getElementById("sub2").style.backgroundColor="#FFFF99";
}

function assign()
{
                document.getElementById("pid").style.backgroundColor="#FFFFFF";
                document.getElementById("iarr").style.visibility="hidden";
                document.getElementById("pass").style.backgroundColor="#FFFF99";
                document.getElementById("aarr").style.visibility="visible";


        document.getElementById("assd").innerHTML="      
  double assignmentScore;";
                document.getElementById("idd").style.backgroundColor="#FFFFFF";
                document.getElementById("assd").style.backgroundColor="#FFFF99";

                document.getElementById("pname").innerHTML="assignmentScore";
                document.getElementById("sub1").innerHTML="double";
}

function midterm()
{
                document.getElementById("pass").style.backgroundColor="#FFFFFF";
                document.getElementById("aarr").style.visibility="hidden";
                document.getElementById("pmid").style.backgroundColor="#FFFF99";
                document.getElementById("marr").style.visibility="visible";
```

```
	document.getElementById("midd").innerHTML="      
;  double midtermScore;";
		document.getElementById("assd").style.backgroundColor="#FFFFFF";
		document.getElementById("midd").style.backgroundColor="#FFFF99";

		document.getElementById("pname").innerHTML="midtermScore";
}
function final()
{
		document.getElementById("pmid").style.backgroundColor="#FFFFFF";
		document.getElementById("marr").style.visibility="hidden";
		document.getElementById("pfin").style.backgroundColor="#FFFF99";
		document.getElementById("farr").style.visibility="visible";


	document.getElementById("find").innerHTML="      
  double finalScore;";
		document.getElementById("midd").style.backgroundColor="#FFFFFF";
		document.getElementById("find").style.backgroundColor="#FFFF99";
		document.getElementById("pname").innerHTML="finalScore";
}
function compavg()
{
		document.getElementById("pfin").style.backgroundColor="#FFFFFF";
		document.getElementById("farr").style.visibility="hidden";
		document.getElementById("pdata").style.backgroundColor="#EBEBEB";
		document.getElementById("pfunc").style.backgroundColor="#FFFF99";
		document.getElementById("pavg").style.backgroundColor="#FFFF99";
		document.getElementById("carr").style.visibility="visible";

		document.getElementById("pub").innerHTML="   public:";

	document.getElementById("avgd").innerHTML="      
;  double computeAverage(); ";
		document.getElementById("find").style.backgroundColor="#FFFFFF";
		document.getElementById("pub").style.backgroundColor="#FFFF99";
		document.getElementById("avgd").style.backgroundColor="#FFFF99";

		document.getElementById("pname").innerHTML="computeAverage";
		document.getElementById("ptype").innerHTML="Function";
		document.getElementById("prop3").innerHTML="<strong>Return Type:</strong>";
		document.getElementById("sub1").innerHTML="double";

	document.getElementById("prop4").innerHTML="<strong>Parameter(s):</strong>";
		document.getElementById("sub2").style.backgroundColor="#FFFFFF";
		document.getElementById("sub2").innerHTML="";

	document.getElementById("prop5").innerHTML="<strong>Accessibility:</strong>";
		document.getElementById("prop5i").innerHTML="public";
		document.getElementById("prop5i").style.backgroundColor="#FFFF99";

		document.getElementById("bbuild").value="Integration View";
		document.getElementById("bbuild").style.visibility="visible";
}
function fbuild()
```

```
{
        var nbuild = document.getElementById("build").value;

        setTimeout("id()",2000);
        setTimeout("assign()",4000);
        setTimeout("midterm()",6000);
        setTimeout("final()",8000);
        setTimeout("compavg()",10000);

        /*else if(nbuild==5)
        {
                document.location="p3.html";


        }
        */
}//end fbuild function

<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if ((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.reload();
}
MM_reloadPage(true);
//-->
</script>
</head>

<body onload="fbuild();document.getElementById('build').value=0;">
<div class="style2" id="Layer1" style="position:absolute; width:1009px; height:577px; z-index:1">
  <div class="style2" id="phase" style="position:absolute; width:115px; height:76px; z-index:1; left:
781px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000;
top: 0px;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Observation</div></td>
      </tr>
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Phase 1) </div></td>
      </tr>
    </table>
  </div>
  <div class="style2" id="level" style="position:absolute; width:115px; height:76px; z-index:2; left:
895px; top: 0px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Beginner</div></td>
      </tr>
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Level 1) </div></td>
      </tr>
    </table>
  </div>
```

```
  <div id="Layer6" style="position:absolute; width:98px; height:407px; z-index:3; top: 76px;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000; visibility:
visible;">
  </div>
  <div id="properties" style="position:absolute; width:912px; height:105px; z-index:6; left: 98px; top:
483px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000;">
    <table style="table-layout:fixed" width="730" height="101" border="0">
      <tr>
        <td width="126"><strong>Plan Properties:</strong></td>
      </tr>
      <tr>
        <td><strong>Name:</strong></td>
        <td bgcolor="#FFFF99" id="pname">Student Object</td>
        <td width="116" id="prop3"><strong>Sub-plans:</strong></td>
        <td bgcolor="#FFFFFF" id="sub1">Data Member</td>
        <td id="prop5"> </td>
        <td id="prop5i"> </td>
      </tr>
      <tr>
        <td><strong>Type:</strong></td>
        <td bgcolor="#FFFF99" id="ptype">Class</td>
        <td id="prop4"> </td>
        <td bgcolor="#FFFFFF" id="sub2">Member Function</td>
      </tr>
      <tr>
      </tr>
    </table>
</div>
</div>
<div class="style2" id="Layer2" style="position:absolute; width:98px; height:76px; z-index:2;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 0px none #000000;">
  <div align="center"><br>
    <span class="style5">WPOL</span></div>
</div>
<div class="style2" id="Layer3" style="position:absolute; width:683px; height:76px; z-index:3; left:
108px; top: 15px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000;">
<div align="center"><br>
      <span class="style5">Student Average Plan</span></div>
</div>
<div id="apDiv1"></div>
<div id="apDiv2">
  <input name="build" type="hidden" id="build" value="0">
</div>
<div id="apDiv10">
  <table style="table-layout:fixed" width="208" height="222" border="0" bgcolor="#D6D6D6">
    <tr>
      <td valign="middle" bgcolor="#FFFF99" id="pstud"><span class="style6"><img
src="pics/sstud.png" width="26" height="26">Student Object Plan</span></td>
    </tr>
    <tr>
      <td id="pdata" valign="middle">    <img src="pics/data.png" width="24" height="24">Data
Member Plan</td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pid"><div align="center">id</div></td>
```

```
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pass"><div
align="center">assignmentScore</div></td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pmid"><div
align="center">midtermScore</div></td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pfin"><div align="center">finalScore</div></td>
    </tr>
    <tr>
      <td id="pfunc" valign="middle"><img src="pics/func.png" width="29" height="26">Member
Function Plan</td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pavg"><div
align="center">computeAverage</div></td>
    </tr>
  </table>
</div>
<div id="apDiv12">
  <input type="submit" name="bbuild" id="bbuild" value="Implement"
onClick="document.location='p3.html'">
</div>
<div id="apDiv13">
  <table width="267" height="318" border="0" cellpadding="0" cellspacing="0">
    <tr>
      <td bgcolor="#FFFF99" id="classd">class Student</td>
    </tr>
    <tr>
      <td bgcolor="#FFFF99" id="classb">{</td>
    </tr>
    <tr>
      <td bgcolor="#FFFFFF" id="priv">   </td>
    </tr>
    <tr>
      <td bgcolor="#FFFFFF" id="idd">   </td>
    </tr>
    <tr>
      <td bgcolor="#FFFFFF" id="assd">  </td>
    </tr>
    <tr>
      <td bgcolor="#FFFFFF" id="midd">   </td>
    </tr>
    <tr>
      <td bgcolor="#FFFFFF" id="find">   </td>
    </tr>
    <tr>
      <td bgcolor="#FFFFFF"> </td>
    </tr>
    <tr>
      <td bgcolor="#FFFFFF" id="pub">   </td>
    </tr>
    <tr>
```

```
    <td bgcolor="#FFFFFF" id="avgd">   </td>
   </tr>
   <tr>
    <td bgcolor="#FFFF99" id="classe"> };//Student class </td>
   </tr>
  </table>
</div>
<div id="sarr"><img src="pics/arr1.png" width="210" height="54"></div>
<div id="iarr"><img src="pics/arr2.png" width="210" height="54"></div>
<div id="marr"><img src="pics/arr2.png" width="210" height="54"></div>
<div id="carr"><img src="pics/arr3.png" width="210" height="54"></div>
<div id="farr"><img src="pics/arr3.png" width="210" height="54"></div>
<div id="aarr"><img src="pics/arr2.png" width="210" height="54"></div>
</body>
</html>
```

Observation Phase Sample 3

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>WPOL</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<script language="JavaScript" type="text/JavaScript">

function isNumber(sText)
{
  var ValidChars = "0123456789.";
  var IsNumber=true;
  var Char;

  for (i = 0; i < sText.length && IsNumber == true; i++)
    {
    Char = sText.charAt(i);
    if (ValidChars.indexOf(Char) == -1)
      {
      IsNumber = false;
      }
    }
  return IsNumber;

  }

function idEnter(e){

   e= e || window.event;
   if (e.keyCode == 13){
   if(!isNumber(studentForm.ide.value))
   {
          alert("Please enter a number");
          studentForm.ide.value="";
 }
 else
```

```
  {
  studentForm.ide.readOnly=true;
  document.getElementById("pid").innerHTML="<div
align='center'>id:"+studentForm.ide.value+"</div>";
document.getElementById("pid").style.backgroundColor="#FFFF99";
document.getElementById("setdiv").style.backgroundColor="#FFFF99";
 document.getElementById("assm").style.visibility="visible";

studentForm.asse.readOnly=false;
studentForm.asse.focus();
                                        }
   }
}

function assignmentEnter(e){

    e= e || window.event;
    if (e.keyCode == 13){
    if(!isNumber(studentForm.asse.value))
   {
   alert("Please enter a number");
   studentForm.asse.value="";
   }
else
{
            document.getElementById("pid").style.backgroundColor="#FFFFFF";
             studentForm.asse.readOnly=true;
            document.getElementById("pass").innerHTML="<div align='center'>assignmentScore:
            "+studentForm.asse.value+"</div>";
            ocument.getElementById("pass").style.backgroundColor="#FFFF99";
             document.getElementById("midm").style.visibility="visible";
             studentForm.mide.readOnly=false;
             studentForm.mide.focus();
}
}
}

function midtermEnter(e){

    e= e || window.event;
    if (e.keyCode == 13){
    if(!isNumber(studentForm.mide.value))
   {
            alert("Please enter a number");
            studentForm.mide.value="";
   }
   else
   {
   document.getElementById("pass").style.backgroundColor="#FFFFFF";
   studentForm.mide.readOnly=true;

   document.getElementById("pmid").innerHTML="<div align='center'>midtermScore:
"+studentForm.mide.value+"</div>";
  document.getElementById("pmid").style.backgroundColor="#FFFF99";
  document.getElementById("finm").style.visibility="visible";
```

```
 studentForm.fine.readOnly=false;
 studentForm.fine.focus();
}
}
}

function finalEnter(e){

   e= e || window.event;
   if (e.keyCode == 13){
  if(!isNumber(studentForm.fine.value))
  {
  alert("Please enter a number");
  studentForm.fine.value="";
  }
  else
  {
  document.getElementById("pmid").style.backgroundColor="#FFFFFF";
  studentForm.fine.readOnly=true;
  studentForm.fine.blur();

document.getElementById("pfin").innerHTML="<div align='center'>finalScore:
"+studentForm.fine.value+"</div>";
document.getElementById("pfin").style.backgroundColor="#FFFF99";
var id;
var assignmentScore=0;
var midtermScore=0;
var finalScore=0;
id = studentForm.ide.value;
assignmentScore = studentForm.asse.value;
midtermScore = studentForm.mide.value;
finalScore = studentForm.fine.value;

var total=0;
var average;
total = parseInt(assignmentScore) + parseInt(midtermScore) + parseInt(finalScore);
average = total / 3;

studentForm.avgo.value="Average for student "+id+":        "+average.toFixed(2);

document.getElementById("pavg").innerHTML="<div align='center'>computeAverage:
"+average.toFixed(2)+"</div>";
document.getElementById("pfin").style.backgroundColor="#FFFFFF";
document.getElementById("getdiv").style.backgroundColor="#FFFF99";
document.getElementById("setdiv").style.backgroundColor="#FFFFFF";

document.getElementById("bdiv").style.visibility="visible";
}
}
}

function clearinput()
{
            studentForm.ide.value="";
            studentForm.asse.value="";
            studentForm.mide.value="";
```

```
            studentForm.fine.value="";
            studentForm.avgo.value="";

}

<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if ((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.reload();
}
MM_reloadPage(true);
//-->
</script>
</head>

<body onload="document.forms.studentForm.ide.focus(); clearinput();">
<div class="style2" id="Layer1" style="position:absolute; width:1009px; height:577px; z-index:1;
visibility: visible;">
  <div class="style2" id="phase" style="position:absolute; width:115px; height:76px; z-index:1; left:
781px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000;
top: 0px; visibility: visible;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Observation</div></td>
      </tr>
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Phase 1) </div></td>
      </tr>
    </table>
  </div>
  <div class="style2" id="level" style="position:absolute; width:115px; height:76px; z-index:2; left:
895px; top: 0px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000; visibility: visible;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Beginner</div></td>
      </tr>
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Level 1) </div></td>
      </tr>
    </table>
  </div>
  <div id="Layer6" style="position:absolute; width:98px; height:407px; z-index:3; top: 76px;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000; visibility:
visible;">
  </div>
  <div id="properties" style="position:absolute; width:912px; height:105px; z-index:6; left: 98px; top:
483px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000;
visibility: visible;">
    <table style="table-layout:fixed" width="790" height="89" border="0">
      <tr>
        <td width="96"><strong>Plan Properties:</strong></td>
      </tr>
      <tr>
        <td><strong>Name:</strong></td>
```

```
      <td bgcolor="#FFFFFF" id="pname">Student Average</td>
      <td width="90" id="prop3"><strong>Objects:</strong></td>
      <td bgcolor="#FFFFFF" id="sub1">Student</td>
      <td id="prop5"><strong>Sub-plans</strong></td>
      <td bgcolor="#FFFFFF" id="prop5i">Student Average Main</td>
      <td bgcolor="#E8E8E8" id="prop5i"> </td>
      <td bgcolor="#E8E8E8" id="prop5i"> </td>
    </tr>
    <tr>
      <td><strong>Type:</strong></td>
      <td bgcolor="#FFFFFF" id="ptype">Master</td>
      <td id="prop4"> </td>
      <td bgcolor="#E8E8E8" id="sub2"> </td>
      <td> </td>
      <td bgcolor="#E8E8E8"> </td>
      <td> </td>
      <td bgcolor="#E8E8E8"> </td>
    </tr>
    <tr>
    </tr>
  </table>
</div>
</div>
<div class="style2" id="Layer2" style="position:absolute; width:98px; height:76px; z-index:2;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 0px none #000000; visibility:
visible;">
  <div align="center"><br>
    <span class="style5">WPOL</span></div>
</div>
<div class="style2" id="Layer3" style="position:absolute; width:683px; height:76px; z-index:3; left:
108px; top: 15px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000; visibility: visible;">
<div align="center"><br>
     <span class="style5">Student Average Plan</span></div>
</div>
<div id="apDiv1"></div>
<div id="apDiv2">
  <input name="build" type="hidden" id="build" value="0">
</div>
<div id="apDiv10">
  <table style="table-layout:fixed" width="208" height="222" border="0" bgcolor="#D6D6D6">
    <tr>
      <td valign="middle" bgcolor="#898989" id="pstud"><span class="style6"><img
src="pics/sstud.png" width="26" height="26">Student Object Plan</span></td>
    </tr>
    <tr>
      <td id="pdata" valign="middle">    <img src="pics/data.png" width="24" height="24">Data
Member Plan</td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pid"><div align="center">id</div></td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pass"><div
align="center">assignmentScore</div></td>
    </tr>
```

```
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pmid"><div
align="center">midtermScore</div></td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pfin"><div align="center">finalScore</div></td>
    </tr>
    <tr>
      <td id="pfunc" valign="middle"><img src="pics/func.png" width="29" height="26">Member
Function Plan</td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#99FF99" id="pavg"><div
align="center">computeAverage</div></td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#D6D6D6" id="pavg3"><img src="pics/tools.png" width="24"
height="24">Object Utilities</td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pavg2"><div align="center">Set Plan</div></td>
    </tr>
    <tr>
      <td valign="middle" bgcolor="#FFFFFF" id="pavg4"><div align="center">Get Plan</div></td>
    </tr>
  </table>
</div>
<div id="bdiv">
  <input type="button" name="bbuild" id="bbuild" value="View Plan Code"
onClick="document.location='code.html'">
</div>
<div id="maindiv"><img src="pics/main.png" width="28" height="26">Student Average Main
Plan<br>
  <br>
  <br>
</div>
<div id="studobjdiv"><img src="pics/sstud.png" width="26" height="26"> Student Object</div>
<div class="style7" id="inputdiv">
  <div align="center">Input Student Plan<br>
    [Mode:Console]<br>
  </div>
</div>
<div class="style7" id="displaydiv">
  <div align="center">Display Student Plan<br>
    [Mode:Console]</div>
</div>
<div id="apparrdiv">
  <div align="center"><img src="pics/apparr.png" width="35" height="35"></div>
</div>
<div id="setdiv">
  <div align="center" class="style10">Set Plan</div>
</div>
<div id="compdiv">
  <div align="center">computeAverage</div>
</div>
<div id="getdiv">
```

```
  <div align="center" class="style10">Get Plan</div>
</div>
<div id="apDiv12">
<form name="studentForm">
  <table width="332" height="152" border="0" cellpadding="0" cellspacing="0">
    <tr>
      <td width="164" id="idm"><span class="style7">Enter student id:</span></td>
      <td width="168"><input type="text" name="ide" id="ide" onkeypress="idEnter(event);"
tabindex="1"       style=" COLOR: white; BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE:
none; BORDER-LEFT-STYLE: none; BACKGROUND-COLOR: black; BORDER-BOTTOM-
STYLE: none" ></td>
    </tr>
    <tr>
      <td id="assm" style="visibility:hidden"><span class="style7">Enter assignment
score:</span></td>
      <td><input name="asse" type="text" id="asse" onkeypress="assignmentEnter(event);"
readonly="readonly"       style=" COLOR: white; BORDER-TOP-STYLE: none; BORDER-RIGHT-
STYLE: none; BORDER-LEFT-STYLE: none; BACKGROUND-COLOR: black; BORDER-
BOTTOM-STYLE: none" ></td>
    </tr>
    <tr>
      <td id="midm" style="visibility:hidden"><span class="style7">Enter midterm score:</span></td>
      <td><input name="mide" type="text" id="mide" onkeypress="midtermEnter(event);"
readonly="readonly"       style=" COLOR: white; BORDER-TOP-STYLE: none; BORDER-RIGHT-
STYLE: none; BORDER-LEFT-STYLE: none; BACKGROUND-COLOR: black; BORDER-
BOTTOM-STYLE: none" ></td>
    </tr>
    <tr>
      <td id="finm" style="visibility:hidden"><span class="style7">Enter final score:</span></td>
      <td><input name="fine" type="text" id="fine" onkeypress="finalEnter(event);"
readonly="readonly"       style=" COLOR: white; BORDER-TOP-STYLE: none; BORDER-RIGHT-
STYLE: none; BORDER-LEFT-STYLE: none; BACKGROUND-COLOR: black; BORDER-
BOTTOM-STYLE: none" ></td>
    </tr>
    <tr>
      <td colspan="2" id="avgm"><span class="style7">
        <input name="avgo" style="COLOR:#FFFFFF; BORDER-TOP-STYLE: none; BORDER-
RIGHT-STYLE: none; BORDER-LEFT-STYLE: none; BACKGROUND-COLOR: black; BORDER-
BOTTOM-STYLE: none" type="text" id="avgo" size="40">
      </span></td>
      </tr>
  </table>
  </form>
</div>
<div id="apDiv16"><img src="pics/intarr.png" width="55" height="55"></div>
</body>
</html>
```

Observation Phase Sample 4

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
<title>WPOL</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<script language="JavaScript" type="text/JavaScript">

function studarray()
{
	document.getElementById("arraydiv").style.visibility="visible";
	document.getElementById("studarrc").style.visibility="visible";

	document.getElementById("pname").innerHTML="Array";  //Plan properties
	document.getElementById("ptype").innerHTML="Lang Const";
	document.getElementById("prop3").innerHTML="<b>Elements:</b>";
	document.getElementById("sub1").innerHTML="Students";
	document.getElementById("prop4").innerHTML="<b>Size:</b>";
	document.getElementById("sub2").innerHTML="5";
	document.getElementById("prop5").innerHTML="";
	document.getElementById("prop51").innerHTML="";
	document.getElementById("prop52").innerHTML="";
	document.getElementById("prop53").innerHTML="";
	document.getElementById("prop54").innerHTML="";

	document.getElementById("sub1").style.backgroundColor="#FFFF99";
	document.getElementById("sub2").style.backgroundColor="#FFFF99";
	document.getElementById("prop5").style.backgroundColor="#E8E8E8";
	document.getElementById("prop51").style.backgroundColor="#E8E8E8";
	document.getElementById("prop52").style.backgroundColor="#E8E8E8";
	document.getElementById("prop53").style.backgroundColor="#E8E8E8";
	document.getElementById("prop54").style.backgroundColor="#E8E8E8";

}//show inner looop

function inloop()
{
	document.getElementById("studarrc").style.backgroundColor="#FFFFFF";

	document.getElementById("iloopdiv2").style.visibility="visible";
	document.getElementById("iloopc").style.visibility="visible";
	document.getElementById("iloopcbeg").style.visibility="visible";
	document.getElementById("iloopc1").style.visibility="visible";
	document.getElementById("iloopc2").style.visibility="visible";
	document.getElementById("iloopcend").style.visibility="visible";

	document.getElementById("pname").innerHTML="Loop";  //Plan properties
	document.getElementById("prop3").innerHTML="<b>Loop Type:</b>";
	document.getElementById("sub1").innerHTML="For";
	document.getElementById("prop4").innerHTML="<b>Iterations:</b>";
	document.getElementById("prop5").innerHTML="<b>Sub-plans:</b>";
	document.getElementById("prop51").innerHTML="Input Student";

	document.getElementById("prop51").style.backgroundColor="#FFFFFF";


}//show outer loop
```

```
function showinput()
{
	document.getElementById("studarrc").style.visibility="hidden";
	document.getElementById("iloopc").style.visibility="hidden";
	document.getElementById("iloopcbeg").style.visibility="hidden";
	document.getElementById("iloopc1").style.visibility="hidden";
	document.getElementById("iloopc2").style.visibility="hidden";
	document.getElementById("iloopcend").style.visibility="hidden";
	document.getElementById("main1").style.visibility="hidden";

	document.getElementById("inputdiv").style.visibility="visible";
	document.getElementById("setdiv").style.visibility="visible";
	document.getElementById("setdiv").style.backgroundColor="#FFFF99";

	document.getElementById("maincode").style.visibility="visible";

	document.getElementById("codebubble").style.visibility="visible";

}//show inner looop


function showbubble()
{
	document.getElementById("codebubble").style.visibility="hidden";

	document.getElementById("maincode").style.visibility="hidden";
	document.getElementById("maindiv").style.visibility="hidden";
	document.getElementById("arraydiv").style.visibility="hidden";
	document.getElementById("iloopdiv2").style.visibility="hidden";
	document.getElementById("inputdiv").style.visibility="hidden";
	document.getElementById("setdiv").style.visibility="hidden";

	//Bubble properties
	document.getElementById("pname").innerHTML="Bubble Sort";
	document.getElementById("ptype").innerHTML="Algorithm / Prob Slv";
	document.getElementById("prop3").innerHTML="<b>Objects:</b>";
	document.getElementById("sub1").innerHTML="Student";
	document.getElementById("prop4").innerHTML="";
	document.getElementById("sub2").innerHTML="";
	document.getElementById("prop4").style.backgroundColor="#E8E8E8";
	document.getElementById("sub2").style.backgroundColor="#E8E8E8";
	document.getElementById("prop51").innerHTML="Loop";
	document.getElementById("prop52").innerHTML="Comp Adj";
	document.getElementById("prop53").innerHTML="Swap";
	document.getElementById("prop54").innerHTML="Null";
	document.getElementById("prop51").style.backgroundColor="#FFFFFF";
	document.getElementById("prop52").style.backgroundColor="#FFFFFF";
	document.getElementById("prop53").style.backgroundColor="#FFFFFF";
	document.getElementById("prop54").style.backgroundColor="#FFFFFF";
	document.getElementById("sub1").style.backgroundColor="#FFFFFF";
	//hide main div and code

	document.getElementById("bubblediv").style.visibility="visible";
	document.getElementById("bubblecode").style.visibility="visible";
}//show bubble div and code
```

```
function oloop()
{
	document.getElementById("oloopdiv").style.visibility="visible";
	document.getElementById("oloopbeg").style.visibility="visible";
	document.getElementById("oloopend").style.visibility="visible";
}//show outer looop


function iloop()
{

	document.getElementById("iloopdiv").style.visibility="visible";
	document.getElementById("iloopbeg").style.visibility="visible";
	document.getElementById("iloopend").style.visibility="visible";

	document.getElementById("bubblecode").style.backgroundColor="#99FFFF";

}//show inner looop

function comp()
{

	document.getElementById("compdiv").style.visibility="visible";
	document.getElementById("compln").style.visibility="visible";

}//show comp plan

function swap()
{

	document.getElementById("swapdiv").style.visibility="visible";
	document.getElementById("nulldiv").style.visibility="visible";
	document.getElementById("arrldiv").style.visibility="visible";
	document.getElementById("arrrdiv").style.visibility="visible";
	document.getElementById("sw1").style.visibility="visible";
	document.getElementById("sw2").style.visibility="visible";
	document.getElementById("sw3").style.visibility="visible";
	document.getElementById("sw4").style.visibility="visible";

	document.getElementById("mainb").style.visibility="visible";

}//integrate swap and null plans

function mainBubble()
{
//show input student

	document.getElementById("main2").style.visibility="visible";
	document.getElementById("maindiv").style.visibility="visible";
	document.getElementById("arraydiv").style.visibility="visible";
	document.getElementById("iloopdiv2").style.visibility="visible";
	document.getElementById("inputdiv").style.visibility="visible";
	document.getElementById("setdiv").style.visibility="visible";
	document.getElementById("setdiv").style.backgroundColor="#FFFFFF";

//show bubble in main
```

```
		document.getElementById("bubblediv2").style.visibility="visible";
		document.getElementById("apparr1").style.visibility="visible";
//hide bubble

		document.getElementById("bubblediv").style.visibility="hidden";
		document.getElementById("bubblecode").style.visibility="hidden";
		document.getElementById("oloopdiv").style.visibility="hidden";
		document.getElementById("iloopdiv").style.visibility="hidden";
		document.getElementById("compdiv").style.visibility="hidden";
		document.getElementById("swapdiv").style.visibility="hidden";
		document.getElementById("nulldiv").style.visibility="hidden";
		document.getElementById("arrldiv").style.visibility="hidden";
		document.getElementById("arrrdiv").style.visibility="hidden";
		document.getElementById("oloopbeg").style.visibility="hidden";
		document.getElementById("oloopend").style.visibility="hidden";
		document.getElementById("iloopbeg").style.visibility="hidden";
		document.getElementById("iloopend").style.visibility="hidden";
		document.getElementById("compln").style.visibility="hidden";
		document.getElementById("sw1").style.visibility="hidden";
		document.getElementById("sw2").style.visibility="hidden";
		document.getElementById("sw3").style.visibility="hidden";
		document.getElementById("sw4").style.visibility="hidden";

		document.getElementById("mainb").style.visibility="hidden";

		document.getElementById("pname").innerHTML="Sort Students Main";
		document.getElementById("ptype").innerHTML="Main Function";
		document.getElementById("prop51").innerHTML="Loop";
		document.getElementById("prop52").innerHTML="Bubble Sort";
		document.getElementById("prop53").innerHTML="Input Student";
		document.getElementById("prop54").innerHTML="Display Student";

}//include bubble in main

function displayLoop()
{
		document.getElementById("oloopdiv2").style.visibility="visible";
		document.getElementById("apparr2").style.visibility="visible";

		document.getElementById("dl1").style.visibility="visible";
		document.getElementById("dl2").style.visibility="visible";
		document.getElementById("dl3").style.visibility="visible";

}//display loop

function getcomp()
{

		document.getElementById("displaydiv").style.visibility="visible";
		document.getElementById("compdiv2").style.visibility="visible";
		document.getElementById("getdiv").style.visibility="visible";
		document.getElementById("getdiv").style.backgroundColor="#FFFF99";
		document.getElementById("d1").style.visibility="visible";
		document.getElementById("d2").style.visibility="visible";
		document.getElementById("d3").style.visibility="visible";
		document.getElementById("d4").style.visibility="visible";
```

```
        document.getElementById("d5").style.visibility="visible";
        document.getElementById("d6").style.visibility="visible";
        document.getElementById("d7").style.visibility="visible";

        document.getElementById("bdiv").style.visibility="visible";
}//get comp

function fbuild()
{
                setTimeout("studarray()", 2000); //main code
                setTimeout("inloop()", 4000);
                setTimeout("showinput()", 6000);


}//end fbuild function

function fbuild2()
{
                showbubble();
                setTimeout("oloop()", 2000); //bubble sort code
                setTimeout("iloop()", 4000);
                setTimeout("comp()", 6000);
                setTimeout("swap()", 8000);

}//bubble code

function fbuild3()
{
                mainBubble();
                setTimeout("displayLoop()",2000);
                setTimeout("getcomp()",4000);
                setTimeout("showbutton()",10000);
}//display student code

<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if ((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.reload();
}
MM_reloadPage(true);
//-->
</script>
</head>

<body onload="fbuild(); document.getElementById('build').value=0;">
<div class="style2" id="Layer1" style="position:absolute; width:1009px; height:577px; z-index:1;
visibility: visible;">
  <div class="style2" id="phase" style="position:absolute; width:115px; height:76px; z-index:1; left:
781px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000;
top: 0px; visibility: visible;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Observation</div></td>
      </tr>
      <tr>
```

```
      <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Phase 1) </div></td>
     </tr>
    </table>
  </div>
  <div class="style2" id="level" style="position:absolute; width:115px; height:76px; z-index:2; left:
895px; top: 0px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000; visibility: visible;">    <br>
    <table width="110" border="0">
     <tr>
      <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Intermediate</div></td>
     </tr>
     <tr>
      <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Level 2) </div></td>
     </tr>
    </table>
  </div>
  <div id="Layer6" style="position:absolute; width:98px; height:407px; z-index:3; top: 76px;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000; visibility:
visible;">
  </div>
  <div id="properties" style="position:absolute; width:912px; height:105px; z-index:6; left: 98px; top:
483px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000;
visibility: visible;">
    <table style="table-layout:fixed" width="790" height="89" border="0">
     <tr>
      <td width="96"><strong>Plan Properties:</strong></td>
      <td width="130"> </td>
      <td width="90"> </td>
      <td width="96"> </td>
      <td width="86"> </td>
      <td width="99"> </td>
      <td width="87"> </td>
      <td width="72"> </td>
     </tr>
     <tr>
      <td><strong>Name:</strong></td>
      <td bgcolor="#FFFF99" id="pname">Sort Students Main</td>
      <td width="90" id="prop3"><strong>Objects:</strong></td>
      <td bgcolor="#FFFFFF" id="sub1">Student</td>
      <td id="prop5"><strong>Sub-plans</strong></td>
      <td bgcolor="#FFFFFF" id="prop51">Loop</td>
      <td bgcolor="#FFFFFF" id="prop53">Input Student</td>
      <td bgcolor="#E8E8E8" id="prop5i"> </td>
     </tr>
     <tr>
      <td><strong>Type:</strong></td>
      <td bgcolor="#FFFF99" id="ptype">Main Function</td>
      <td id="prop4"> </td>
      <td bgcolor="#E8E8E8" id="sub2"> </td>
      <td> </td>
      <td bgcolor="#FFFFFF" id="prop52">Bubble Sort</td>
      <td bgcolor="#FFFFFF" id="prop54">Display Student</td>
      <td bgcolor="#E8E8E8"> </td>
     </tr>
     <tr>
     </tr>
```

```
   </table>
</div>
</div>
<div class="style2" id="Layer2" style="position:absolute; width:98px; height:76px; z-index:9;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 0px none #000000; visibility:
visible;">
  <div align="center"><br>
    <span class="style5">WPOL</span></div>
</div>
<div class="style2" id="Layer3" style="position:absolute; width:683px; height:76px; z-index:10; left:
108px; top: 15px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000; visibility: visible;">
<div align="center"><br>
     <span class="style5">Sort Students Plan</span></div>
</div>
<div id="apDiv1">
  <div id="maindiv">
    <div id="setdiv">
      <div align="center" class="style10">Set Plan</div>
    </div>
    <div class="style7" id="inputdiv">
    <div align="center">Input Student Plan<br>
      [Mode:Console]<br>
    </div>
  </div>
<img src="../l1/pics/main.png" alt="" width="28" height="26">Sort Students Main Plan<br>
     <br>
     <div id="arraydiv">
       <div align="center"><img src="../l1/pics/sstud.png" width="26" height="26"><img
src="../l1/pics/sstud.png" width="26" height="26"><img src="../l1/pics/sstud.png" width="26"
height="26"><img src="../l1/pics/sstud.png" width="26" height="26"><img src="../l1/pics/sstud.png"
width="26" height="26">Student Array</div>
    </div>
     <div id="iloopdiv2"><img src="pics/testloop.png" width="220" height="130"></div>
     <div id="apparr1">
     <div align="center"><img src="../l1/pics/apparr.png" width="35" height="35"></div>
    </div>
<div id="bubblediv2">
     <div align="center"><img src="pics/sort1.png" width="58" height="30"> Bubble Sort</div>
    </div>
    <div id="apparr2"><img src="pics/apparr.png" width="35" height="35"></div>
    <br>
  </div>
  <div id="oloopdiv2"><img src="pics/testloop.png" width="220" height="130"></div>
  <div class="style7" id="displaydiv">
    <div align="center">Display Student Plan<br>
      [Mode:Console]</div>
  </div>
  <div id="getdiv">
    <div align="center" class="style10">Get Plan</div>
  </div>
  <div id="compdiv2">
    <div align="center">computeAverage</div>
  </div>
</div>
<div id="apDiv2">
```

```
  <input name="build" type="hidden" id="build" value="0">
</div>
<div id="bdiv">
  <input type="submit" name="bbuild" id="bbuild" value="View Plan Code"
onClick="document.location='code.html'">
</div>
<div id="bubblediv"><img src="pics/sort1.png" alt="" width="58" height="30"> Bubble Sort<br>
  <br>
</div>
<div id="iloopdiv"><img src="pics/inloop.png" width="255" height="185"></div>
<div id="oloopdiv"><img src="pics/outloop.png" width="315" height="240"></div>
<div id="compdiv">
  <div align="center">Compare <br>
  Adjacent</div>
</div>
<div id="swapdiv">
  <div align="center">Swap</div>
</div>
<div id="nulldiv">
  <div align="center">Null</div>
</div>
<div id="arrldiv"><img src="pics/arrl.png" width="30" height="38"></div>
<div id="arrrdiv"><img src="pics/arrr.png" width="34" height="35"></div>
<div id="bubblecode">
  <table width="525" height="354" border="0" cellpadding="0" cellspacing="0">

    <tr>
      <td bgcolor="#66CCCC" class="style14" id="oloopbeg" style="visibility:hidden"> for(int
pass=1;pass&lt;5;pass++){</td>
    </tr>
    <tr>
      <td bgcolor="#99FFFF" class="style14" id="iloopbeg" style="visibility:hidden">
   for(int scan=0; scan&lt;5-pass;scan++){</td>
    </tr>
    <tr>
      <td class="style14" id="bk1"> </td>
    </tr>
    <tr>
      <td bgcolor="#99FF99" class="style14" id="compln"
style="visibility:hidden">   if(studentArray[scan]-
&gt;computeAverage()&gt;studentArray[scan+1]-&gt;computeAverage()){</td>
    </tr>
    <tr>
      <td class="style14" id="bk2"> </td>
    </tr>
    <tr>
      <td bgcolor="#FFFF66" class="style14" id="sw1" style="visibility:hidden">
      Student* temp;</td>
    </tr>
    <tr>
      <td bgcolor="#FFFF66" class="style14" id="sw2" style="visibility:hidden">
      temp=studentArray[scan];</td>
    </tr>
    <tr>
      <td bgcolor="#FFFF66" class="style14" id="sw3" style="visibility:hidden">
      studentArray[scan]=studentArray[scan+1];</td>
```

```
   </tr>
   <tr>
     <td bgcolor="#FFFF66" class="style14" id="sw4" style="visibility:hidden">
      studentArray[scan+1]=temp; } </td>
   </tr>
   <tr>
     <td class="style14" id="bk3"> </td>
   </tr>
   <tr>
     <td bgcolor="#99FFFF" class="style14" id="iloopend" style="visibility:hidden">
   }//end scan</td>
   </tr>
   <tr>
     <td bgcolor="#66CCCC" class="style14" id="oloopend" style="visibility:hidden"> }//end
pass</td>
   </tr>
  </table>
</div>
<div id="maincode">
  <iframe src="mainc1.html" width="100%" height="100%" frameborder="0"></iframe>
</div>
<div id="main1">
  <table width="360" border="0" cellpadding="0" cellspacing="0">
   <tr>
     <td> </td>
   </tr>
   <tr>
     <td><span class="style16">int main()</span></td>
   </tr>
   <tr>
     <td><span class="style16">{</span></td>
   </tr>
   <tr>
     <td bgcolor="#898989" id="studarrc" style="visibility:hidden"><span class="style16 style4 style7
style14">   Student *studentArray[5];</span></td>
   </tr>
   <tr>
     <td style="visibility:hidden"> </td>
   </tr>
   <tr>
     <td bgcolor="#66CCCC" id="iloopc" style="visibility:hidden"><span class="style14"><span
class="style18">   for(int i=0; i&lt;5; i++)</span></span></td>
   </tr>
   <tr>
     <td bgcolor="#66CCCC" id="iloopcbeg" style="visibility:hidden"><span
class="style19">   {</span></td>
   </tr>
   <tr>
     <td bgcolor="#66CCCC" id="iloopc1" style="visibility:hidden"><span class="style10">
    </span></td>
   </tr>
   <tr>
     <td bgcolor="#66CCCC" id="iloopc2" style="visibility:hidden"><span
class="style10">   </span></td>
   </tr>
   <tr>
```

```
    <td bgcolor="#66CCCC" id="iloopcend" style="visibility:hidden"><span class="style10"><span
class="style19">   }//for loop</span></span></td>
   </tr>
  </table>
</div>
<div id="codebubble">
  <input type="submit" name="bubbleb" id="bubbleb" value="Code Bubble" onClick="fbuild2()">
</div>
<div id="mainb">
  <input type="submit" name="mainc" id="mainc" value="Main Code" onClick="fbuild3()">
</div>
<div id="main2">
  <table width="360" height="334" border="0" cellpadding="0" cellspacing="0">
   <tr>
    <td bgcolor="#000000" id="d1" style="visibility:hidden"><span class="style20"> cout&lt;&lt;
&quot;Student id    First       Last    
email       Average &quot;&lt;&lt;endl;</span></td>
   </tr>
   <tr>
    <td><span class="style14"></span></td>
   </tr>
   <tr>
    <td bgcolor="#66CCCC" id="dl1" style="visibility:hidden"><span class="style14"> for(int
j=0;j&lt;5; j++)</span></td>
   </tr>
   <tr>
    <td bgcolor="#66CCCC" id="dl2" style="visibility:hidden"><span class="style14">{</span></td>
   </tr>
   <tr>
    <td bgcolor="#000000" id="d2" style="visibility:hidden"><span class="style14">
   <span class="style7">cout&lt;&lt;<span class="style21">studentArray[j]-
&gt;getId()</span>&lt;&lt;&quot;     &quot;;</span></span></td>
   </tr>
   <tr>
    <td bgcolor="#333333" id="d3" style="visibility:hidden"><span class="style14">
  <span class="style7"> cout&lt;&lt;<span class="style21">studentArray[j]-
&gt;getFname()</span>&lt;&lt;&quot;     &quot;;</span></span></td>

   </tr>
   <tr>
    <td bgcolor="#000000" id="d4" style="visibility:hidden"><span class="style14">
  <span class="style7"> cout&lt;&lt;<span class="style21">studentArray[j]-
&gt;getLname()</span>&lt;&lt;&quot;     &quot;;</span></span></td>

   </tr>
   <tr>
    <td bgcolor="#333333" id="d5" style="visibility:hidden"><span class="style14">
   <span class="style7">cout&lt;&lt;<span class="style21">studentArray[j]-
&gt;getEmail()</span>&lt;&lt;&quot;     &quot;;</span></span></td>

   </tr>
   <tr>
    <td bgcolor="#333333" id="d6" style="visibility:hidden"><span class="style20">
   cout&lt;&lt;<span class="style25">studentArray[j]-
&gt;computeAverage()</span>&lt;&lt;&quot;     &quot;;</span></td>
   </tr>
```

```
   <tr>
     <td bgcolor="#000000" id="d7" style="visibility:hidden"><span class="style20">
&amp;nbsp;&amp;nbsp;&amp;nbsp;cout&amp;lt;&amp;lt;endl;</span></td>
   </tr>
   <tr>
     <td bgcolor="#66CCCC" id="dl3" style="visibility:hidden"><span class="style14">}</span></td>
   </tr>
   <tr>
     <td><span class="style14"> return 0;</span></td>
   </tr>
   <tr>
     <td><span class="style14">}//main</span></td>
   </tr>
 </table>
</div>
</body>
</html>
```

Observation Phase Sample 5

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>WPOL</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<script language="JavaScript" type="text/JavaScript">

<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if ((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.reload();
}
MM_reloadPage(true);
//-->

</script>
</head>

<body>
<div class="style2" id="Layer1" style="position:absolute; width:1009px; height:577px; z-index:1;
visibility: visible;">
  <div class="style2" id="phase" style="position:absolute; width:115px; height:76px; z-index:1; left:
781px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000;
top: 0px; visibility: visible;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Observation</div></td>
      </tr>
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Phase 1) </div></td>
      </tr>
    </table>
```

```
 </div>
 <div class="style2" id="level" style="position:absolute; width:115px; height:76px; z-index:2; left:
895px; top: 0px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000; visibility: visible;">    <br>
   <table width="110" border="0">
     <tr>
       <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Intermediate</div></td>
     </tr>
     <tr>
       <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Level 2) </div></td>
     </tr>
   </table>
 </div>
 <div id="Layer6" style="position:absolute; width:98px; height:407px; z-index:3; top: 76px;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none #000000; visibility:
visible;">
   <table width="94" height="358" border="0">
     <tr>
       <td valign="top"><p align="center"><img src="pics/personIcon.png" width="81"
height="81"></p>
       <p><img src="pics/student3.png" width="90" height="90"></p></td>
     </tr>
   </table>
 </div>
 </div>
<div class="style2" id="Layer2" style="position:absolute; width:98px; height:76px; z-index:2;
background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 0px none #000000; visibility:
visible;">
 <div align="center"><br>
   <span class="style5">WPOL</span></div>
</div>
<div class="style2" id="Layer3" style="position:absolute; width:683px; height:76px; z-index:3; left:
108px; top: 15px; background-color: #E8E8E8; layer-background-color: #E8E8E8; border: 1px none
#000000; visibility: visible;">
<div align="center"><br>
     <span class="style5">Sort Students Plan</span></div>
</div>
<div id="apDiv1">
 <pre>
<code><span style="font: 10pt Courier New;"><span class="cpp1-
preprocessor">#include&lt;iostream&gt;
#include&lt;string&gt;
</span><span class="cpp1-reservedword">using</span><span class="cpp1-space"> </span><span
class="cpp1-reservedword">namespace</span><span class="cpp1-space"> std;

</span><span class="cpp1-reservedword">class</span><span class="cpp1-space"> Person
{
 </span><span class="cpp1-reservedword">private</span><span class="cpp1-symbol">:
                    string fname;
                    string lname;
                    string email;
 </span><span class="cpp1-reservedword">public</span><span class="cpp1-symbol">:
                    Person();
                    Person(string f, string l, string a);
                    ~Person();
```

```
                    </span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
setFname(string f);
                    </span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
setLname(string l);
                    </span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
setEmail(string e);
                    string getFname();
                    string getLname();
                    string getEmail();};</span>
<span class="cpp1-comment">//Personclass</span><span class="cpp1-identifier">Person::Person()
{
        fname=</span><span class="cpp1-string">&amp;quot;&amp;quot;</span><span class="cpp1-
symbol">;
        lname=</span><span class="cpp1-string">&amp;quot;&amp;quot;</span><span class="cpp1-
symbol">;
        email=</span><span class="cpp1-string">&amp;quot;&amp;quot;</span><span class="cpp1-
symbol">;
}</span><span class="cpp1-comment">//Person class default constructor

</span><span class="cpp1-identifier">Person::Person(string f, string l, string e)
{
        fname=f;
        lname=l;
        email=e;
}</span><span class="cpp1-comment">//Person class contructor

</span><span class="cpp1-identifier">Person::~Person(){}

</span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
Person::setFname(string f)
{
        fname=f;
}</span><span class="cpp1-comment">//setFname function

</span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
Person::setLname(string l)
{
        lname=l;
}</span><span class="cpp1-comment">//setLname function

</span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
Person::setEmail(string e)
{
        email=e;
}</span><span class="cpp1-comment">//setEmail function

</span><span class="cpp1-identifier">string Person::getFname()
{
        </span><span class="cpp1-reservedword">return</span><span class="cpp1-space"> fname;
}</span><span class="cpp1-comment">//getFname function

</span><span class="cpp1-identifier">string Person::getLname()
{
        </span><span class="cpp1-reservedword">return</span><span class="cpp1-space"> lname;
}</span><span class="cpp1-comment">//getLname function
```

```
</span><span class="cpp1-identifier">string Person::getEmail()
{
            </span><span class="cpp1-reservedword">return</span><span class="cpp1-space"> email;
}</span><span class="cpp1-comment">//getEmail function

</span><span class="cpp1-reservedword">class</span><span class="cpp1-space"> Student:
</span><span class="cpp1-reservedword">public</span><span class="cpp1-space"> Person
{
 </span><span class="cpp1-reservedword">private</span><span class="cpp1-symbol">:
        </span><span class="cpp1-reservedword">int</span><span class="cpp1-space"> id;
        </span><span class="cpp1-reservedword">double</span><span class="cpp1-space">
assignmentScore;
        </span><span class="cpp1-reservedword">double</span><span class="cpp1-space">
midtermScore;
        </span><span class="cpp1-reservedword">double</span><span class="cpp1-space"> finalScore;

 </span><span class="cpp1-reservedword">public</span><span class="cpp1-symbol">:
       Student();
       Student(</span><span class="cpp1-reservedword">int</span><span class="cpp1-space"> i,
</span><span class="cpp1-reservedword">double</span><span class="cpp1-space"> a, </span><span
class="cpp1-reservedword">double</span><span class="cpp1-space"> m, </span><span class="cpp1-
reservedword">double</span><span class="cpp1-space"> f);
       ~Student();
       </span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
setId(</span><span class="cpp1-reservedword">int</span><span class="cpp1-space"> i);
       </span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
setAssignmentScore(</span><span class="cpp1-reservedword">double</span><span class="cpp1-
space"> a);
       </span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
setMidtermScore(</span><span class="cpp1-reservedword">double</span><span class="cpp1-
space"> m);
       </span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
setFinalScore(</span><span class="cpp1-reservedword">double</span><span class="cpp1-space">
f);
       </span><span class="cpp1-reservedword">double</span><span class="cpp1-space">
computeAverage();
       </span><span class="cpp1-reservedword">int</span><span class="cpp1-space"> getId();
};</span><span class="cpp1-comment">//Student class

</span><span class="cpp1-identifier">Student::Student()
{
   id=</span><span class="cpp1-number">0</span><span class="cpp1-symbol">;
   assignmentScore=</span><span class="cpp1-number">0</span><span class="cpp1-symbol">;
   midtermScore=</span><span class="cpp1-number">0</span><span class="cpp1-symbol">;
   finalScore=</span><span class="cpp1-number">0</span><span class="cpp1-symbol">;
}</span><span class="cpp1-comment">//Student class default constructor

</span><span class="cpp1-identifier">Student::Student(</span><span class="cpp1-
reservedword">int</span><span class="cpp1-space"> i, </span><span class="cpp1-
reservedword">double</span><span class="cpp1-space"> a, </span><span class="cpp1-
reservedword">double</span><span class="cpp1-space"> m, </span><span class="cpp1-
reservedword">double</span><span class="cpp1-space"> f)
{
    id = i;
    assignmentScore = a;
    midtermScore = m;
```

```
    finalScore = f;
}</span><span class="cpp1-comment">//Student class constructor

</span><span class="cpp1-identifier">Student::~Student(){}

</span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
Student::setId(</span><span class="cpp1-reservedword">int</span><span class="cpp1-space"> i)
{
    id = i;
}</span><span class="cpp1-comment">//setId function
</span><span class="cpp1-space">
</span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
Student::setAssignmentScore(</span><span class="cpp1-reservedword">double</span><span
class="cpp1-space"> a)
{
    assignmentScore = a;
}</span><span class="cpp1-comment">//setAssignmentScore function

</span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
Student::setMidtermScore(</span><span class="cpp1-reservedword">double</span><span
class="cpp1-space"> m)
{
    midtermScore = m;
}</span><span class="cpp1-comment">//setMidtermScore funtion

</span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
Student::setFinalScore(</span><span class="cpp1-reservedword">double</span><span class="cpp1-
space"> f)
{
    finalScore = f;
}</span><span class="cpp1-comment">//setFinalScore function

</span><span class="cpp1-reservedword">double</span><span class="cpp1-space">
Student::computeAverage()
{
     </span><span class="cpp1-reservedword">double</span><span class="cpp1-space"> average;
     </span><span class="cpp1-reservedword">double</span><span class="cpp1-space"> sum;
     sum = </span><span class="cpp1-number">0</span><span class="cpp1-symbol">;
     sum = assignmentScore + midtermScore + finalScore;
     average = sum / </span><span class="cpp1-number">3</span><span class="cpp1-symbol">;
     </span><span class="cpp1-reservedword">return</span><span class="cpp1-space"> average;
}</span><span class="cpp1-comment">//computeAverage function

</span><span class="cpp1-reservedword">int</span><span class="cpp1-space"> Student::getId()
{
   </span><span class="cpp1-reservedword">return</span><span class="cpp1-space"> id;
}</span><span class="cpp1-comment">//getId function

</span><span class="cpp1-reservedword">int</span><span class="cpp1-space"> main()
{
        Student *studentArray[</span><span class="cpp1-number">5</span><span class="cpp1-
symbol">];

        </span><span class="cpp1-reservedword">int</span><span class="cpp1-space"> id;
        string fname, lname, email, phone;
```

```
        </span><span class="cpp1-reservedword">double</span><span class="cpp1-space">
assignment, midterm, final;

        </span><span class="cpp1-reservedword">for</span><span class="cpp1-
symbol">(</span><span class="cpp1-reservedword">int</span><span class="cpp1-space">
i=</span><span class="cpp1-number">0</span><span class="cpp1-symbol">; i&amp;lt;</span><span
class="cpp1-number">5</span><span class="cpp1-symbol">; i++)
        {
                studentArray[i]=</span><span class="cpp1-reservedword">new</span><span
class="cpp1-space"> Student;

                cout&amp;lt;&amp;lt;</span><span class="cpp1-string">&amp;quot;Enter student id:
&amp;quot;</span><span class="cpp1-symbol">;
                cin&amp;gt;&amp;gt;id;
                studentArray[i]-&amp;gt;setId(id);

                cout&amp;lt;&amp;lt;</span><span class="cpp1-string">&amp;quot;Enter student's first name:
&amp;quot;</span><span class="cpp1-symbol">;
                cin&amp;gt;&amp;gt;fname;
                studentArray[i]-&amp;gt;setFname(fname);

                cout&amp;lt;&amp;lt;</span><span class="cpp1-string">&amp;quot;Enter student's last name:
&amp;quot;</span><span class="cpp1-symbol">;
                cin&amp;gt;&amp;gt;lname;
                studentArray[i]-&amp;gt;setLname(lname);

                cout&amp;lt;&amp;lt;</span><span class="cpp1-string">&amp;quot;Enter student's email:
&amp;quot;</span><span class="cpp1-symbol">;
                cin&amp;gt;&amp;gt;email;
                studentArray[i]-&amp;gt;setEmail(email);

                cout&amp;lt;&amp;lt;</span><span class="cpp1-string">&amp;quot;Enter assignment score:
&amp;quot;</span><span class="cpp1-symbol">;
                cin&amp;gt;&amp;gt;assignment;
                studentArray[i]-&amp;gt;setAssignmentScore(assignment);

                cout&amp;lt;&amp;lt;</span><span class="cpp1-string">&amp;quot;Enter midterm score:
&amp;quot;</span><span class="cpp1-symbol">;
                cin&amp;gt;&amp;gt;midterm;
                studentArray[i]-&amp;gt;setMidtermScore(midterm);

                cout&amp;lt;&amp;lt;</span><span class="cpp1-string">&amp;quot;Enter final exam score:
&amp;quot;</span><span class="cpp1-symbol">;
                cin&amp;gt;&amp;gt;final;
                studentArray[i]-&amp;gt;setFinalScore(final);
        }</span><span class="cpp1-comment">//for loop
</span><span class="cpp1-space">
   </span><span class="cpp1-reservedword">for</span><span class="cpp1-symbol">(</span><span
class="cpp1-reservedword">int</span><span class="cpp1-space"> pass=</span><span class="cpp1-
number">1</span><span class="cpp1-symbol">;pass&amp;lt;</span><span class="cpp1-
number">5</span><span class="cpp1-symbol">;pass++){

                </span><span class="cpp1-reservedword">for</span><span class="cpp1-
symbol">(</span><span class="cpp1-reservedword">int</span><span class="cpp1-space">
scan=</span><span class="cpp1-number">0</span><span class="cpp1-symbol">;
scan&amp;lt;</span><span class="cpp1-number">5</span><span class="cpp1-symbol">-pass;scan++){
```

```
                                </span><span class="cpp1-reservedword">if</span><span class="cpp1-
symbol">(studentArray[scan]-&gt;computeAverage()&gt;studentArray[scan+</span><span
class="cpp1-number">1</span><span class="cpp1-symbol">]-&gt;computeAverage()){
                                        Student* temp;
                                temp=studentArray[scan];
                                studentArray[scan]=studentArray[scan+</span><span class="cpp1-
number">1</span><span class="cpp1-symbol">];
                                studentArray[scan+</span><span class="cpp1-number">1</span><span
class="cpp1-symbol">]=temp; }

                        }</span><span class="cpp1-comment">//end scan}//end pass

</span><span class="cpp1-space">      cout&lt;&lt; </span><span class="cpp1-string">&quot;Student
id  First  Last  email  Average &quot;</span><span class="cpp1-symbol">&lt;&lt;endl;

     </span><span class="cpp1-reservedword">for</span><span class="cpp1-symbol">(</span><span
class="cpp1-reservedword">int</span><span class="cpp1-space"> j=</span><span class="cpp1-
number">0</span><span class="cpp1-symbol">;j&lt;</span><span class="cpp1-
number">5</span><span class="cpp1-symbol">; j++){

                        cout&lt;&lt;studentArray[j]-&gt;getId()&lt;&lt;</span><span class="cpp1-
string">&quot;  &quot;</span><span class="cpp1-symbol">;
                        cout&lt;&lt;studentArray[j]-&gt;getFname()&lt;&lt;</span><span class="cpp1-
string">&quot;  &quot;</span><span class="cpp1-symbol">;
                        cout&lt;&lt;studentArray[j]-&gt;getLname()&lt;&lt;</span><span class="cpp1-
string">&quot;  &quot;</span><span class="cpp1-symbol">;
                        cout&lt;&lt;studentArray[j]-&gt;getEmail()&lt;&lt;</span><span class="cpp1-
string">&quot;  &quot;</span><span class="cpp1-symbol">;
                        cout&lt;&lt;studentArray[j]-&gt;computeAverage()&lt;&lt;</span><span
class="cpp1-string">&quot;  &quot;</span><span class="cpp1-symbol">;

                        cout&lt;&lt;endl;}

   </span><span class="cpp1-reservedword">return</span><span class="cpp1-space"> </span><span
class="cpp1-number">0</span><span class="cpp1-symbol">;
}</span><span class="cpp1-comment">//main

</span></span>
</code></pre>
</div>
<div id="apDiv2"></div>
</body>
</html>
```

Integration Phase Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>WPOL</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<script language="JavaScript" type="text/JavaScript">
```

```
<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if ((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.reload();
}
MM_reloadPage(true);
//-->

function clearCheckBoxes()
{
	document.getElementById("inputbx").checked=false;
	document.getElementById("loopbx").checked=false;
	document.getElementById("databx").checked=false;
	document.getElementById("utilitiesbx").checked=false;
	document.getElementById("displaybx").checked=false;
	document.getElementById("functionbx").checked=false;
	document.getElementById("selectInt").value="Integration Mode";
}

function checkAns()
{
	var correct=1;

		if(!document.getElementById("databx").checked ||
				!document.getElementById("utilitiesbx").checked ||
					!document.getElementById("functionbx").checked)
					{
						alert("Class Component Missing");

	document.getElementById("selectInt").value="Integration Mode";
						correct=0;
					}

		if(document.getElementById("inputbx").checked)
		{
			alert("Input Employee Plan is not a Class Component");
			document.getElementById("inputdiv2").style.visibility="hidden";
			document.getElementById("inputbx").checked=false;
			document.getElementById("selectInt").value="Integration Mode";
			correct=0;
		}
		if(document.getElementById("loopbx").checked)
		{
			alert("Loop Plan is not a Class Component");
			document.getElementById("looppdiv2").style.visibility="hidden";
			document.getElementById("loopbx").checked=false;
			document.getElementById("selectInt").value="Integration Mode";
			correct=0;
		}
		if(document.getElementById("displaybx").checked)
		{
			alert("Display Employee Plan is not a Class Component");
			document.getElementById("displaydiv2").style.visibility="hidden";
			document.getElementById("displaybx").checked=false;
			document.getElementById("selectInt").value="Integration Mode";
```

```
                        correct=0;
                }
        if(document.getElementById("databx").checked &&
                document.getElementById("utilitiesbx").checked &&
                        document.getElementById("functionbx").checked && correct)
                                {

        if(document.getElementById("selectInt").value=="Embedded")
                                        {
                                                alert("Correct!");
                                                document.location="p2.html";
                                        }
                                        else
                                        {
                                                alert("Incorrect Integration Mode");

        document.getElementById("selectInt").value="Integration Mode";
                                        }
                                }//end if
}

function ibx()
{
        document.getElementById("inputdiv2").style.visibility="visible";
}

function lbx()
{
        document.getElementById("looppdiv2").style.visibility="visible";
}
function dbx()
{
        document.getElementById("empdata").style.visibility="visible";

}
function disbx()
{
        document.getElementById("displaydiv2").style.visibility="visible";
}
function ubx()
{
        document.getElementById("emput").style.visibility="visible";
}
function fbx()
{
        document.getElementById("empfunc").style.visibility="visible";
}
function hibx()
{
        document.getElementById("inputdiv2").style.visibility="hidden";
}

function hlbx()
{
        document.getElementById("looppdiv2").style.visibility="hidden";
}
```

```
function hdbx()
{
	document.getElementById("empdata").style.visibility="hidden";

}
function hdisbx()
{
	document.getElementById("displaydiv2").style.visibility="hidden";
}
function hubx()
{
	document.getElementById("emput").style.visibility="hidden";
}
function hfbx()
{
	document.getElementById("empfunc").style.visibility="hidden";
}
</script>
</head>

<body onLoad="clearCheckBoxes()">
<div class="style7" id="inputdiv2">
  <div align="center"><span class="style13">Input Employee Plan<br>
    [Mode:Data File]</span><br>
  </div>
</div>
<div class="style7" id="displaydiv2">
  <div align="center" class="style13">Display Employee Plan<br>
    [Mode:Console]</div>
</div>
<div id="looppdiv2"><img src="pics/loop2.png" alt="" width="132" height="78"></div>
<div class="style2" id="Layer1" style="position:absolute; width:1009px; height:577px; z-index:1">
  <div class="style2" id="phase" style="position:absolute; width:115px; height:76px; z-index:1; left:
781px; background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px none #000000;
top: 0px;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Integration</div></td>
      </tr>
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Phase 2) </div></td>
      </tr>
    </table>
  </div>
  <div class="style2" id="level" style="position:absolute; width:115px; height:76px; z-index:2; left:
895px; top: 0px; background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px none
#000000;">    <br>
    <table width="110" border="0">
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Beginner</div></td>
      </tr>
      <tr>
        <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Level 1) </div></td>
      </tr>
    </table>
  </div>
```

```
  <div id="Layer6" style="position:absolute; width:98px; height:407px; z-index:3; top: 76px;
background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px none #000000;
visibility: visible; left: 2px;">
  </div>
  <div id="properties" style="position:absolute; width:912px; height:106px; z-index:6; left: 98px; top:
483px; background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px none
#000000;">
    <table width="730" height="101" border="0">
      <tr>
        <td width="126"><strong>Plan Properties:</strong></td>
      </tr>
      <tr>
        <td><strong>Name:</strong></td>
        <td bgcolor="#FFFFFF">Employee Object </td>
        <td><strong>Sub-plans:</strong></td>
        <td bgcolor="#FFFFFF"> </td>
        <td bgcolor="#EBEBEB"> </td>
        <td> </td>
      </tr>
      <tr>
        <td><strong>Type:</strong></td>
        <td bgcolor="#FFFFFF">Class</td>
      </tr>
    </table>
</div>
</div>
<div class="style2" id="Layer2" style="position:absolute; width:98px; height:76px; z-index:5;
background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 0px none #000000;">
  <div align="center"><br>
    <span class="style5">WPOL</span></div>
</div>
<div class="style2" id="Layer3" style="position:absolute; width:683px; height:76px; z-index:6; left:
108px; top: 15px; background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px
none #000000;">
<div align="center"><br>
     <span class="style5">Payroll  Plan</span></div>
</div>
<div id="apDiv1">
  <div id="datadiv"><img src="pics/data.png" alt="" width="24" height="24"><span
class="style13">Data Member Plan</span></div>
  <div id="funcdiv"><img src="pics/func.png" width="29" height="26"><span
class="style13">Member Function Plan</span></div>
  <div id="utilitiesdiv"><img src="pics/tools.png" width="24" height="24"><span
class="style13">Object Utilities</span></div>
  <div class="style7" id="inputdiv">
    <div align="center"><span class="style13">Input Employee Plan<br>
      [Mode:Data File]</span><br>
    </div>
  </div>
  <div id="looppdiv"><img src="pics/loop2.png" width="132" height="78"></div>
  <div class="style7" id="displaydiv">
    <div align="center" class="style13">Display Employee Plan<br>
      [Mode:Console]</div>
  </div>
  <div id="apDiv3">
    <input type="checkbox" id="loopbx" onClick="this.checked ? lbx() : hlbx()">
```

```
 </div>
 <div id="apDiv4">
  <input type="checkbox" id="databx" onClick="this.checked ? dbx() : hdbx()">
 </div>
 <div id="apDiv5">
  <input type="checkbox" id="utilitiesbx" onClick="this.checked ? ubx() : hubx()">
 </div>
 <div id="apDiv6">
  <input type="checkbox" id="displaybx" onClick="this.checked ? disbx() : hdisbx()">
 </div>
 <div id="apDiv7">
  <input type="checkbox" id="functionbx" onClick="this.checked ? fbx() : hfbx()" >
 </div>
 </div>
<div id="apDiv2"></div>
<div id="apDiv14">
 <table width="228" height="285" border="0" cellpadding="0" cellspacing="0">
  <tr>
   <td bgcolor="#D6D6D6"><span class="style13"><img src="pics/semp.png" width="26"
height="26">Employee Object Plan</span></td>
  </tr>
  <tr>
   <td bgcolor="#D6D6D6" id="empdata" style="visibility:hidden"><img src="pics/data.png" alt=""
width="24" height="24"><span class="style13">Data Member Plan</span></td>
  </tr>
  <tr>
   <td bgcolor="#D6D6D6" id="empfunc" style="visibility:hidden"><img src="pics/func.png"
width="29" height="26"><span class="style13">Member Function Plan</span></td>
  </tr>
  <tr>
   <td bgcolor="#D6D6D6" id="emput" style="visibility:hidden"><img src="pics/tools.png"
width="24" height="24"><span class="style13">Object Utilities</span></td>
  </tr>
 </table>
</div>
<div class="style15" id="apDiv15">
 <div align="center">Please select the Class Components to Integrate into the Employee Object Plan
<br>
  and then
 Select the Integration Mode</div>
</div>
<div id="apDiv16">
 <input type="checkbox" id="inputbx" onClick="this.checked ? ibx() : hibx()">
</div>
<div id="apDiv22">
 <select name="selectInt" id="selectInt" onChange="checkAns()">
          <option value="Integration Mode">Integration Mode</option>
          <option value="Appended">Appended</option>
  <option value="Branched">Branched</option>
  <option value="Embedded">Embedded</option>
  <option value="Interleaved">Interleaved</option>
 </select>
</div>
</body>
</html>
```

Creation Phase Sample

```
#include<string>
#include<iostream>
using namespace std;
class Cgi{
      string env;
   public:
      Cgi(){
         cin>>env;
         cout<<"Content-type: text/html\n\n";
 }
   string getStr(string name){
      int i;
      int loc;// = npos;
      loc = env.find(name);
      loc = loc + name.length() + 1;
      char s[100];
      for(i=0; env[loc]!='&' && env[loc]!=0; i++){
         s[i] = env[loc];
         loc++; }
      s[i]=0;
      return s; }
   };
int main(){
   Cgi obj;
   string cName;
   cout<<"<html>";
   cName = obj.getStr("setNameI");
   cout<<"class "<<cName<<"{<br>"; //class name
   //print data members
```

```
    if(obj.getStr("vNameI1")!=""){

    cout<<obj.getStr("vAcc1")<<": <br>   "<<obj.getStr("vType1")<<"
"<<obj.getStr("vNameI1")<<";<br>";}

    if(obj.getStr("vNameI2")!=""){

    if(obj.getStr("vAcc2")!=obj.getStr("vAcc1"))

    cout<<obj.getStr("vAcc2")<<": <br>";

    cout<<"   "<<obj.getStr("vType2")<<" "<<obj.getStr("vNameI2")<<";<br>";}

    if(obj.getStr("vNameI3")!=""){

    if(obj.getStr("vAcc3")!=obj.getStr("vAcc2"))

    cout<<obj.getStr("vAcc3")<<": <br>";

    cout<<"   "<<obj.getStr("vType3")<<" "<<obj.getStr("vNameI3")<<";<br>";}

    //constructor and destructor

    cout<<"public: <br>";

    cout<<"   "<<cName<<"();<br>";

    cout<<"   ~"<<cName<<"();<br>";

    //print member function prototypes

    if(obj.getStr("fNameI1")!=""){

    if(obj.getStr("fAcc1")!="public") cout<<obj.getStr("fAcc1")<<": <br>";

    cout<<"   "<<obj.getStr("fRType1")<<" "<<obj.getStr("fNameI1")<<"(";

    if(obj.getStr("fParamN11")!="") cout<<obj.getStr("fParamT11")<<" "<<obj.getStr("fParamN11");

    if(obj.getStr("fParamN21")!="") cout<<", "<<obj.getStr("fParamT21")<<"
"<<obj.getStr("fParamN21");

    if(obj.getStr("fParamN31")!="") cout<<", "<<obj.getStr("fParamT31")<<"
"<<obj.getStr("fParamN31");

    cout<<");<br>"; }

    if(obj.getStr("fNameI2")!=""){

    if(obj.getStr("fAcc2")!=obj.getStr("fAcc1"))

    cout<<"   "<<obj.getStr("fAcc2")<<": <br>";

    cout<<"   "<<obj.getStr("fRType2")<<" "<<obj.getStr("fNameI2")<<"(";

    if(obj.getStr("fParamN12")!="") cout<<obj.getStr("fParamT12")<<" "<<obj.getStr("fParamN12");

    if(obj.getStr("fParamN22")!="") cout<<", "<<obj.getStr("fParamT22")<<"
"<<obj.getStr("fParamN22");
```

```
   if(obj.getStr("fParamN32")!="") cout<<", "<<obj.getStr("fParamT32")<<"
"<<obj.getStr("fParamN32");

   cout<<");<br>";}

   //set and get functions

   if(obj.getStr("vNameI1")!=""){

   cout<<"  &nbspvoid
set_"<<obj.getStr("vNameI1")<<"("<<obj.getStr("vType1")<<");<br>";}

   if(obj.getStr("vNameI2")!=""){

   cout<<"  &nbspvoid
set_"<<obj.getStr("vNameI2")<<"("<<obj.getStr("vType2")<<");<br>";}

   if(obj.getStr("vNameI3")!=""){

   cout<<"  &nbspvoid
set_"<<obj.getStr("vNameI3")<<"("<<obj.getStr("vType3")<<");<br>";}

   if(obj.getStr("vNameI1")!=""){

   cout<<"   "<<obj.getStr("vType1")<<"
get_"<<obj.getStr("vNameI1")<<"();<br>";}

   if(obj.getStr("vNameI2")!=""){

   cout<<"   "<<obj.getStr("vType2")<<"
get_"<<obj.getStr("vNameI2")<<"();<br>";}

   if(obj.getStr("vNameI3")!=""){

   cout<<"   "<<obj.getStr("vType3")<<"
get_"<<obj.getStr("vNameI3")<<"();<br>";}

   cout<<"};<br><br>";

   return 0;

}


<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>WPOL</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<script language="JavaScript" type="text/JavaScript">
<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if ((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.reload();
}
```

```
MM_reloadPage(true);
//-->

function setName()
{
	document.getElementById("objName").innerHTML=("<img
src='pics/obj2.png'>"+setObjPlan.setNameI.value+" Object Plan");
	document.getElementById("objDiv").style.visibility="visible";
	document.getElementById("pName").innerHTML=(setObjPlan.setNameI.value+" Object");

	document.getElementById("setObjDiv").style.visibility="hidden";
	document.getElementById("setV1Div").style.visibility="visible";
	document.getElementById("empdata").style.backgroundColor="#CCFFFF";
	document.getElementById("objName").style.backgroundColor="#D6D6D6";

}

function setV1()
{
	document.getElementById("v1").innerHTML=("<center>"+setObjPlan.vNameI1.value+"</ce
nter>");
	document.getElementById("v1").style.backgroundColor="#FFFFFF";
	document.getElementById("setV1Div").style.visibility="hidden";
	document.getElementById("setV2Div").style.visibility="visible";
}
function setV2()
{
	document.getElementById("v2").innerHTML=("<center>"+setObjPlan.vNameI2.value+"</ce
nter>");
	document.getElementById("v2").style.backgroundColor="#FFFFFF";
	document.getElementById("setV2Div").style.visibility="hidden";
	document.getElementById("setV3Div").style.visibility="visible";
}
function setV3()
{
	document.getElementById("v3").innerHTML=("<center>"+setObjPlan.vNameI3.value+"</ce
nter>");
	document.getElementById("v3").style.backgroundColor="#FFFFFF";
	document.getElementById("setV2Div").style.visibility="hidden";
	document.getElementById("funcDiv1").style.visibility="visible";
	document.getElementById("empdata").style.backgroundColor="#D6D6D6";
	document.getElementById("empfunc").style.backgroundColor="#CCFFFF";
}
function setFunc1()
{
	document.getElementById("f1").innerHTML=("<center>"+setObjPlan.fNameI1.value+"</ce
nter>");
	document.getElementById("f1").style.backgroundColor="#FFFFFF";
	document.getElementById("funcDiv1").style.visibility="hidden";
	document.getElementById("funcDiv2").style.visibility="visible";
}
function setFunc2()
{
	document.getElementById("f2").innerHTML=("<center>"+setObjPlan.fNameI2.value+"</ce
nter>");
	document.getElementById("f2").style.backgroundColor="#FFFFFF";
```

```
            document.getElementById("FuncDiv2").style.visibility="hidden";
            document.getElementById("codeDiv").style.visibility="visible";
            document.getElementById("empfunc").style.backgroundColor="#D6D6D6";
}

</script>
</head>

<body>
<div id="codeDiv">
  <iframe name="code" width="470px" height="360px" scrolling="auto"></iframe>
</div>
<form name="setObjPlan" action="http://www.wpol.org/cgi-bin/objPlan.cgi" method="post"
target="code">
<div id="funcDiv2">
  <p><strong>Function Plan Properties:</strong></p>
  <p>Name:
    <input type="text" name="fNameI2" id="fNameI2">
  </p>
  <p>Return Type:
    <select name="fRType2" id="fRType2">
       <option value="void" selected>void</option>
       <option value="int">integer</option>
       <option value="double">double</option>
       <option value="string">string</option>
       <option value="char">character</option>
     </select>
  </p>
  <p>Accessibility:
    <select name="fAcc2" id="fAcc2">
       <option value="public">public</option>
       <option value="private">private</option>
       <option value="protected">protected</option>
     </select>
  </p>
  <p>Parameters (Optional):</p>
  <p>
    <select name="fParamT12" id="fParamT12">
      <option value="int">integer</option>
      <option value="double">double</option>
      <option value="string">string</option>
      <option value="char">character</option>
      <option value="float">float</option>
    </select>
    <input type="text" name="fParamN12" id="fParamN12">
  </p>
  <p>
    <select name="fParamT22" id="fParamT2">
      <option value="int">integer</option>
      <option value="double">double</option>
      <option value="string">string</option>
      <option value="char">character</option>
      <option value="float">float</option>
    </select>
    <input type="text" name="fParamN22" id="fParamN2">
  </p>
```

```
  <p>
   <select name="fParamT32" id="fParamT3">
    <option value="int">integer</option>
    <option value="double">double</option>
    <option value="string">string</option>
    <option value="char">character</option>
    <option value="float">float</option>
   </select>
   <input type="text" name="fParamN32" id="fParamN3">
   <input type="submit" name="setFunc1B2" id="setFunc1B2" value="Set" onClick="setFunc2()">
  </p>
</div>
<div id="setV3Div">
  <p><strong>Variable Plan Properties:</strong></p>
  <p>Name:
   <input type="text" name="vNameI3" id="vNameI3">
  </p>
  <p>Type:
   <select name="vType3" id="vType3">
      <option value="int" selected>integer</option>
      <option value="string">string</option>
      <option value="char">character</option>
      <option value="double">double</option>
      <option value="float">float</option>
      <option value="Employee">Employee</option>
      <option value="Student">Student</option>
    </select>
  </p>
  <p>Accessibility:
   <select name="vAcc3" id="vAcc3">
      <option value="private" selected>private</option>
      <option value="public">public</option>
      <option value="protected">protected</option>
    </select>
   <input type="button" name="setV1B3" id="setV1B3" value="Set" onClick="setV3()">
  </p>
</div>
<div id="setV2Div">
  <p><strong>Variable Plan Properties:</strong></p>
  <p>Name:
   <input type="text" name="vNameI2" id="vNameI2">
  </p>
  <p>Type:
   <select name="vType2" id="vType2">
      <option value="int" selected>integer</option>
      <option value="string">string</option>
      <option value="char">character</option>
      <option value="double">double</option>
      <option value="float">float</option>
      <option value="Employee">Employee</option>
      <option value="Student">Student</option>
    </select>
  </p>
  <p>Accessibility:
   <select name="vAcc2" id="vAcc2">
      <option value="private" selected>private</option>
```

```
      <option value="public">public</option>
      <option value="protected">protected</option>
    </select>
    <input type="button" name="setV1B2" id="setV1B2" value="Set" onClick="setV2()">
 </p>
 </div>
<div class="style2" id="Layer1" style="position:absolute; width:1009px; height:577px; z-index:1;
visibility: visible;">
 <div class="style2" id="phase" style="position:absolute; width:115px; height:76px; z-index:1; left:
781px; background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px none #000000;
top: 0px; visibility: visible;">   <br>
   <table width="110" border="0">
     <tr>
       <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Creation</div></td>
     </tr>
     <tr>
       <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Phase 3) </div></td>
     </tr>
   </table>
 </div>
 <div class="style2" id="level" style="position:absolute; width:115px; height:76px; z-index:2; left:
895px; top: 0px; background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px none
#000000; visibility: visible;">   <br>
   <table width="110" border="0">
     <tr>
       <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">Beginner</div></td>
     </tr>
     <tr>
       <td bordercolor="#333333" bgcolor="#FFFFFF"><div align="center">(Level 1) </div></td>
     </tr>
   </table>
 </div>
 <div id="Layer6" style="position:absolute; width:98px; height:407px; z-index:3; top: 76px;
background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px none #000000;
visibility: visible; left: 2px;">
 </div>
 <div id="properties" style="position:absolute; width:912px; height:106px; z-index:6; left: 98px; top:
483px; background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px none #000000;
visibility: visible;">
   <table width="730" height="101" border="0">
     <tr>
       <td width="126"><strong>Plan Properties:</strong></td>
       <td width="129"> </td>
       <td width="94"> </td>
       <td width="116"> </td>
       <td width="116"> </td>
       <td width="123"> </td>
     </tr>
     <tr>
       <td><strong>Name:</strong></td>
       <td id="pName" bgcolor="#FFFFFF"> </td>
       <td><strong>Sub-plans:</strong></td>
       <td bgcolor="#FFFFFF">Data Member</td>
       <td bgcolor="#EBEBEB"> </td>
       <td> </td>
     </tr>
```

```
      <tr>
        <td><strong>Type:</strong></td>
        <td bgcolor="#FFFFFF">Class</td>
        <td> </td>
        <td bgcolor="#FFFFFF">Member Function</td>
        <td> </td>
        <td> </td>
      </tr>
      <tr>
        <td> </td>
        <td> </td>
        <td> </td>
        <td bgcolor="#FFFFFF">Object Utilities</td>
        <td> </td>
        <td> </td>
      </tr>
    </table>
</div>
</div>
<div class="style2" id="Layer2" style="position:absolute; width:98px; height:76px; z-index:5;
background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 0px none #000000;
visibility: visible;">
  <div align="center"><br>
    <span class="style5">WPOL</span></div>
</div>
<div class="style2" id="Layer3" style="position:absolute; width:683px; height:76px; z-index:6; left:
108px; top: 15px; background-color: #EBEBEB; layer-background-color: #EBEBEB; border: 1px
none #000000; visibility: visible;">
<div align="center"><br>
      <span class="style5">Object Plan</span></div>
</div>
<div id="apDiv1"></div>
<div id="apDiv2"></div>
<div id="objDiv">
  <table width="228" height="285" border="0" cellpadding="0" cellspacing="0">
    <tr>
      <td bgcolor="#CCFFFF" id="objName"><img src="pics/obj2.png" width="26" height="26">
Object Plan</td>
    </tr>
    <tr>
      <td bgcolor="#D6D6D6" id="empdata"><img src="pics/data.png" alt="" width="24"
height="24"><span class="style13">Data Member Plan</span></td>
    </tr>
    <tr>
      <td id="v1" bgcolor="#D6D6D6"> </td>
    </tr>
    <tr>
      <td id="v2" bgcolor="#D6D6D6"> </td>
    </tr>
    <tr>
      <td id="v3" bgcolor="#D6D6D6"> </td>
    </tr>
    <tr>
      <td bgcolor="#D6D6D6" id="empfunc"><img src="pics/func.png" width="29"
height="26"><span class="style13">Member Function Plan</span></td>
    </tr>
```

```
    <tr>
      <td id="f1" bgcolor="#D6D6D6"> </td>
    </tr>
    <tr>
      <td id="f2" bgcolor="#D6D6D6"> </td>
    </tr>
    <tr>
      <td bgcolor="#D6D6D6" id="emput"><img src="pics/tools.png" width="24" height="24"><span
class="style13">Object Utilities</span></td>
    </tr>
  </table>
</div>
<div class="style15" id="apDiv15">
  <div align="center"><strong>Object Plan - Creation Phase</strong> Create your own Object by
adding sub-plans and setting plan properties. </div>
</div>
<div class="style13" id="setObjDiv">
  <p><strong>Object Plan Properties:</strong></p>
  <p>Name:
    <input type="text" name="setNameI" id="setNameI">
    <input type="button" name="setNameB" id="setNameB" value="Set" onClick="setName()">
  </p>
</div>
<div id="setV1Div">
  <p><strong>Variable Plan Properties:</strong></p>
  <p>Name:
    <input type="text" name="vNameI1" id="vNameI1">
  </p>
  <p>Type:
    <select name="vType1" id="vType1">
      <option value="int" selected>integer</option>
      <option value="string">string</option>
      <option value="char">character</option>
      <option value="double">double</option>
      <option value="float">float</option>
      <option value="Employee">Employee</option>
      <option value="Student">Student</option>
    </select>
  </p>
  <p>Accessibility:
    <select name="vAcc1" id="vAcc1">
      <option value="private" selected>private</option>
      <option value="public">public</option>
      <option value="protected">protected</option>
    </select>
    <input type="button" name="setV1B" id="setV1B" value="Set" onClick="setV1()">
  </p>
</div>
<div id="funcDiv1">
  <p><strong>Function Plan Properties:</strong></p>
  <p>Name:
    <input type="text" name="fNameI1" id="fNameI1">
  </p>
  <p>Return Type:
    <select name="fRType1" id="fRType1">
      <option value="void" selected>void</option>
```

```
    <option value="int">integer</option>
    <option value="double">double</option>
    <option value="string">string</option>
    <option value="char">character</option>
   </select>
  </p>
  <p>Accessibility:
   <select name="fAcc1" id="fAcc1">
    <option value="public">public</option>
    <option value="private">private</option>
    <option value="protected">protected</option>
   </select>
  </p>
  <p>Parameters (Optional):</p>
  <p>
   <select name="fParamT11" id="fParamT11">
    <option value="int">integer</option>
    <option value="double">double</option>
    <option value="string">string</option>
    <option value="char">character</option>
    <option value="float">float</option>
   </select>
   <input type="text" name="fParamN11" id="fParamN11">
  </p>
  <p>
   <select name="fParamT21" id="fParamT21">
    <option value="int">integer</option>
    <option value="double">double</option>
    <option value="string">string</option>
    <option value="char">character</option>
    <option value="float">float</option>
   </select>
   <input type="text" name="fParamN21" id="fParamN21">
  </p>
  <p>
   <select name="fParamT31" id="fParamT31">
    <option value="int">integer</option>
    <option value="double">double</option>
    <option value="string">string</option>
    <option value="char">character</option>
    <option value="float">float</option>
   </select>
   <input type="text" name="fParamN31" id="fParamN31">
   <input type="button" name="setFunc1B" id="setFunc1B" value="Set" onClick="setFunc1()">
  </p>
 </div>
 </form>
 </body>
 </html>
```

Bibliography

Software Errors Cost U.S. Economy $59.5 Billion Annually. June 28, 2002. National Institute of Standards and Technology (NIST). April 21, 2008. <http://www.nist.gov/public_affairs/releases/n02-10.htm>

ARNOW, D., & BARSHAY, O. WebToTeach: A Web-based Automated Program Checker. Frontiers in Education (FIE '99), San Juan, Puerto Rico (Nov. 1999).

BRUCE, K. B. 2005. Controversy on how to teach CS 1: a discussion on the SIGCSE-members mailing list. *SIGCSE Bull.* 37, 2 (Jun. 2005), 111-117.

DEWAR, R. AND SCHONBERG, E. "Computer Science Education: Where Are the Software Engineers of Tomorrow?" Cross Talk: The Journal of Defense Software Engineering, Jan 2008

EBRAHIMI, A. Novice Programmer Errors: Language Constructs and Plan Composition. In *International Journal of Human-Computer Studies Volume 41, Issue 4* (Oct. 1994), 457-480.

EBRAHIMI, A. VPCL: A Visual Language for Teaching and Learning Programming (A Picture is Worth a Thousand Words) *Journal of Visual Languages and Computing*, 3 (1992), 299-317.

EBRAHIMI, A., & SCHWEIKERT, C. "Empirical Study of Novice Programming with Plans and Objects", ACM Inroads 38, 4 (Dec. 2006), 52-54.

CONWAY, M, PIERCE, J, PAUSCH, R, *et al.* Alice: Lessons Learned from Building a 3D System for Novices. *Proceedings of CHI 2000,* (2000), 486-493.

GLASS, R. “Silver Bullet” Milestones in Software History *Communications of the ACM, 48(8),* (2005) 15-18.

HRISTOVA, M, MISRA, A, RUTTER, M, MERCURI, R. Identifying and Correcting Java Programming Errors for Introductory Computer Science Students. *Proc. the 34th SIGCSE Technical Symposium on Computer Science Education,* Reno, Nevada, USA, ACM Press, 34 (2003), 153-156.

HU, C. 2008. Just say 'A Class Defines a Data Type'. *Communications of the ACM* 51, 3 (Mar. 2008), 19-21.

HU, C. “Dataless Objects Considered Harmful” *Communications of the ACM*, 48(2), (2005), 99-101

JENKINS, S. “Musings of an ‘Old-School’ Programmer” *Communications of the ACM*, 49(5), (2006), 124-126.

KNUTH, D. The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, 1997.

KO, A. J., MYERS, B. A. Development and Evaluation of a Model of Programming Errors, *IEEE Symposia on Human-Centric Computing Languages and Environments*, Auckland, New Zealand, (2003), 7-14.

KOPEC, D, CLOSE, D, AMAN, J. "Teaching in shifting sands: changes in CS2", Journal of Computing Sciences in Colleges 17(3), (2002).

KOPEC, D, CLOSE, D, AMAN, J. "CS1: Perspectives on Programming Languages and the Breadth-First Approach" in Journal of Computing in Small Colleges, Proceedings of The Fifth Annual Consortium for Computing in Small colleges: Northeastern Conference, Ramapo College, NJ., (Apr. 2000), 228-234.

LIFFICK, B, AIKEN, R. A novice programmer's support environment. Proceedings of the 1st conference on Integrating technology into computer science education, 28, 24 Issue SI (1996), 1-3.

KAASBØLL, J., BERGE, O., BORGE, R. E., FJUK, A., HOLMBOE, C. AND SAMUELSEN, T. (2004). Learning Object-Oriented Programming. In Dunican, E and Green, T. (Eds.): *Proceedings of the 16th Annual Workshop of the Psychology of Programming Interest Group*: 86-96.

MCCRACKEN, M., ALMSTRUM, V., DIAZ, D. et al, A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-Year CS Students. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, (December 01, 2001), Canterbury, UK.

POWERS, K., ECOTT, S., and HIRSHFIELD, L. M. 2007. Through the looking glass: teaching CS0 with Alice. *SIGCSE Bull.* 39, 1 (Mar. 2007), 213-217.

REED, D. Rethinking CS0 with JavaScript. SIGCSE Bulletin 33(1), (2001), 100-104.

JOHNSON, W. Intention-Based Diagnosis of Novice Programming Errors, Morgan Kaufman Publishers, Inc., Los Altos, California (1986).

SCHANK, R, ABELSON, R. Scripts, plans and knowledge. Proceedings of the fourth international joint conference on artificial intelligence, Tbilisi, (1975) Re-published in P. Johnson-Laird and P. Wason, (Eds.), Thinking. Cambridge, England: Cambridge University Press, (1977).

SOLOWAY, E. Learning to program = Learning to construct mechanisms and explanations. *Communications of the ACM,* 29(9), (1986), 850-858.

SOLOWAY, E, EHRLICH, K, BONAR, J. Tapping into Tacit Programming Knowledge. Proceedings of the Conference on Human Factors in Computing Systems, NBS, Gaithersburg, Md, (1982).

SPOHRER, J. C., SOLOWAY, E. Alternatives to construct-based programming misconceptions. ACM SIGCHI Bulletin , Proceedings of the SIGCHI conference on Human factors in computing systems, 17, 4, (Apr. 1986).

SPOHRER, J. C., SOLOWAY, E, POPE, E. A goal-plan analysis of buggy Pascal programs. Human-Computer Interaction, 1, (1985), 163-207.

THOMASSON, B., RATCLIFFE, M., THOMAS, L. Identifying novice difficulties in object oriented design. In *Proceedings of the 11th Annual SIGCSE Conference on innovation and Technology in Computer Science Education* (Bologna, Italy, June 26 - 28, 2006). ITICSE '06. ACM, New York, NY, (2006), 28-32.

TRUONG, N, BANCROFT, P, ROE, P. Learning to program through the web. Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education ITiCSE '05, (2005).

WEI, F., MORITZ, S., PARVEZ, S., AND BLANK, G. D., A Student Model for Object-Oriented Design and Programming*, Consortium for Computing Sciences in Colleges* (*CCSCNE*), Providence, RI, (April 2005).

YU, C. C., ROBERTSON, S. P. Plan-based representations of Pascal and Fortran code. Proceedings of the SIGCHI conference on Human factors in computing systems, (May 1988).