

# ICS 121 / Informatics 111: Software Tools and Methods Fall Quarter 2005

## Assignment 2

Due Monday, November 7, 2005 at 11:59 pm

### Part 1. Effort Estimation (30 points)

This problem is the similar to Part 1 from Assignment 1. If you added categories previously, you can use them again for this assignment if that seems appropriate.

**1.1 A priori estimate (4 points)** When you start working on this assignment, write down an estimate of how long it will take you to complete each section of this assignment. Break down this estimate according to both the sections of the assignment and the categories of tasks.

The table for breaking down the estimate might look like the following. You can change the column categories to match what you are using.

Task	Project Management	Implementation	Testing	Reading	Writing	Research	Process Improvement	...	Row Sum
2.1									
2.2									
...									
3.4									
Column Sum									

The sum of the rows must equal the sum of the columns, which is the same as the overall estimate that was generated. An Excel spreadsheet is available in Assignment2.zip to help with this.

**1.2 Logging (16 points)** As you are working on the assignment, record what you are doing and how long you spent, in the same manner as Assignment 1 (either one combined or two separate logs). For this question, quantity of data, i.e. number of entries, is important. As a rule of thumb, you should add a log entry every time you switch tasks or at least one entry per hour. If you two or three things in half an hour, you must have a log entry for each of them. You do not need to include time for logging, but should include the time spent answering the other parts of this question.

Place your log under revision control. (It can be in the same project as the Scheduler in Part 2.) Every day that you update your log, you must check it in to CVS. You will be submitting the revision history along with the log in your written report. This revision history can be submitted as a screen shot of the history view in the CVS Perspective of Eclipse.

**1.3 Summary and Error Calculation (4 points)** For each of the entries in the time log, allocate the time spent to a cell in the table. Calculate the error for each cell. Calculate summary statistics for the rows and columns. The supplied spreadsheet makes this fairly easy.

**1.4 Discussion (6 points)** Did you benefit from the experience you gained in doing an effort estimate for Assignment 1? Did you learn anything new this time? Did you do the overall estimate first and break it down for the table? Or did you generate estimates for the table cells and add them together for the total estimate? Does breaking down the estimate make the job easier or harder?

### **Deliverables**

- A log of your time spent on the assignment
- CVS revision history of the log (screen print)
- A table or spreadsheet similar to Part1.xls showing effort estimate, actual time spent, and the error. Include summary statistics for the row and column totals.
- Discussion

## **Part 2. Unit Testing (70 points)**

For this part, you will be unit testing a component of a seminar scheduler. There are four classes in the package, `Attendee.java`, `Seminar.java`, `Scheduler.java`, and `Invitation.java`. Seminars are attended by Attendees. For each Seminar, the Scheduler generates Invitations to send to each person (Attendee). The Invitation tells the Attendee when the Seminar will occur in his or her local time zone. This component is intended to be part of a larger seminar scheduling system. For example, there's no UI for entering the information and no back end to store information about people and seminars. But we'll leave that for Assignment 3.

In this assignment, you will need to create a project in Eclipse for the scheduler package. After that there are three tasks: develop JUnit tests, correct the defects in the scheduler package, and write a Test Report. Although these tasks are described sequentially, you may find it easier to do them in parallel. For example, outlining the test report will help you develop a test plan that will guide development of the test cases.

### **2.1. Set up a JUnit Project in Eclipse (5 points)**

- Download files from Assignment2.zip for this assignment. The zip file contains four Java files and a test stub class.
- Create a project in Eclipse called "Assignment 2" with a package for the source code as distributed and a package for the test cases you are going to create.
- Set the javac compliance level to 1.4.
- Decompress the zip file and import the .java files into the project.
- Place this project under revision control using CVS.

## **2.2 Convert TestStub.java into a JUnit Test (5 points)**

A test stub has been included in the code provided to you. However, this test stub has been implemented as a simple Java class rather than as a JUnit task. Convert this test case into a JUnit test class.

In order to use JUnit effectively, you will need to consult the Javadocs and other documentation on <http://www.junit.org/>. You may also want to consult the recommended text for this course, “Eclipse in Action” by Gallardo, Burnette, and McGovern.

## **2.3 Develop JUnit Test Cases for the Scheduler Package (30 points)**

Implement test cases for Attendee class in AttendeeTest, and test cases for Seminar in SeminarTest, and so on for Invitation and Scheduler classes. If you want to create additional JUnit test classes, use a sensible naming convention or add numbers after these class names. The test methods must begin with "test," so JUnit will find them, but otherwise give them meaningful names. Your test cases must be placed in one or more test suites.

Your objective is to find as many defects as you can in the source code. To help you get started, here are some places to look for defects.

- Input checking (range, validity)
- Constructors
- Format of dates, email addresses, etc.
- Correctness of the scheduler
- Time zone handling, especially daylight saving vs. standard time
- Leap years and days

## **2.4 Correct Errors the Scheduler Package (10 points)**

Based on your test cases and the defects that you found, modify the classes in the Scheduler package to make your test cases run successfully.

## 2.5 Compose a Test Report (20 points)

Write a report to document the test cases that you developed. The report must include a description of each test case that was implemented. The report should be organized according to the strategies or approach used to identify the bugs. For instance, all the tests of constructors can be grouped together, or all of the tests for a class can be grouped together, or all of the statement coverage tests can be grouped together. In any case, the organization should reflect the logic behind your choice of test cases.

Documentation for individual test cases must include the following information:

- **Unique name / ID for the test case**
- **File name** – Name of JUnit .java file in which the test case resides.
- **Purpose** – Goal of test case, including name of class and method tested, when appropriate.
- **Prerequisites** – System or user information that needs to be in place prior to test.
- **Test Data** – Input data that will be used in the test case.
- **Expected Result** – The output that is expected from the schedule system program.
- **Defects Exposed** – The defects that were revealed by the test case (in the original, unmodified Scheduler system). Information on both the problem and the location of the defect in the code should be included.
- **Corrections** – Your correction or solution to the defect (English summary, not source code).

### Deliverables

- Screen shot of the project opened in Eclipse's Resource Perspective
- Source code for the JUnit test classes
- Source code for modified versions the Scheduler package programs
- Test Report

## Handing In Your Assignment

Your assignment must be submitted electronically to [checkmate.ics.uci.edu](mailto:checkmate.ics.uci.edu). You will submit three files:

1. A file named part1.doc or part1.pdf containing all deliverables for Part 1.
2. A file names part2.doc or part2.pdf containing the screen shot and Test Report for Part 2.
3. A zip file called part2.zip with the test cases and source code from Part 2.