Software Requirements Specification
Client: Dr. Darren Lim, Assistant Professor

Proposed Project: Java Online Learning Toolkit (J.O.L.T.)

Delivered by: 518 Interactive

Team Members:
Lawrence Gregory
Christopher Hughto
Erik Stegmann
Connor Vander Bogart
Jedidiah Turnbull

Revision: 1.0
Date: 10/28/09

Contents

1 Introduction

1.1 Purpose

This document defines the software requirements identified for the *Java Online Learning Toolkit* (JOLT) for the client, Dr. Darren Lim, Assistant Professor of the Computer Science Department of Siena College. These requirements are subject to review by *518 Interactive* and Dr. Lim.

This document serves these main purposes:

- To ensure that both *518 Interactive* and Dr. Lim understand and agree to each requirement; and,
- To serve as a basis for the design phase as well as *518 Interactive* Quality Assurance (QA) testing.

There are screenshot prototypes contained within this document that are provided to aid the developers in working towards design documentation and to provide Dr. Lim with a general idea of how the functionality requested will be implemented. This information can only be considered as examples and are supporting materials. The implementation of such items may slightly differ from the specifications.

All references to *Source Code* imply Java™ Source Code, made to work with Java™ Version 1.6

1.2 Scope

The requirements in this document comprise the scope of JOLT's requirements. This Software Requirements Specification (SRS) document supersedes any prior documents as the descriptor of JOLT's requirements.

1.3 Audience

This document is intended for *518 Interactive*, Dr. Lim, members of the Fall 2009 Software Engineering I class, and the other clients, Dr. Lederman and Dr. Breimer.

2 JOLT Requirements Specification

2.1 Product Overview and Summary

The proposed *Java Online Learning Toolkit* (JOLT) is a comprehensive web application designed to enhance the experience of learning the Java programming language. JOLT shall allow for students enrolled in programming courses to solve programming problems, which are created and entered by the Computer Science faculty. JOLT shall provide a personal gradebook for all students, as well as a course gradebook for Computer Science faculty members which will be used to track progress.

2.2 Development, Operating, and Maintenance Environments

JOLT will be developed in Siena College's Software Engineering Lab, located in Roger Bacon, third floor. The members of *518 Interactive* will be using the following resources:

Windows Machine:

- **Operating System**: Microsoft Windows Vista Enterprise
  - Build: 6002
  - Revision: 18005
  - Service Pack 2
- **Processor**: Intel® Core™2 Duo CPU
- **Model**: E7500
- **Speed** 2.93 GHz
- **Memory (RAM):** 4.00 GB
- **System Type**: 32-bit
- **Dual Monitor Setup**
- **Software Installed**:
  - Microsoft Office 2007 (Including Microsoft Project)
  - Macromedia Dreamweaver, Fireworks, Flash , Freehand, Studio (2004 Versions)
  - Internet Explorer, Mozilla Firefox, Google Chrome

Macintosh Machine:

- **Operating System**: Apple Mac OS X
  - Version 10.4.11
  - Model: iMac5
- **Processor**: Intel Core2 Duo
  - Speed: 2 GHz
- **Memory (RAM):** 1.00 GB
- **Dual Monitor Setup**
- **Software Installed**:
  - Microsoft Office 2004 for Mac
  - Macromedia Dreamweaver, Fireworks, Flash, Freehand, Studio (2004 Versions)
  - Safari, Mozilla Firefox

JOLT will be implemented, and designed to run on the following specifications:

- **Operating System**: CentOS (Linux) Release 5.2 (Final)
- **Server Name**: oraserv.cs.siena.edu
- **CPU Type**: x86_64
- **Web Server**: Apache Version 2.2.9
- **PHP Version**: 5.2.6
- **Database**: MySQL Version 5.0.45; Oracle Version 9i
- **Java™ Version**: 1.6.0_10-rc
- **Java™ SE Runtime Environment**: Build 1.6.0_10-rc-b28
- **Java HotSpot™ 64-Bit Server VM**: Build 11.0-b15, mixed mode)

Users of JOLT will be able to access the web application through an Internet connection, with any of the following browsers (of the latest version):

- Microsoft Internet Explorer
- Mozilla Firefox
- Apple Safari
- Google Chrome
- Opera Software's Opera Browser

The operating and maintenance environments have not been determined at this time.

2.3 User Case Narratives

User Case Narratives describe the interaction between an actor (or category of users) to achieve an observable goal. The following generalizes, in prose, what each user shall be able to accomplish through interacting with JOLT:

2.3.1 Student User Case Narrative

Students shall have the ability to register an account with the System. Once registered, students will be able to log into the system via their unique username and password. Once logged in, students will be able to enroll in only the courses they are currently taking. Students will have the ability to view problems in a categorized manner. Students will also be able to take exams and solve individual problems created by their instructor. Students will be able to solve problems by submitting Java source code, which the system will compile and run against provided test data. While solving a set of problems, students will be able to navigate from problem to problem without completing them in a specific order. Students will be able to save their progress for any individual problem and work on it again during a later session.

Students will have a report card view which will allow them to view their own grades and progress in all current and past courses. Students will be able to browse all of their own solutions as well.

2.3.2 Faculty User Case Narrative

Faculty shall have the ability to log into the system via a unique username and password. Once logged in, Faculty will be able to select a course and then create individual problems as well as problem sets for that course. Each problem will be categorized based on type and difficulty. Faculty will be able to assign a grading scheme to problems and problem sets. Each problem created will have the ability to be modified; however, all changes will be recorded into the System as a new problem. This will allow users to view problems and problem sets in a temporal manner.

Faculty will have the ability to submit their problems and problem sets to a “Course Pool”, which will allow other faculty members who teach the same course access to their problems and problem sets. Faculty will have the ability to search a Course Pool or the Global Pool.

Faculty will have a grade book view which will allow them to see the progress of each student that they instruct or have previously instructed. Faculty will have the ability to alter any grade that was assigned to a student in their course. Faculty will have the ability to send a broadcast message to students that they instruct. Faculty will have the ability to go into a Student mode which will modify their access to that of a Student.

2.3.3 Course Coordinator User Case Narrative

The Course Coordinator shall have the ability to log into the System via a unique username and password. Once logged in, the Course Coordinator will be able to create Faculty accounts. The Course Coordinator will also be able to create courses and assign the courses to specific Faculty members. The Course Coordinator will have access to course-wide reporting tools, which will allow for statistical analysis of problems and problem sets.

The Course Coordinator will be able to manage the "Course Pool" for each course they manage. The Course Coordinator will be responsible for adding, modifying, and deleting problems and problem sets from the pool. The Course Coordinator submits problems and problem sets to the "Global Pool" for use by all faculty members.

The Course Coordinator will be able to send broadcast messages to faculty members and students that participate in the courses that the Course Coordinator manages, or a subset thereof.

2.3.4 Administrator User Case Narrative

The Administrator shall be able to log into the System via a unique username and password. Once logged in, the Administrator will be able to create and manage Course Coordinator, Faculty, and Student accounts. The Administrator has the same abilities as a Course Coordinator. The Administrator will be able to send broadcast messages to all users, or a subset thereof. The Administrator will manage the "Global Pool" of problems and problem sets.

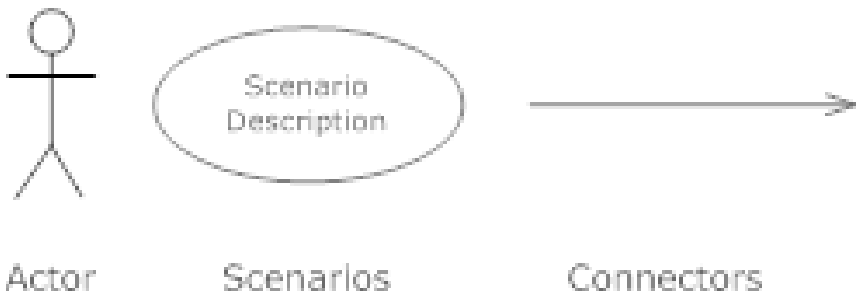
2.4 UML Use Case Diagram

A UML Use Case Diagram is commonly utilized in Software Engineering to provide a graphical representation of the basic, high level functionality a system will support. The main purpose of a UML Use Case Diagram is to clearly communicate what the most important, high level functions are which are performed by the system as well as which actors are involved with each of these important functions. Use Case Diagrams contain two components: Actors and Scenarios.

Actors are entities that interact with the System. Actors can be either human or non-human. Actors are drawn outside of the System Boundary. Human Actors are drawn to the left of the system boundary. Non-Human Actors are drawn to the right of the system boundary. Actors are represented as stick figures.

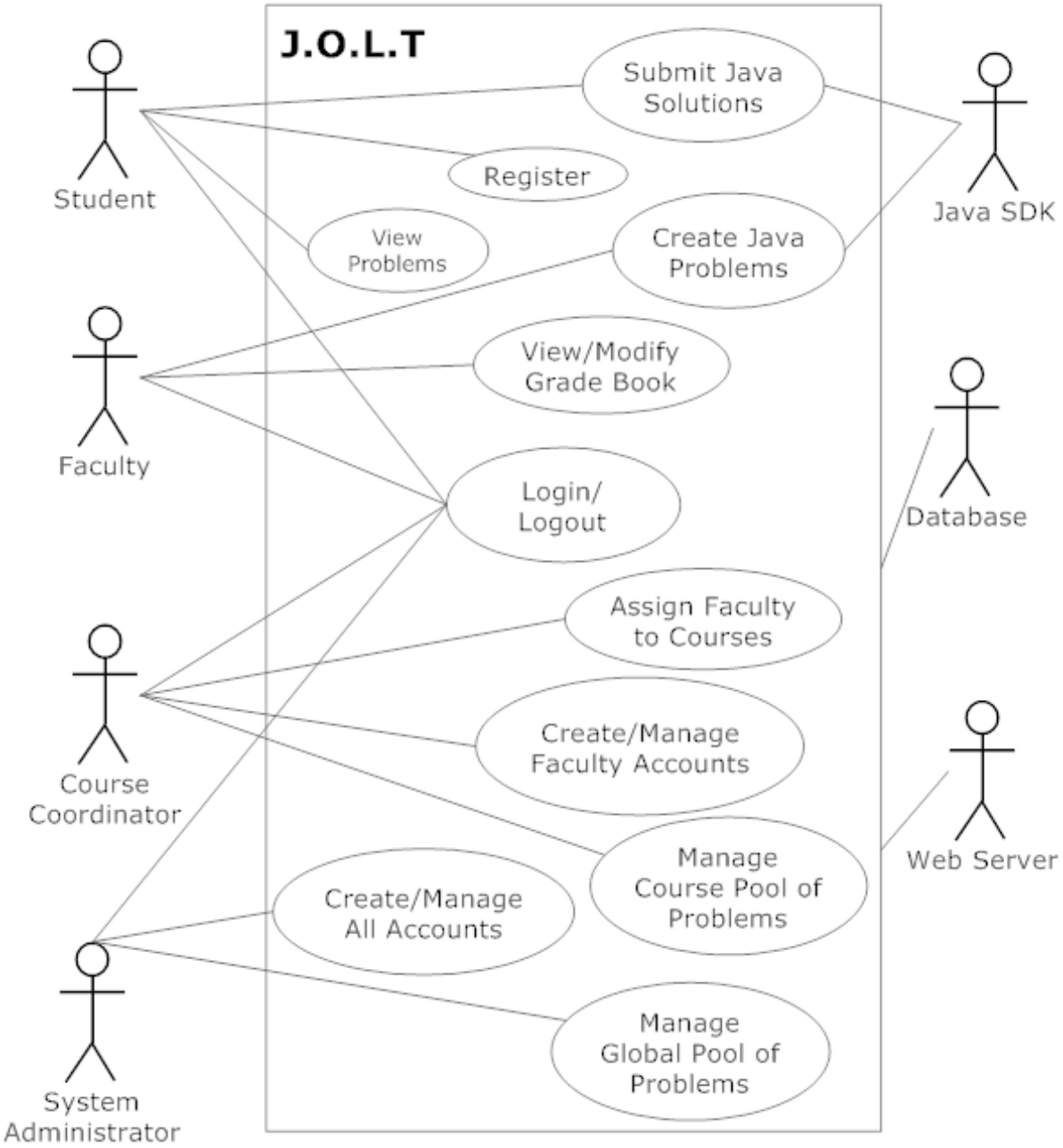
A Scenario is a specific use case, represented as an oval. Scenarios are contained inside of the System Boundary. An interaction between an Actor and a Scenario is represented with a solid line connecting the two components.

2.4.1 UML Use Case Diagram Notation



2.4.2 UML Use Case Diagram for JOLT

The Use Case Diagram below is for JOLT. Notice the lines connecting the Database and the Web Server to the System are drawn to the System Boundary, and not to Scenarios. This represents the fact that these actors interact with every Scenario within the System. This is non-standard notation, but for readability, it was included.



2.5 Data Flow Diagrams

The Data Flow Diagrams (DFDs) are used for structure analysis and design. DFDs show the flow of data from external entities into the system. DFDs also show how the data moves from one process to another, as well as its logical storage. The following symbols are used within DFDs. For clarity, a key has been provided at the bottom of this page.

Source/Sink: Represented by rectangles in the diagram. Sources and Sinks are external entities which are sources or destinations of data, respectively.

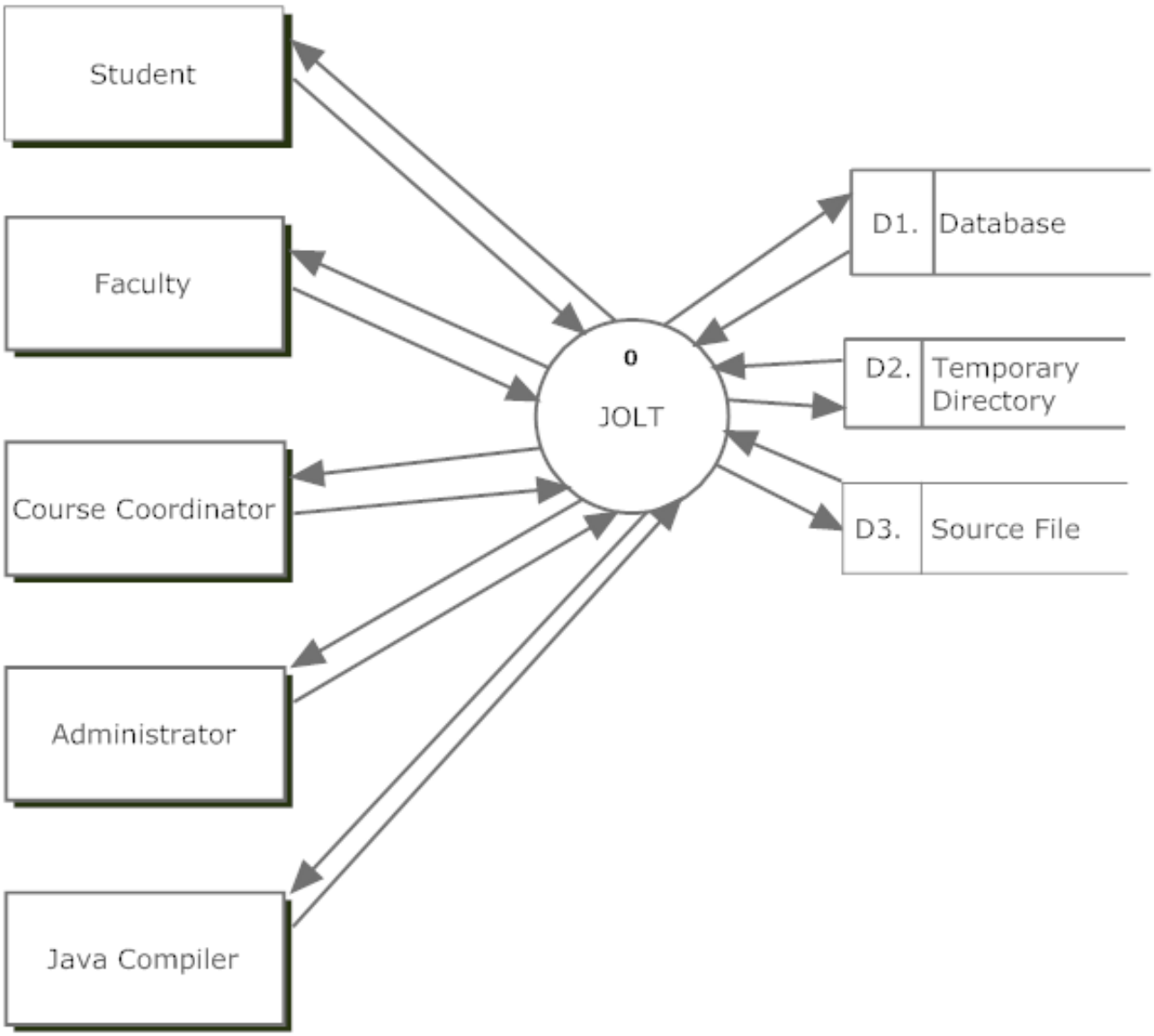
Process: Represented by circles in the diagram. Processes are responsible for manipulating the data. They take data as input and output an altered version of the data.

Data Store: Represented by a segmented rectangle with an open end on the right. Data Stores are both electronic and physical locations of data. Examples include databases, directories, files, and even filing cabinets and stacks of paper.

Data Flow: Represented by a unidirectional arrow. Data Flows show how data is moved through the System. Data Flows are labeled with a description of the data that is being passed through it.

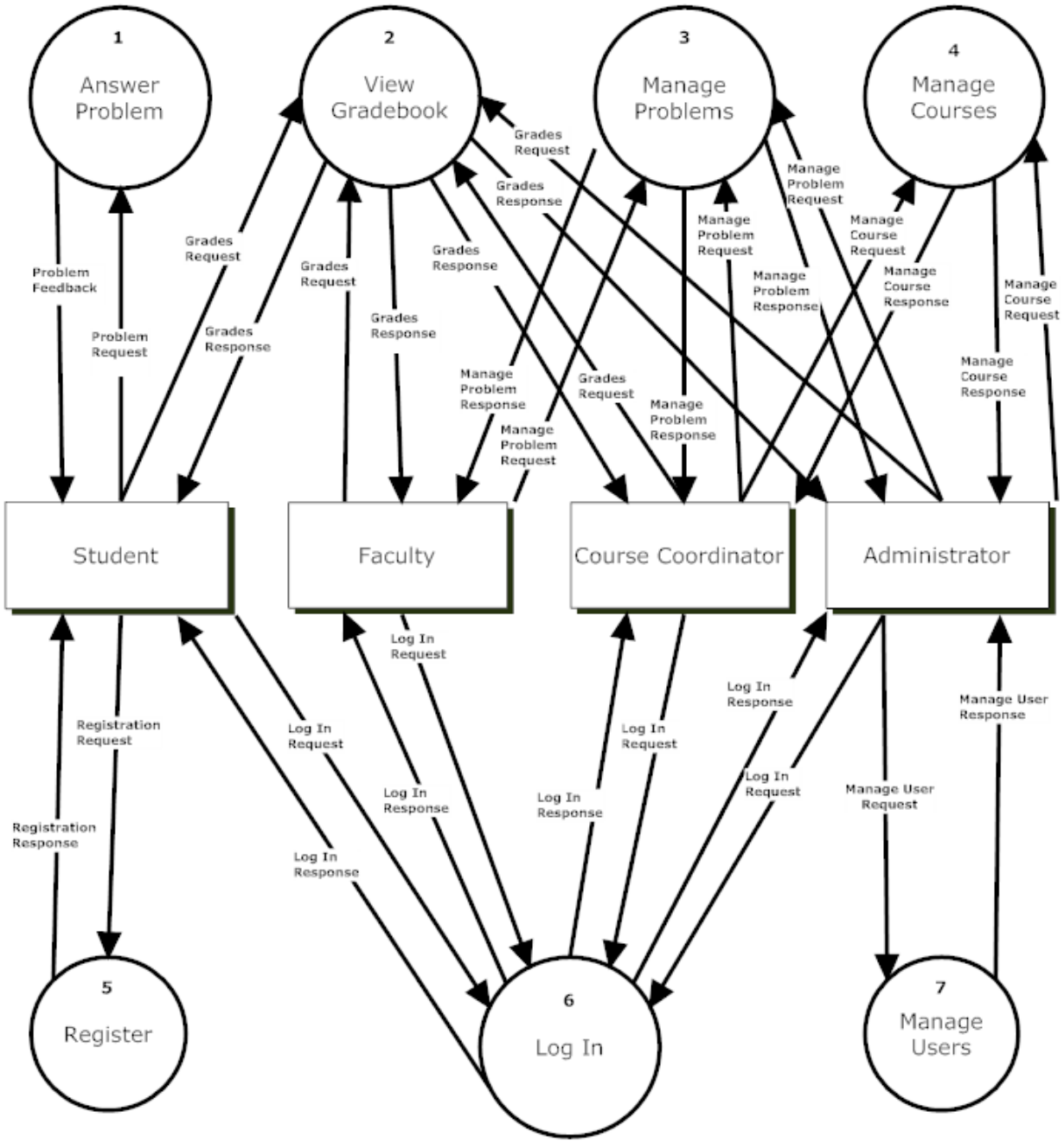
2.5.1 Data Flow Diagram: Context

A context diagram is a top level data flow diagram. It only contains one process node (Process 0) that generalizes the function of the entire system in relationship to external entities.



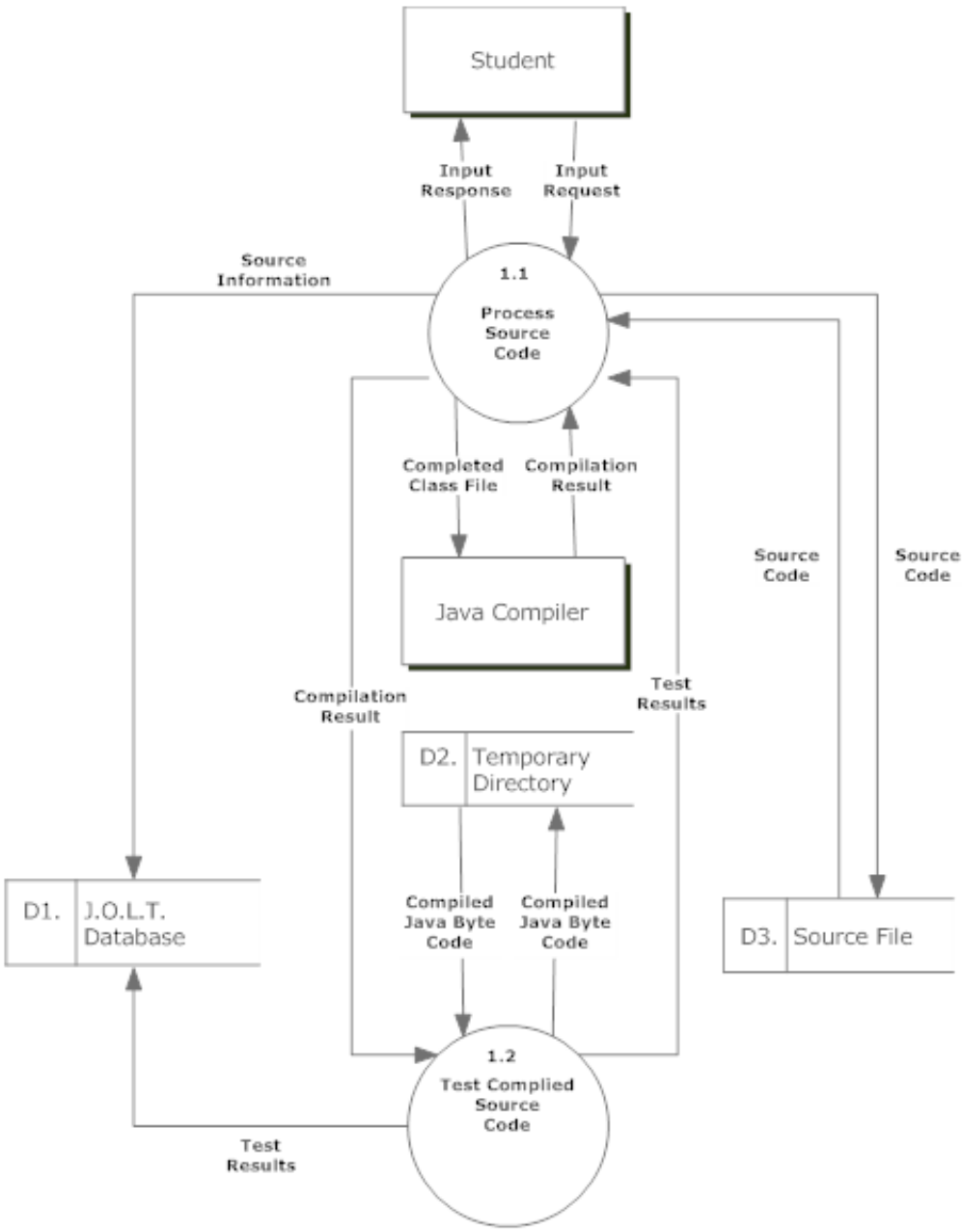
2.5.2 Data Flow Diagram: Level 0

The Level 0 Data Flow Diagram shows all of the main, high-level functions of JOLT. Note that each process uses one or more data stores. For clarity, the data stores have been omitted from this diagram. Further detail is provided in subsequent diagrams.



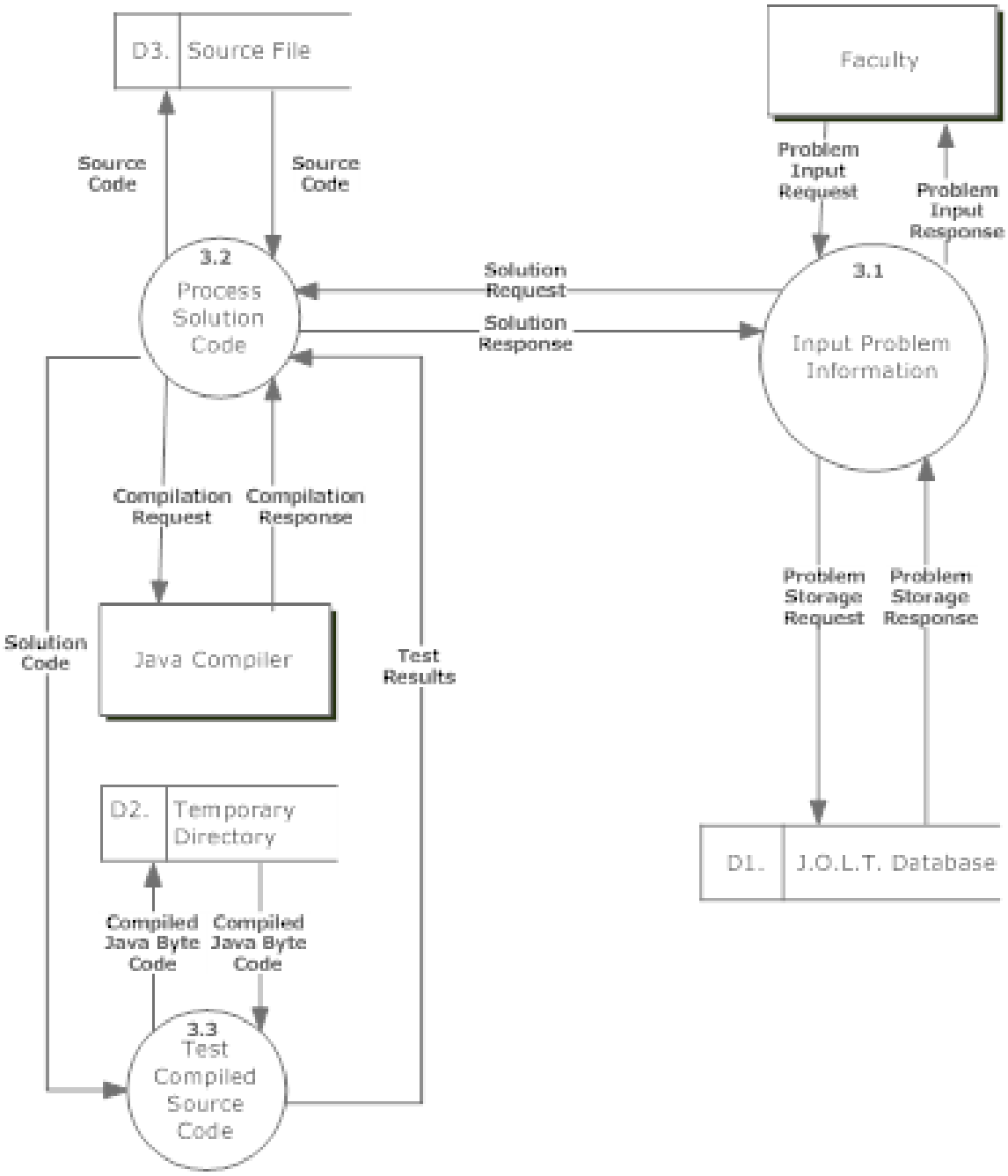
2.5.3 Data Flow Diagram: Level 1.1

The Level 1.1 Data Flow Diagram depicts, in further detail, what the “Answer Problem” process in the Level 0 Diagram does.



2.5.4 Data Flow Diagram: Level 3.1

The Level 3.1 Data Flow Diagram depicts, in further detail, what one of the processes within the "Manage Problems" process in the Level 0 Diagram does.



2.6 Activity Diagrams

Activity Diagrams are a UML (Unified Modeling Language) specified diagram which shows workflows of stepwise activities and actions, with support for choice, iteration, and concurrency. It outlines the process that Actors (both human and non-human) go through while interacting with the System to accomplish a specific task. The following constructs are used to build Activity Diagrams. A key is provided at the bottom of this page for clarity.

Activity: Activity Building Blocks are the processes that the System and/or Actor goes through to accomplish an activity. Activity Building Blocks are represented as rectangles within the diagram. The rectangles have descriptive text within, which outlines what gets accomplished at each step.

Time Event: A Time Event represents a wait period. It is a "pause" in the activity for a specified amount of time.

Flow: The Flow is depicted as a unidirectional arrow. The Flow designates the direction and order that the activity takes place in.

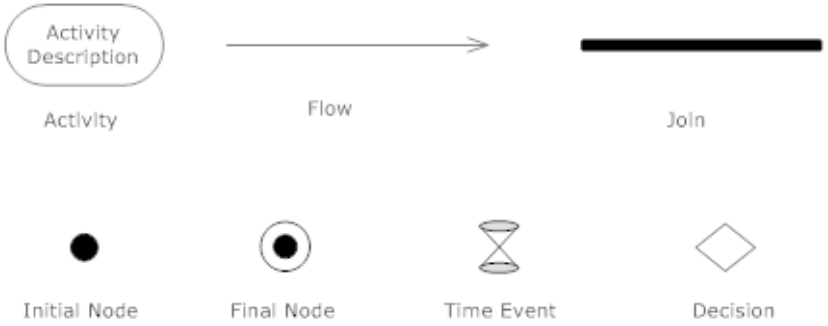
Join: The Join is depicted as a thick, solid line. The Join is used purely for aesthetics. It improves the readability of the diagram by associating multiple flows to a particular object, such as a Final Node or Decision.

Initial Node: The Initial Node is represented as a solid circle. The Initial Node defines the entry point of the Activity. All Activities always start at the Initial Node.

Final Node: The Final Node is represented as a solid circle encased in another circle. The Final Node defines the exit point of the Activity. All Activities end at the Final Node. There may be multiple ways of reaching the Final Node within each activity.

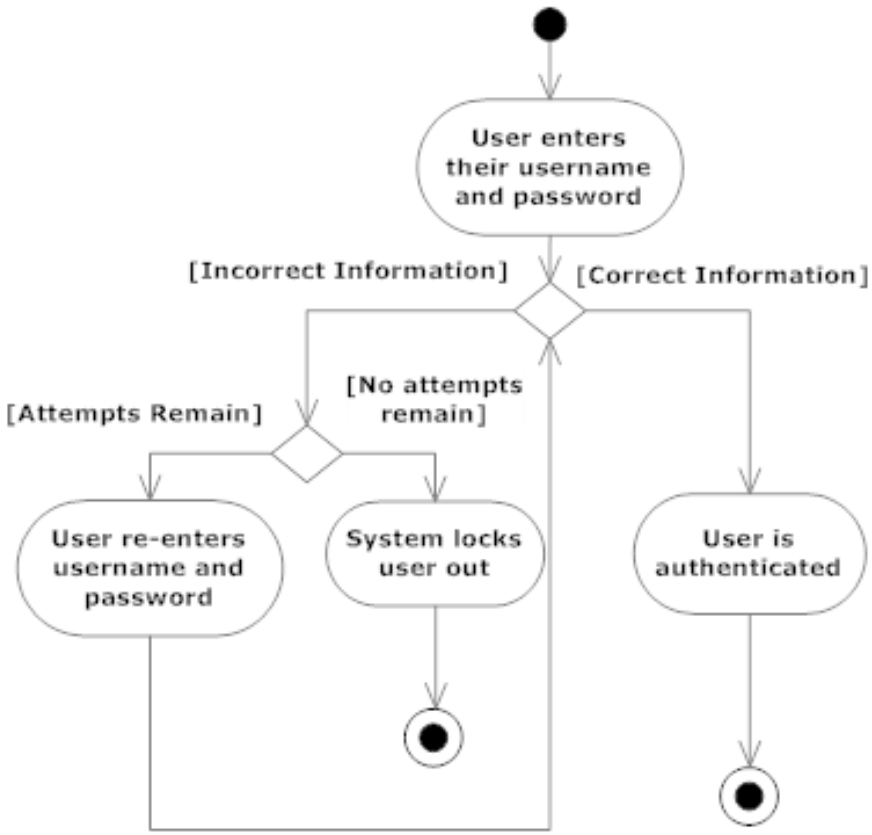
Decisions: Decisions are represented as a diamond within the diagram. Decisions are typically conditional constructs, where different computations or activities are performed depending on the condition. Decisions have two or more Flows coming out of them, with each flow labeled to identify which to follow based on the condition.

Decisions are also used within Activity Diagrams to join two or more Flows together. Multiple Flows may join together if they all lead to the same activity.



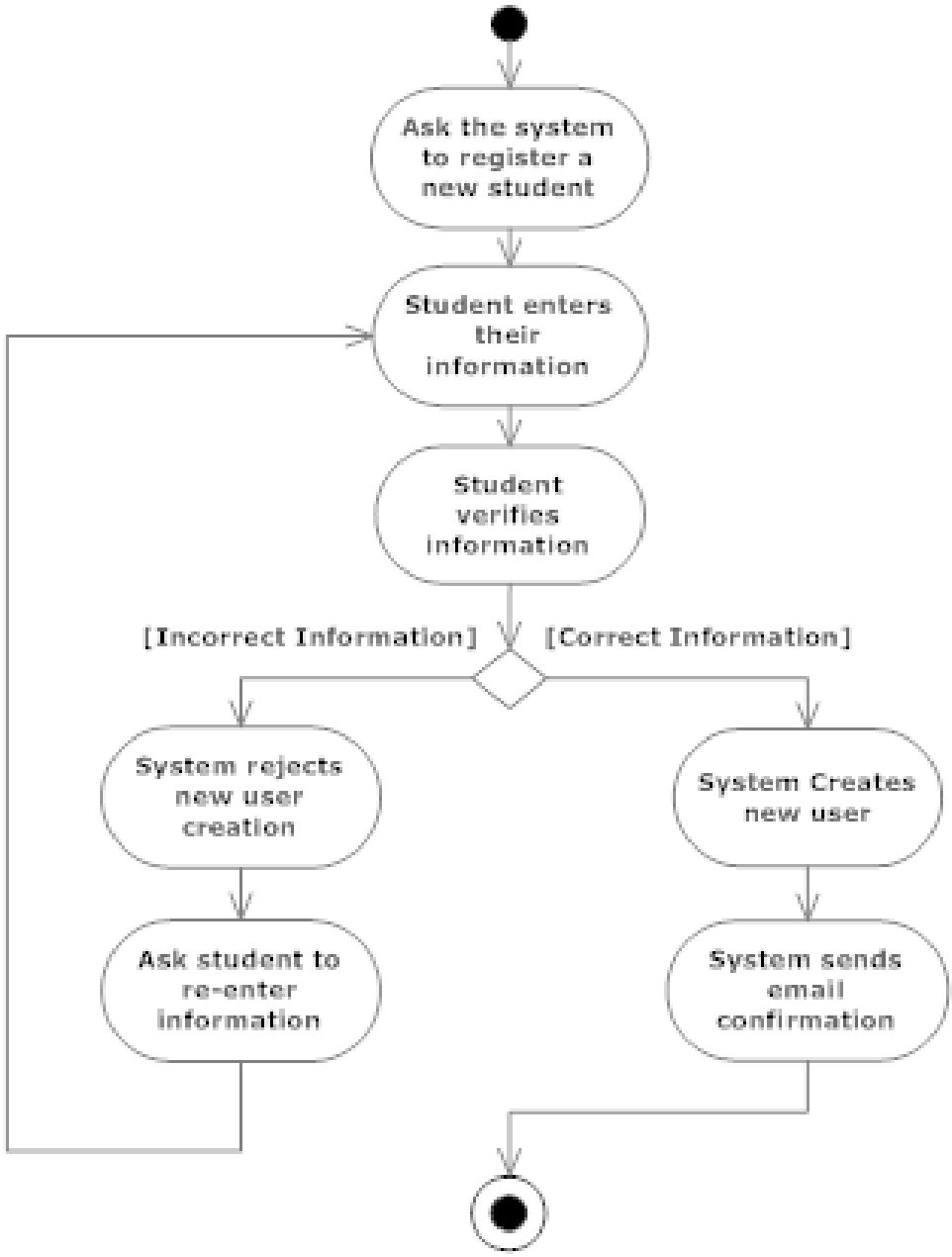
2.6.1 Activity Diagram: Authentication

This Activity Diagram depicts the authentication process that all users will go through. This diagram assumes that the user's account exists.



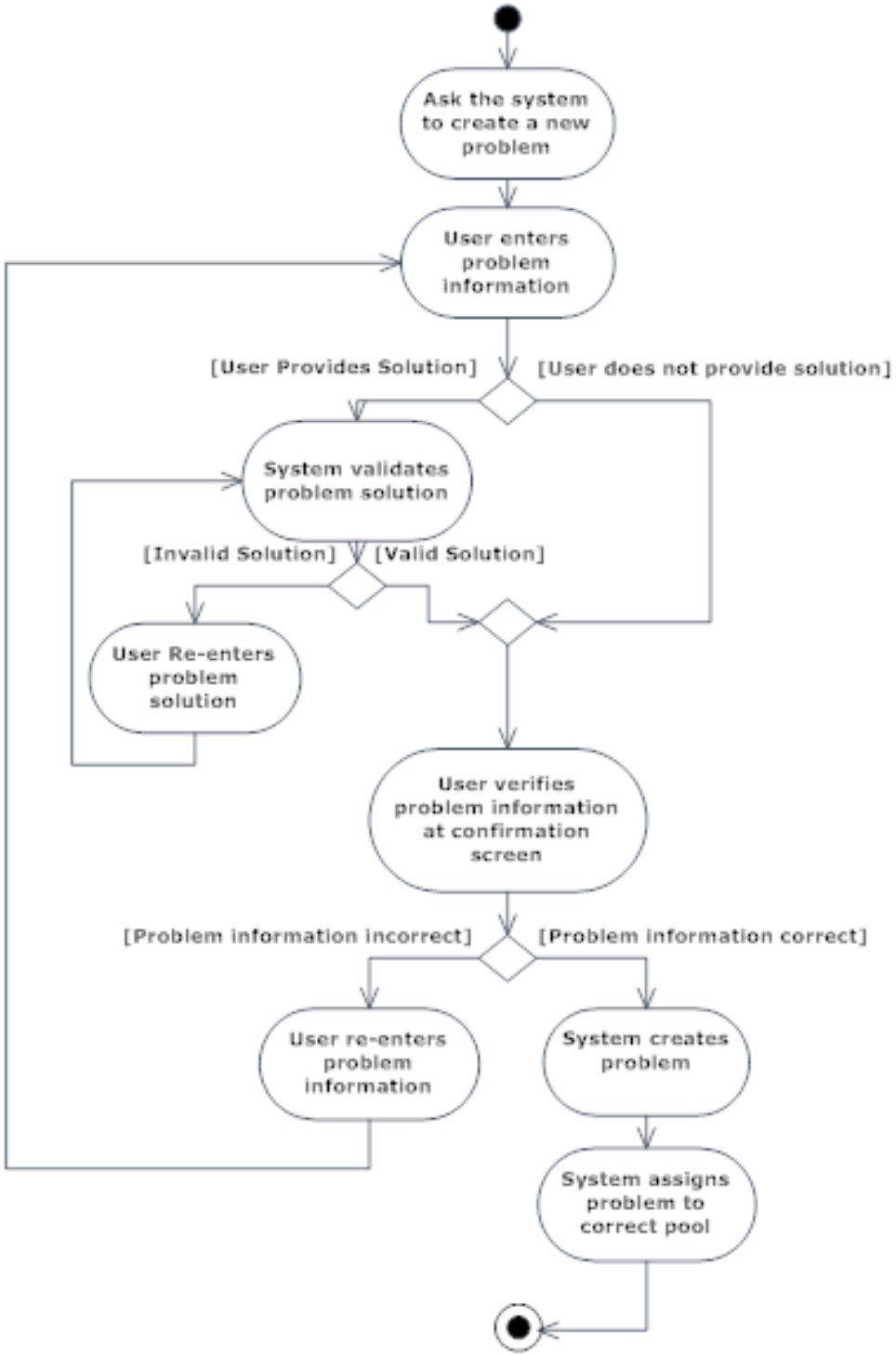
2.6.2 Activity Diagram: Student Self-Registration.

This Activity Diagram depicts the process that a Student user goes through to register an account with JOLT.



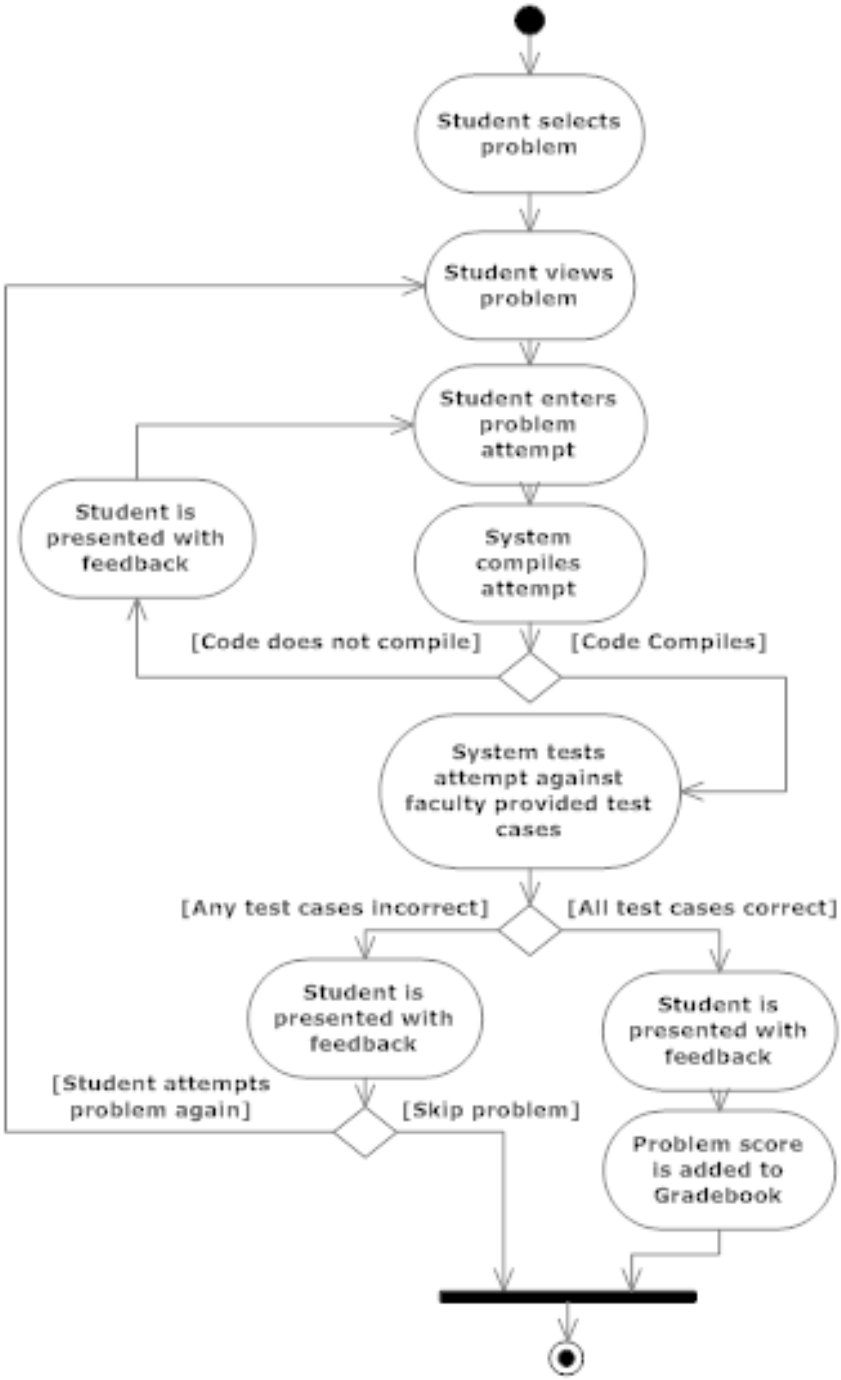
2.6.3 Activity Diagram: Faculty Problem Creation

This Activity Diagram depicts the process that Faculty members go through to create a new problem to be added to their pool. Notice this diagram utilizes both uses of the Decision: both to branch on a condition, and to join multiple processes together.



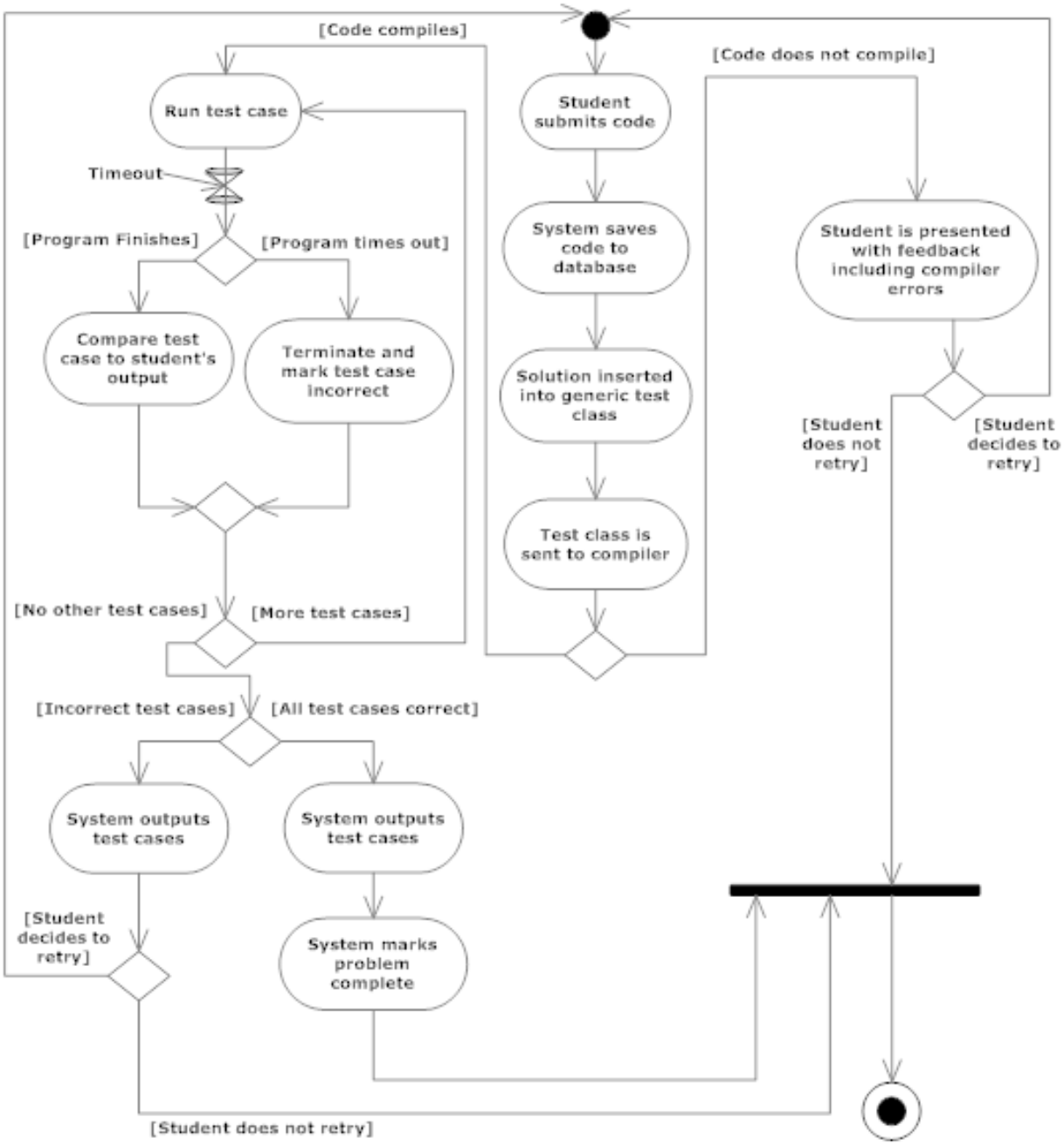
2.6.4 Activity Diagram: Student Solve Problem

This Activity Diagram depicts the process that Students go through to solve an individual problem within a Problem Set.



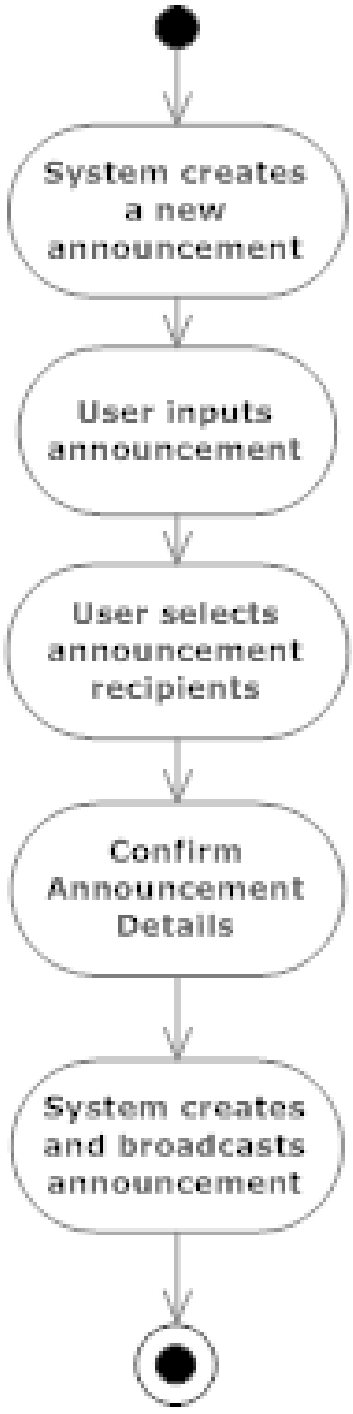
2.6.5 Activity Diagram: Testing Student Submission

This Activity Diagram is an extension of the one in Section 2.4.4. This Activity Diagram depicts the process that JOLT goes through when a student submits code to solve a problem.



2.6.6 Activity Diagram: Create Broadcast Announcement

This Activity Diagram shows the process a faculty, course coordinator, or administrator goes through to generate a broadcast announcement.



2.7 Functional Requirements Inventory

The following list outlines the required functionality to be included in the final solution.

Java Online Learning Toolkit will be a web-based application viewable on the major browsers. Browsers included will be Internet Explorer 8, Mozilla Firefox, Safari, and Google Chrome.

All references to *Source Code* imply Java™ Source Code, made to work with Java™ Version 1.6

The requirements are listed according to user case, as follows:

2.7.1 Functional Requirements: Student:

- Will be able to register online with the system
  - Will receive email confirmation following registration
- Will be able to log into system.
  - A failed log in will display an appropriate error message.
  - A link to an identity validation page will be provided if password is forgotten.
- Will be able to enroll into courses they are currently taking
  - PIN number provided by instructor required to enroll into course on the system
- Will be able to view announcements sent to them.
- Will be able to view problem sets for each course they are in enrolled in
  - Will be able to view each individual problem within the problem set
  - Will be able to view hints and solutions to individual problems, if provided by problem creator.
- Will be able to submit solutions to individual problems within active problem sets in the form of Java™ source code
  - Code will be compiled by the system online
  - Student will receive immediate, automatic feedback on compile errors, if any
  - Student will receive immediate, automatic feedback on how their solution compares to the test cases
- Will be able to complete problem sets
  - Will be able to navigate to individual problems in a problem set without completing them in a specific order
  - Will be able to save any progress made for a problem or problem set
- Will be able to view grades for each class
- Will be able to view all previously submitted solutions
- Will be able to log out

2.7.2 Functional Requirements: Faculty:

- Will be able to log into the system
- Will be able to create individual problems
- Will be able to create problem sets
    - Will be able to import previously created problems to a problem set
    - Will be able to import problems from the course pool to a problem set
    - Will be able to import problems from the global pool to a problem set
    - Will be able to individually create each problem for a problem set
- Will be able to assign problems they create to a category
- Will be able to assign a grading scheme to problem sets
    - Will be able to assign a point value to specific problems within the problem set
- Will be able to assign problem sets to the students in the courses they teach
    - Will be able to set activation date and time of problem set
    - Will be able to set expiration date and time of problem set
- Will be able to submit problems to a Course Pool
- Will be able to search a Course Pool for problems
- Will be able to search the Global Pool for problems
- Will be able to view a grade book for each of the courses
- Will be able to modify grades for all students in each of the courses they are currently teaching
- Will be able to post announcements to students in their courses
- Will be able to view announcements sent to them.
- Will be able to interact with JOLT as a student user
- Will be able to log out

2.7.3 Functional Requirements: Course Coordinator:

- Will be able to log into the system
- Will be able to create faculty accounts
- Will be able to assign faculty to a course
- Will be able to create content for courses they are in charge of
- Will have access to course tools which will provide statistics on problems and grades for a course
    - Will be able to create reports over multiple sections of a course involving all problems and problem sets or any subset thereof.
    - Reports may include general statistics such as number of participants, average score, median, low score, and high score.
- Will be able to manage the course pool for each course they are in charge of
    - Will be able to add, modify, or delete any problem in their course pool
    - Will be able to submit problems to the global pool
- Will be able to modify grades for all students currently enrolled in a course they currently manage
    - Will be able to keep track of all grades and any adjustments that are made
- Will be able to create announcement for all faculty and students of courses they manage or any subset thereof
- Will be able to log out

2.7.4 Functional Requirements: Administrator:

- Will be able log in
- Will be able to manage all accounts on the system
- Will be able to create course coordinator and faculty accounts
    - Will be able to assign courses to course coordinators
- Will have the same abilities as a course coordinator
- Will be able to manage the global pool of problems
- Will have access to tools for management of all accounts
    - Will be able to modify all account information for any user
    - Will be able to delete accounts
    - Will be able reset locked accounts
- Will be able to create announcements for all course coordinators, faculty and students, or any subset thereof
- Will be able to log out

2.7.5 Java SDK:

- Will accept and attempt to compile all Java™ source code submitted by students
    - Will output compile errors, when applicable
    - Will create Java™ Byte Code upon a successful compilation
- Will execute all successfully compiled Java™ solutions
    - Will monitor students' submissions while they are running for runtime errors
    - Will kill a student's submission if it takes too long to run (Timeout)
- Will record the output generated from the students' submissions

2.8 Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are typically used to describe the qualities of a system. Given this definition, there is no concrete way to measure whether or not a non-functional requirement has been met.

Non-Functional requirements have not yet been formally defined for JOLT.

2.9 Performance Requirements

JOLT is a user interface system, allowing for cross platform compatibility. This includes Windows XP, Windows Vista, Max OSX, or Linux OS.

JOLT is a web based system and allows functionality for web browsers with internet access and support for JavaScript. Browsers tested continuously with *JOLT* include the most recent versions of Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, and Google Chrome.

2.10 Exception Handling

The *Java Online Learning Toolkit* will be designed to handle non-standard execution. The following exceptions (not all-inclusive) will be handled by JOLT:

- Exceptions generated by students' submissions (Java™ runtime exceptions) will be handled by marking the appropriate test case as incorrect, and allowing JOLT to proceed with the next test case.
- JOLT will terminate a student's submission if it takes too long to complete, or if it is stuck in an infinite loop.

2.11 Early Subsets and Implementation Priorities

There are many functional requirements of JOLT. While every attempt will be made to complete them, the following have been defined "high priority":

- The ability of Administrators to manage all other users.
- The ability of Course Coordinators to manage all classes for a particular subject.
- The ability of Instructors to upload questions, add them to question sets, and create online testing for students with these questions.
- The ability of Instructors to manage their own pool of questions and add/take from a global pool of questions.
- The ability of Instructors to view the grades of their students and the ability of Students to view their own grades.
- The ability of Students to submit answers to online quizzes/assignments.
- The ability of *JOLT* to grade a student's submitted assignments immediately.

2.12 Foreseeable Modifications and Enhancements

` Possible modifications to *JOLT* may include the following:

- Modify the way problem sets are organized (by assignment type, class, etc.)
- Allow for support of an upgraded version of Java™
- Create more pools of questions than just the global and individual instructor pools.
- Create more sophisticated error handling within the java compiler to prevent students from completing problems a certain way (example: questions to be done recursively only).

2.13 Design Hints and Guidelines

The following guidelines have been established for the design of JOLT:

- Navigation shall be implemented within JOLT in a "Breadcrumb" manner. This will provide links back to each previous page the user navigated through to get to the current page.

2.14 Acceptance Criteria

JOLT will be tested using the following general test plan:

Unit Tests- Each module within JOLT will be tested individually.

System Test- Checks to see that the individual modules work together as a whole.

Acceptance Test- Checks to see which functional requirements have been met, and which have not.

Non Functional Requirements- Since Non Functional Requirements cannot be definitively measured, *518 Interactive* will not be able to apply any formal tests. However, a sound judgment will be made by both *518 Interactive* and Dr. Lim to ascertain the effectiveness of the Non-Functional Requirements.

2.15 Testing Requirements

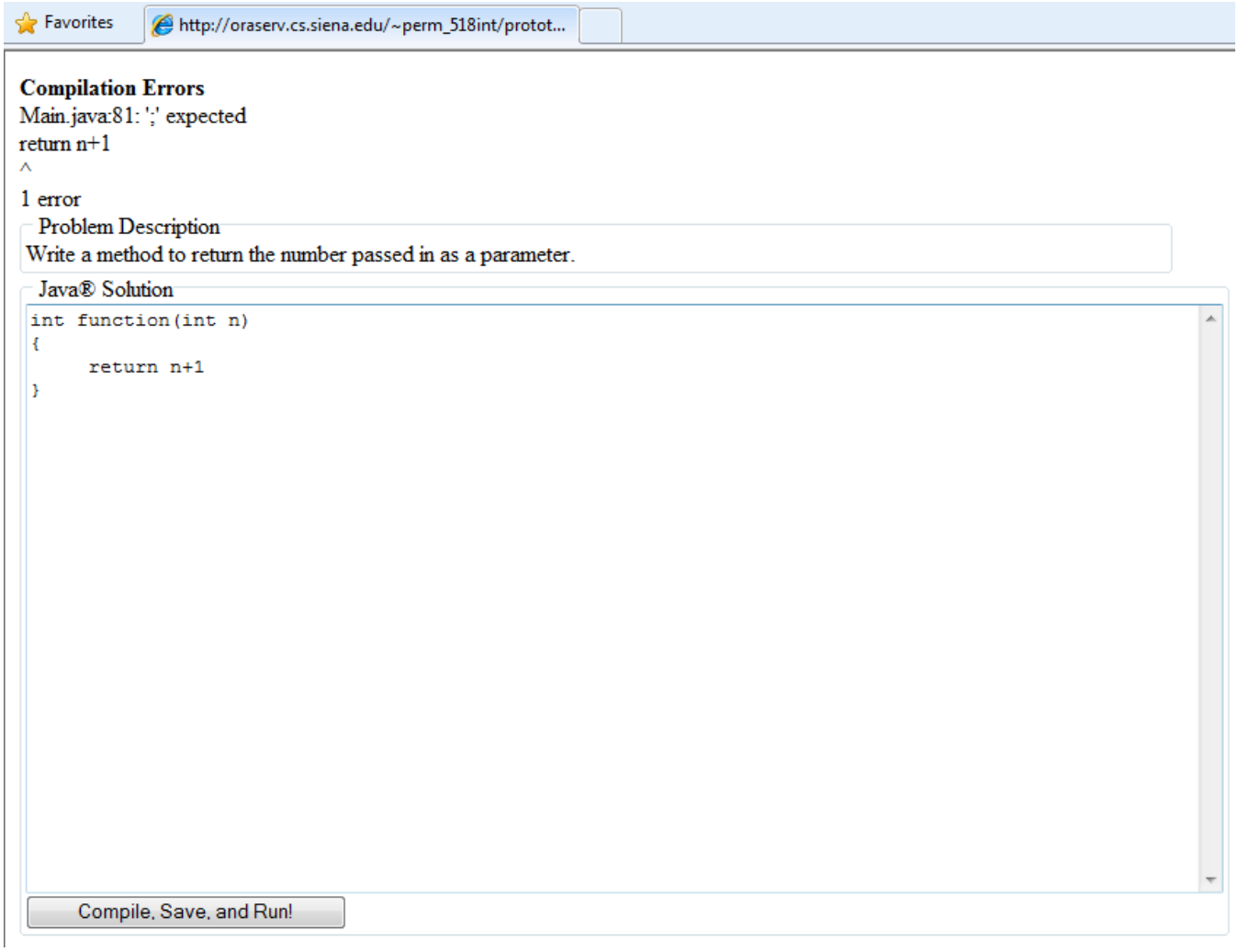
JOLT will be rigorously tested on computers running Windows Vista and Mac OS X version 10 operating systems and have the web browsers: Mozilla Firefox 3, Internet Explorer 8, Google Chrome, Safari 3 and Opera 10. Testing protocol will be designed by *518 Interactive* using the Functional Requirements Inventory to develop the Acceptance Test.

2.16 Early Prototypes

The following prototypes depict screens that were used in the discovery process. They are meant as an aid to fully develop the requirements of JOLT and do not represent the actual design or functionality of JOLT. All screens in this section are subject to revision.

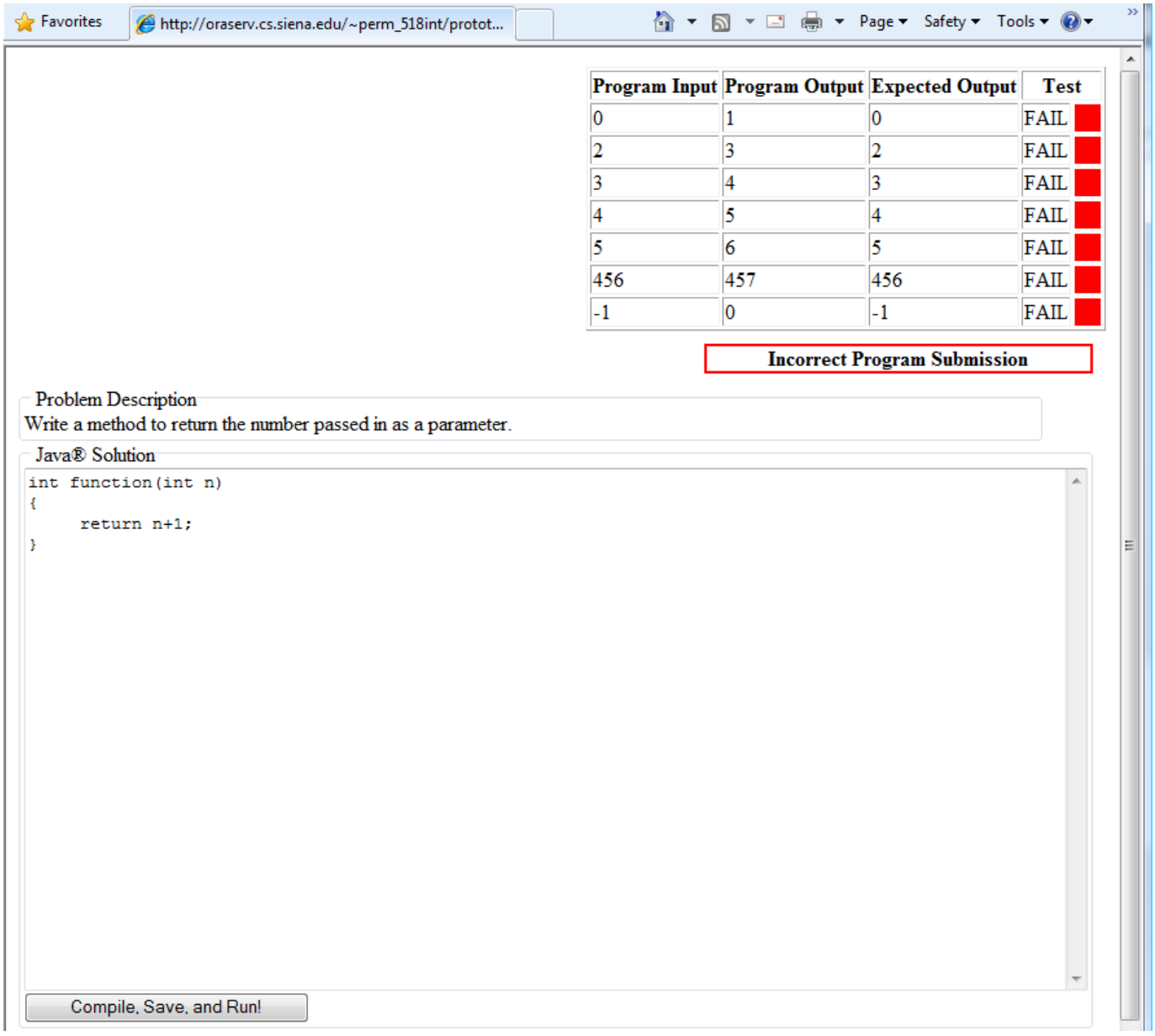
2.16.1 Prototype 1: Student Answer Problem (Compile Error)

This screen depicts what happens when a student submits Java™ code that contains errors. The compiler catches the errors and displays the appropriate error message.



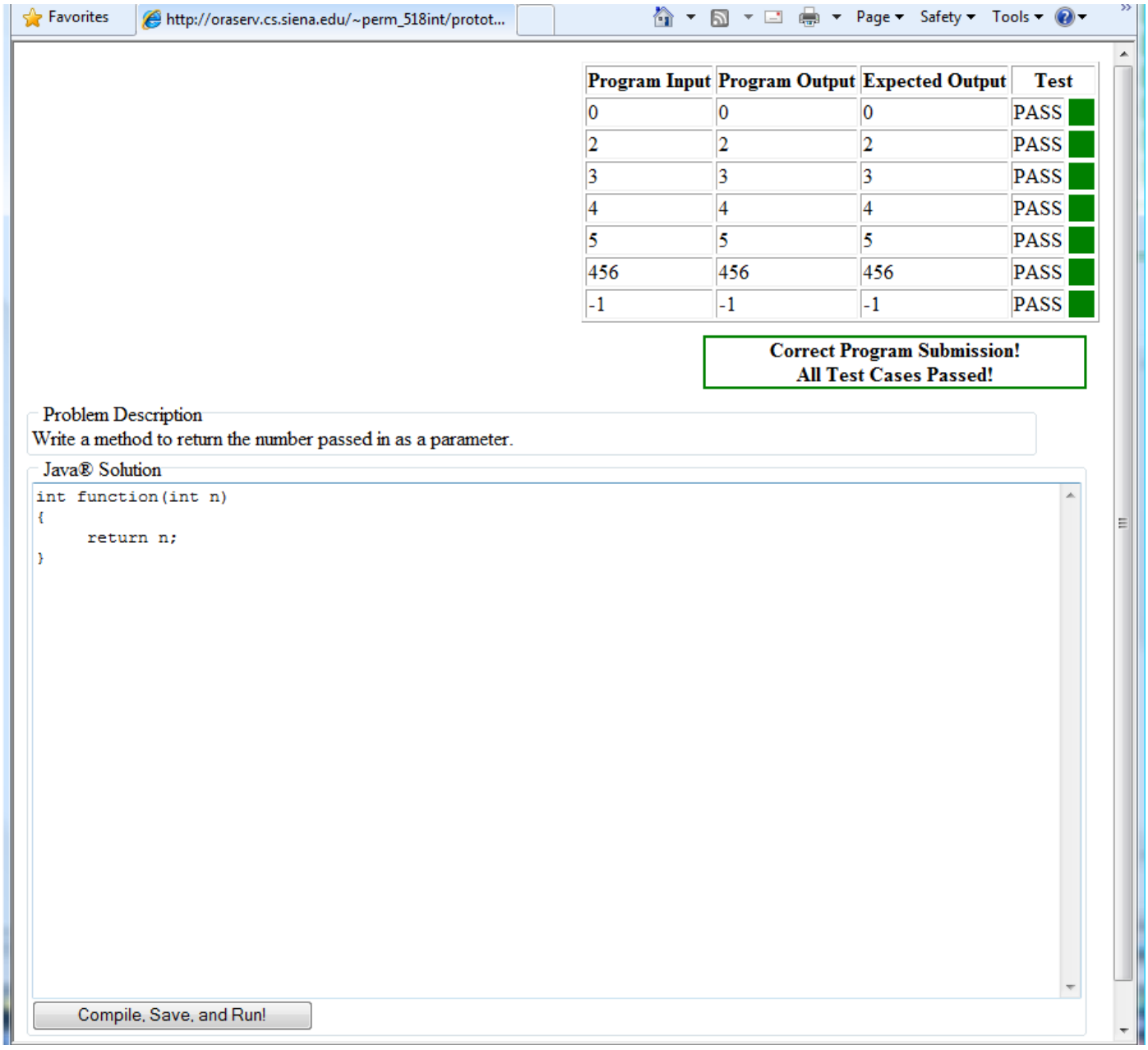
2.16.2 Prototype 2: Student Answer Problem (Incorrect Answer)

This screen depicts what happens when a student enters syntactically correct Java™ code that produces the wrong output for one or more test cases.



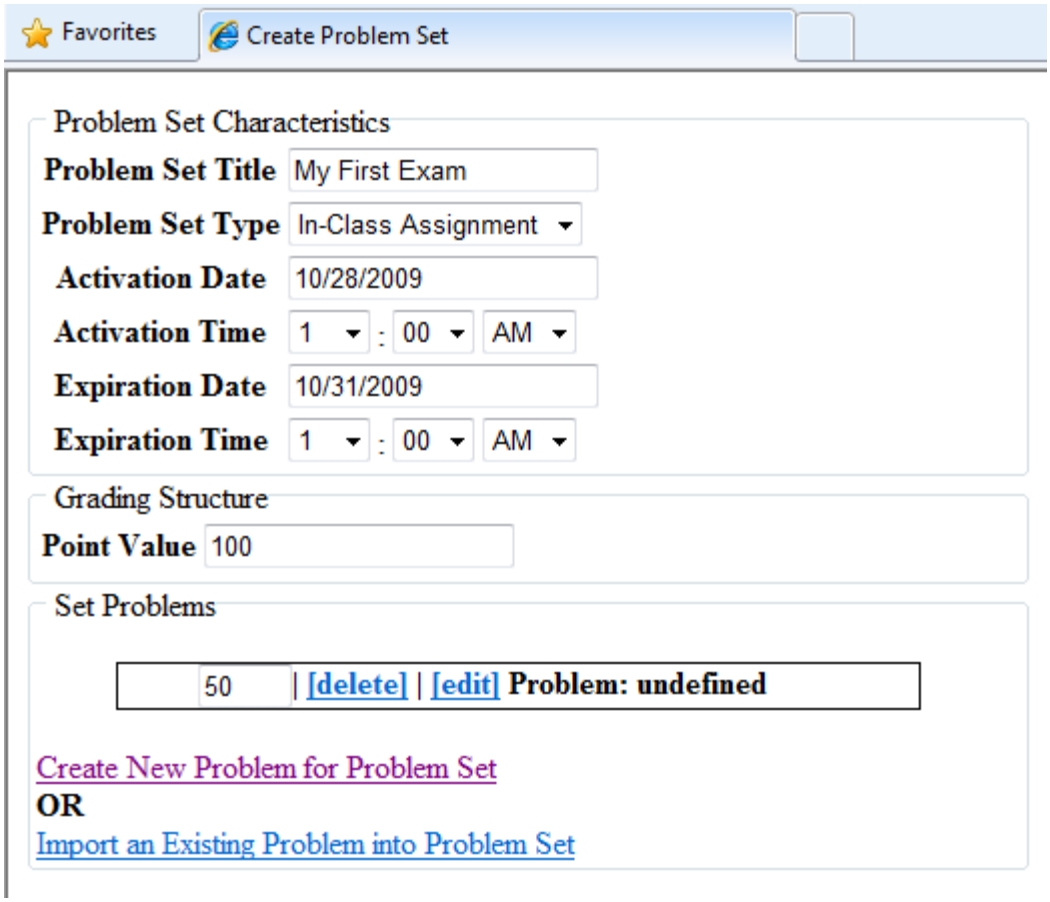
2.16.3 Prototype 3: Student Answer Problem (Correct Answer)

This screen depicts what happens when a student enters syntactically correct Java™ code that produces the correct output for all test cases.



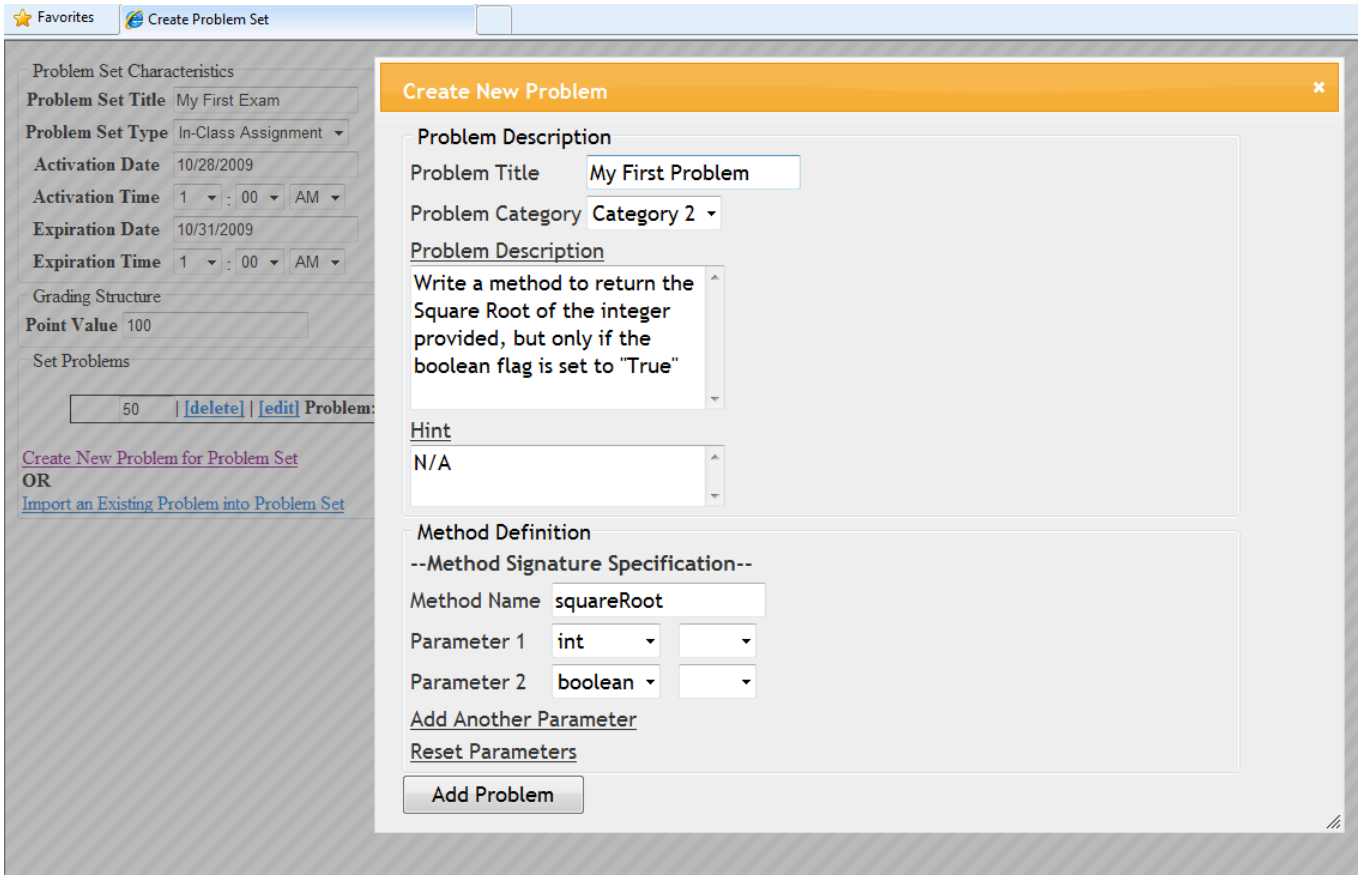
2.16.4 Prototype 4: Faculty Create Problem Set

This screen depicts the creation of a problem set. There is a section labeled “Set Problems”, which shows each problem currently in the set. Each problem in this section has a text field to enter the point value. There are also links to remove the problem from the set as well as to edit the problem.



2.16.5 Prototype 5: Faculty Create Problem Within Problem Set

This screen depicts the creation of an individual problem within a problem set. This dialog appears when the "Create New Problem for Problem Set" button is pressed. There is a spot to define the method name, as well as all parameters that may be needed. Clicking the "Add Problem" button adds another problem to the "Set Problems" section of the Problem Set page underneath the dialog.



3 Appendices

3.1 Appendix 1: Sources of Information

Information found within this Requirement Specification document has been obtained through meetings with our client, Dr. Darren Lim. Information was also obtained through Dr. Lederman's Software Engineering lectures. Information has also been collected from various internet resources, as well as requirement specification documents from previous years.

3.2 Appendix 2: Glossary of Terms

The following are a list of technical terms used within the document. This section is provided to clarify their meaning.

Actor: An entity in UML Use Case Diagrams and UML Activity Diagrams. It represents the human and non-human external entities (outside the system boundary) that interact with the system.

Activity Diagram: A diagram based on the Unified Model Language (UML). This represents the processes that comprise a certain activity within the system. These diagrams are generally created with the perspective of an actor in mind.

Client: Used to refer to Dr. Darren Lim, the client of *518 Interactive* who requested JOLT.

Compiler: A program that reads in source code and generates an executable.

CSS: Cascading Style Sheets – Used within HTML documents in order to control the presentation of web pages.

DFD: Data Flow Diagrams are used to show how data moves and is processed within a system. There are various levels to DFDs, with each subsequent level providing more detail than the previous.

Hardware: The tangible components of a computer and server. Examples include monitors, disk drives, printers, keyboard, processor, and memory.

HTML: Hypertext Markup Language is the scripting language used to describe the information contained on a website. HTML utilizes Cascading Style Sheets (CSS) to generate the style of the page. HTML and CSS are parsed by web browsers, such as Internet Explorer and Firefox, to render the websites for users.

Java: A programming language which the System will be able to compile and execute. This language will be used by the students to solve the assigned problems.

Java Byte Code: The output of the Java™ compiler upon successful compilation of Java™ source code. Java Byte Code is read by the Java™ runtime environment, which in turn executes the proper machine-level commands.

Java SDK: Software Development Kit for Java – a collection of tools used by developers to aid in the creation of programs. The Java SDK includes the Java™ (V. 1.6) compiler. The Java™ SDK also includes the Java™ (V. 1.6) runtime environment, which allows for Java™ Byte Code to be executed.

JOLT: *Java Online Learning Toolkit* is the name of the system being developed for Dr. Lim, the client of *518 Interactive*.

MySQL: A free implementation of a Relational Database Management System. Used to store and retrieve information relevant to the website, such as usernames, passwords, problems, solutions, and scores. Accessing information within the database is achieved by submitting a “query” in the Structured Query Language (SQL) form.

PHP: PHP Hypertext Processor is a programming language used to create dynamic web sites. Has the ability to interact with a database.

Software: The intangible components of a computer and server. It is a set of machine-level instructions that is run from within the memory, and is used to perform a specific set of functions. Examples include Microsoft Word, Adobe Photoshop, and Mozilla Firefox.

Source Code: A document that a compiler parses to generate machine code (which the computer can run directly), or code that gets interpreted by a third-party application, which then gets executed.

Source/Sink: This is a term used within Data Flow Diagrams to represent an entity that either provides (source) or receives (sink) data.

System: Used within this document to describe the Java Online Learning Toolkit (JOLT).

UML: Unified Modeling Language is the industry-standard language for the specification, visualization, construction, and documentation of the components of software systems.

Use Case Diagram: Represents the high-level functions of the system. It also depicts how actors interact with each of those functions.

3.3 Appendix 3: Timeline (GANTT Chart)