# Assignment 2

## *CompSci 230 S1 2013*

*Clark Thomborson*

**Version 1.1**, 30 March 2013: removed erroneous mention of the midterm test; reformatted questions
**Version 1.11**, 30 March 2013: corrected a spelling error; clarified question 1d

If another version is released, it will be notified by email and web-posted to the Assignments area (http://www.cs.auckland.ac.nz/courses/compsci230s1c/assignments/).

**Total marks on this assignment**: 40. This assignment counts 4% of total marks.

**Submission deadline:** 4pm, Friday 19 April. Late submissions will be accepted but heavily penalised, as noted in Slide 7 of the Course Information. Only your last submission will be marked. The lateness penalty is

- 20% of possible marks, for late submissions prior to 4pm Monday 22 April, and
- 50% of possible marks, for later submissions prior to 4pm Wednesday 25 April.

**Submit**: ***one*** file (as a zip archive) to the Cecil dropbox. See Slide 11 of the Course Information. Your zipfile should contain

- one document (in PDF, DOC, DOCX, or ODF) with your textual answers,
- one jarfile with all of your source code, and
- one executable jarfile.

**Learning goals**: by completing this assignment, you will develop the following practical skills

- writing test cases for someone else's code;
- running tests in JUnit;
- locating and repairing a software defect; and
- writing test cases before developing your own code.

**Asking for help.** You may seek assistance from your tutor on any course-related topic. You may seek help from any other source, at any time (except during a test or examination), on learning the concepts taught in this course, on setting up your Java development environment, on using your Java development environment. If any aspect of this assignment handout is unclear, ambiguous, or incorrect, you are encouraged to discuss this aspect with your classmates and tutor; and if a clarification or correction is required, please send an email to cthombor@cs.auckland.ac.nz with the subject "Bug report on Assignment 2".

**Working independently.** Your answers to the questions on this assignment will be individually marked, and **must be your own work.** You will be assigned 0 marks for this entire assignment, if any of your answers to individual questions bears a close resemblance to another student's submission, or to something previously published on the internet or elsewhere – unless you make it clear, with a citation of a published work, that you are extending a previously-published design.

**Resource requirements.** This assignment can be completed if you are using only the basic javac/java support provided in any recent SDK for Java, with a download of JUnit 3 or 4 from http://junit.org/. However you are encouraged to take this opportunity to learn how to use JUnit 3 or 4 in Eclipse. You will also need to download the following resources from http://www.cs.auckland.ac.nz/courses/compsci230s1c/assignments/

- My sample answers to Assignment 1, and
- Asst2.jar. This contains a slightly modified version of the source code I developed for my sample answers to Assignment 1. (I injected one bug into my source code before producing Asst2.jar. I am aware of other bugs in this code. One of your challenges in this assignment is to find bugs: happy hunting!)

**To get started.** You should confirm that you are able to write a simple JUnit test in your development environment *before* you attempt to answer any of the questions in this assignment. You may use either JUnit 3 or JUnit 4. Suggested resources:

- CompSci 230 tutorials during Week 5
- JUnit Tutorial v2.3, by Lars Vogel, available 29 March 2013 at http://www.vogella.com/articles/JUnit/article.html. This tutorial uses JUnit 4.
- Writing and running JUnit tests, in the Getting Started section of the Java Development User Guide for Eclipse, available 29 March 2013 at http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2FgettingStarted%2Fqs-junit.htm. This tutorial uses JUnit 3.
- JUnit Tutorial, by mkyong, available 29 March 2013 at http://www.mkyong.com/tutorials/junit-tutorials/. This tutorial uses JUnit 4.

**Questions to be answered.**

1. **(17 marks)** Refer to the Quiz class, in the class diagram in the sample answers to Assignment 1. **Write 20 test cases for the summariseScores() method**, with

    a. good coverage (3 marks) of the three methods for computing a SummaryStatistic on a vector of integer scores,

    b. good coverage (4 marks) of different lengths of the score vector,

    c. good coverage (5 marks) of integer values within the score vector, and

    d. some coverage (2 marks) of non-integer values in the score vector. (Note: if your JUnit tests for these cases cause a compilation error, you should comment them out with a brief explanation which includes the word **JAVA_TYPE_SAFETY_SUCCESSFUL_TEST**.)

    e. If any of your tests fail, you should spend up to thirty minutes in an attempt to

        i. identify and repair the bug, adding a comment which contains the word **BUGFIX**.

    f. If completing a bug-fix in part 1e would take more than thirty minutes, or if your test is of questionable validity (with respect to the requirements on the prototype, as stated in Assignment 1), you should

        i. invert the direction of your test (assertTrue() to assertFalse(), or vice versa), so that your test will succeed, and

        ii. add a comment to describe the bug and any questions you have about the specification -- this comment must include the word **UNVERIFIED_BUG**.

    In the jarfile you submit for marking,

    g. all of your tests must succeed (1 mark) ; and

    h. there must be at least two comments (2 marks) containing either a **BUGFIX** or an **UNVERIFIED_BUG**.

    In your written answers for this assignment, you should include

    i. a listing of your testSummariseScores() method, as well as

j. listings of any methods you modified with a BUGFIX.

k. If your JUnit fixture uses a setup() method, your written answers should also include a listing of this method.

2. **(5 marks)** Write test cases for the sitQuiz() method of the Quiz class, with

   a. good coverage (3 marks) of the length of the score vector.

   b. Note: your test should use the attemptsAt() method of the Scoresheet class – this method is public in Asst2.jar even though it does not appear in the class diagram of my sample answers to Assignment 1.

   c. If your tests reveal one or more bugs, you should repair these.

   d. Your written answers should include

      i. a listing of your testSitQuiz() method. Your written answers should also include

      ii. (2 marks) either a listing of any methods you repaired with a **BUGFIX**, or (if you didn't discover any bugs) a brief statement of your strategy for generating test cases.

3. **(18 marks)** You have been asked to modify the code in Asst2.jar, so that the modified prototype demonstrates how QuizWhiz would compute scaled marks -- as specified immediately after this question. You should start by designing your test cases in JUnit, with

   a. good coverage (5 marks) of the most important aspects (5 marks) of the specification.

   b. Your written answers should include

      i. a listing of your testScaledMarks() class.

   c. All of your tests must succeed (1 marks) in your executable jarfile. If you discover any bugs in Asst2.jar while developing your implementation, you should

      i. repair these bugs with a **BUGFIX** comment (or alternatively with an **UNVALIDATED_BUG** comment) (2 marks) as in the first question on this assignment.

   d. Your written answers should also include

      i. a brief discussion (2 marks) of your test strategy, that is, of what aspects of the specification you have attempted to cover and (very roughly) how you went about covering these aspects.

   e. After writing your tests, you should modify the behaviour of the reportOnMarks() and reportOnCourse() methods in the QuizWhiz class of Asst2.jar, so that it prints scaled marks rather than summarised scores.

      i. The jarfiles you submit for marking should include your modified QuizWhiz class.

      ii. Your executable jarfile should print the following message (3 marks) when it is executed.

```
Report for instructor I1: In C1, student S1 made 1 attempt at Q1.  Marks = 0.0

Report for student S1: In C1, you made 1 attempt at Q1.  Marks = 0.0
```

**Specification of scaledMarks().** You should add instance variables to the Quiz class, so that each instance of this class records an (integer) maxScore, as well as a (double) maxMarks with the following semantics. The maxScore of a quiz is the highest value that QuestionMark will award to any student taking this quiz. In other words, QuestionMark should report an integer in the range 0..maxScore as the "score" obtained by a student who attempts this quiz. The maxMarks value is used to scale a student's summarised score (e.g. their maximum score, or their average score), for purposes of computing total marks in a course. Thus, scaledMarks() should return a real value in the range 0.0 to maxMarks. For example, if a student gets a score of 1 on their only attempt at a quiz with a maxScore of 3, their scaledMarks() would be 0.1 if maxMarks = 0.3. The general formula is scaledMarks() = summarisedScore / maxScore * maxMarks.