Assignment 3

CompSci 230 S2 2014
Diana Kirk

Submission deadline: 4p.m. Friday 3 October 2014

Version 1.0 of 19 September 2014

Total marks on this assignment: 40. This assignment counts 4% of total marks in COMPSCI 230.

Learning goals. By completing this assignment, you'll get some practical experience with exploring black box testing techniques. You'll also get some practical experience with using JUnit, the de-facto industry standard environment for unit testing Java code.

Asking for help. If you are uncertain about what is required for this assignment, you should first review the lecture notes (Lecture D03 – Black box testing) and consult the prescribed readings. You are welcome to seek assistance from others on "learning the concepts" *after* you have done your assigned readings for this course.

Working independently. You are *not* allowed to have assistance from any other person when you are completing any part of this assignment. This must be your own work, done independently. You may gain design ideas or examples from the internet or a textbook. You should not assist another student with any part of this assignment. However you may help one another with setting up the JUnit environment, if needed, or with using the debugger.

English grammar and spelling. You will not be marked down for grammatical or spelling errors in your submission. However if your meaning is not readily apparent to the marker, then you will lose some marks.

Resource requirements. You'll need

- JUnit
- The compressed file **A3v1.0** – this is available for download at https://www.cs.auckland.ac.nz/courses/compsci230s2c/assignments/A3v1.0.zip. This contains:
  - The UNIVERSITY codebase (includes a skeleton test class).
  - JavaDocs for the *Enrolments* class.
  - The specification of the enrolment data used by the *Enrolments* class.
  - A skeleton Test Design document.

See the Appendix for instructions for importing the UNIVERSITY codebase into Eclipse.

Submission instructions. You must submit electronically, using the Assignment Drop Box (https://adb.auckland.ac.nz/). Your submission will be **a single compressed file** containing:

- an Open Office document that is your submission for Part 1 below (Test Design)
- a compressed file containing the codebase for your submission for Parts 2 and 3.

Note. If you handwrite your answers fpr Part 1, you must scan it before submission. The printer/photocopiers in some computer labs can scan documents and email them to your aucklanduni account. You should learn how to use this functionality well in advance of the submission deadline, to avoid last-hour frustrations and a penalty for a late submission.

ASSIGNMENT OVERVIEW

You are employed by a small software company as Test Engineer. The company has recently hired a new developer to work on a University Enrolment application. The application will eventually interface with the university's student database and will be accessed via a web interface. As a first step, the developer has been asked to produce a small API that will allow a user application to enrol a student in a paper that is delivered in the current semester. Your task is to design and create a set of JUnit tests to test the method ***enrolStudentInPaper().*** Your tests will be used as an aid to establishing desired functionality and for ongoing regression testing. As you also have experience as a developer, you will then be required to fix the defects found when testing.

Although the use of source control is not marked in this assignment, you will be working with several versions of code and are advised to manage these in Subversion.

Part 1: Designing your tests
(15 marks)

Study the javadocs for the ***Enrolments*** class (see document ***compsci230Ass3(EnrolmentsJavadocs)).*** The API exposes 3 methods :

- enrolStudentInPaper() — This is the method you will test. Enrolling a student involves including the student in the paper's list of enrolled students AND including the paper in the student's list of papers.
- listPapersForStudent() — You may use this to help you test.
- listStudentIdsForPaper() — You may use this to help you test.

Eventually, the application will load students and paper data from the UNIVERSITY database. For this draft version, data is hard-coded in the API. Check the document ***compsci230Ass3(EnrolmentDataSpecification)*** which shows a list of valid students and a list of papers and the year and semester in which each is delivered.

Your task is to design a set of tests aimed at exposing defects in the method ***enrolStudentInPaper()***. You should test this method from the perspective of a user application. You should take a black-box approach and apply partitioning and boundary value techniques. You will be awarded 1 mark for each test you define, up to a maximum of 15, as long as the tests focus on different usage aspects. You are encouraged to design as many tests as you can think about. Extra tests won't earn you marks for Parts 1 and 2, but will be important to help you uncover bugs in Part 3.

You will deliver a .pdf document containing a table with your test designs - see the example document ***compsci230Ass3(TestDesignExample)***. The table includes a call to 'enrolStudent()' with a valid student and valid paper, and defines three tests that cover the expected outcomes of the method call i.e. the enrol method should return 'true', the paper should be in the student's list of enrolled papers and the student should be in the papers list of enrolled students.

Part 2: Implementing your tests
(15 marks)

Import version 1.0 of the UNIVERSITY codebase into Eclipse (see the Appendix for instructions). You will see a 'src' package 'University', with 3 classes. The class 'Enrolments' contains the API under test. You will also see a 'test' package 'University', with the class 'EnrolmentsTest' – this contains some skeleton test code for you to use. Note that the skeleton code is consistent with the test design example.

You should implement the tests you designed in Part 1 in the 'EnrolmentsTest' class. When you run your tests, you should find that many fail – this is good as it means your tests are successfully identifying bugs in the code. YOU SHOULD NOT CHANGE ANYTHING IN 'src'. For Part 2, it does not matter whether your tests succeed or fail.

You must present your tests in the same order as in your design document from Part 1 and should name each test in a way that makes it clear to the markers what you are testing. You will be awarded 1 mark for each test you implement, up to a maximum of 15, as long as the test is consistent with a test you designed in Part 1. You will not be awarded marks for tests that are not easily identified in your design document.

Note that you may present each test in it's own test method OR you may group tests for the same method call together. The main presentation objective for this assignment is clarity.

Part 3: Fixing the bugs
(10 marks)

There are at least 4 defects in the UNIVERSITY codebase. You should now use your tests to help you identify and fix these. Clearly, your success in this will depend upon how good your tests are. You may at this stage want to include more tests to help you.

You will be awarded 2.5 marks for each defect you successfully fix. For each, you will get:

- 1 mark for fixing the defect
- 1 mark for including the change in the javadocs at the top of the file (for example, use the @version tag to say who you are, the date and a short (but meaningful) description of the change you made)
- .5 marks if your change is of high quality (standards have been met and you have chosen an elegant solution)

Delivery

You should deliver a single compressed fire containing:

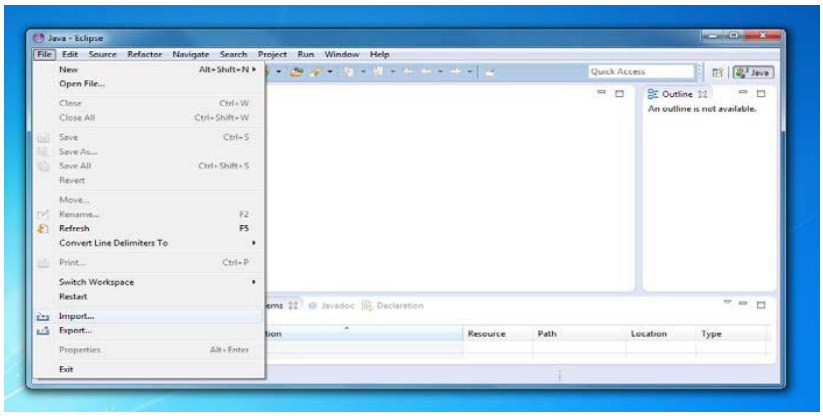
- an Open Office document that is your submission for Part 1 (Test Design)
- a compressed file containing the codebase as given to you, and with your JUnit and source code additions for Parts 2 and 3.

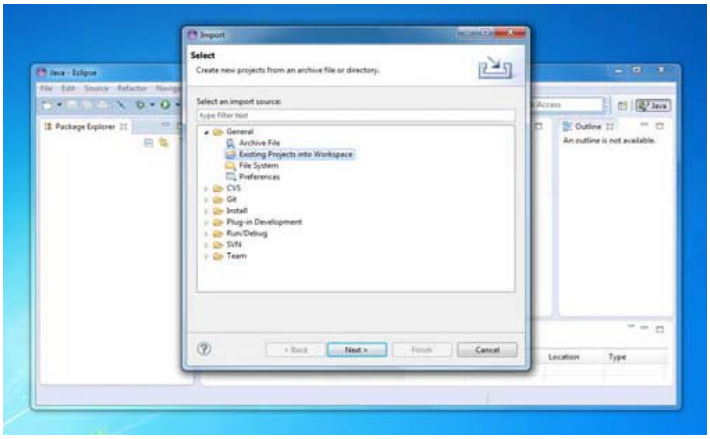
Appendix

Instructions for importing and exporting with Eclipse

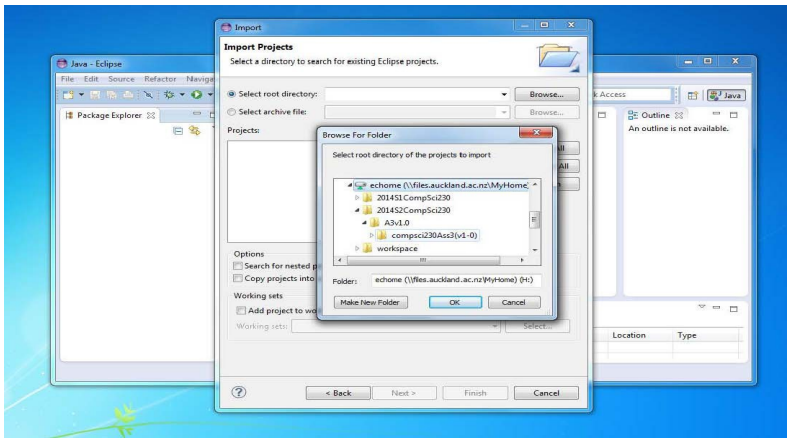
1. Unzip the compressed file **A3v1.0.zip** into the directory you have set up for this assignment. In the figure below, I have set up a directory in my Home drive. There are 4 documents:
    - The codebase, which you will import into your development environment and update in Parts 2 and 3.
    - Data specification and JavaDocs, which you will require as reference documents.
    - Test design example document (Open Office), which you may use as basis for the Design Document you will deliver for Part 1.

2. In Eclipse, select File:Import and then Existing Projects into Workspace.

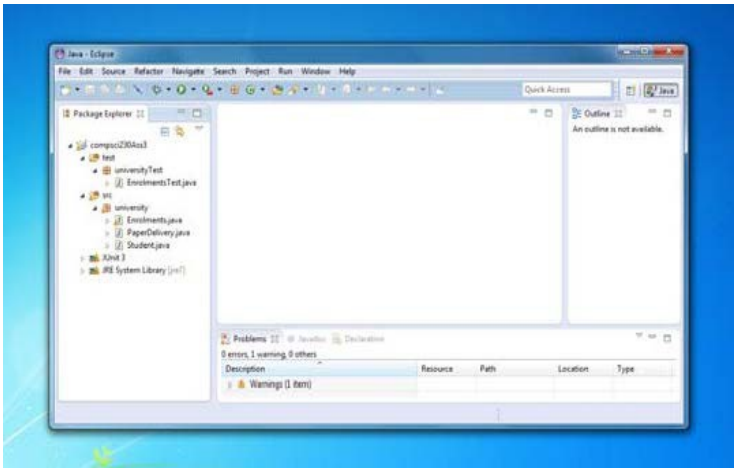




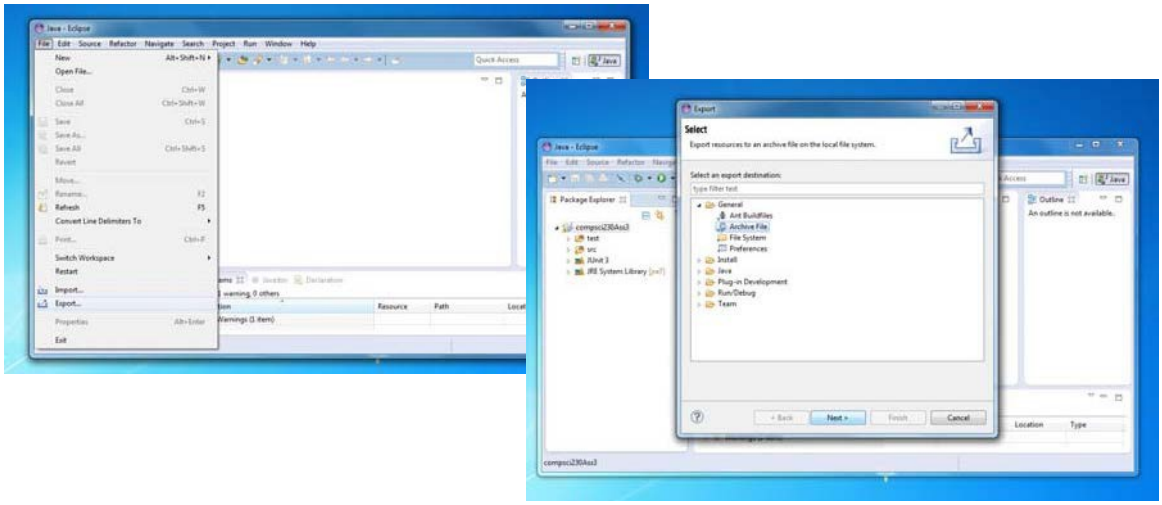
3. Select the codebase from your working directory.



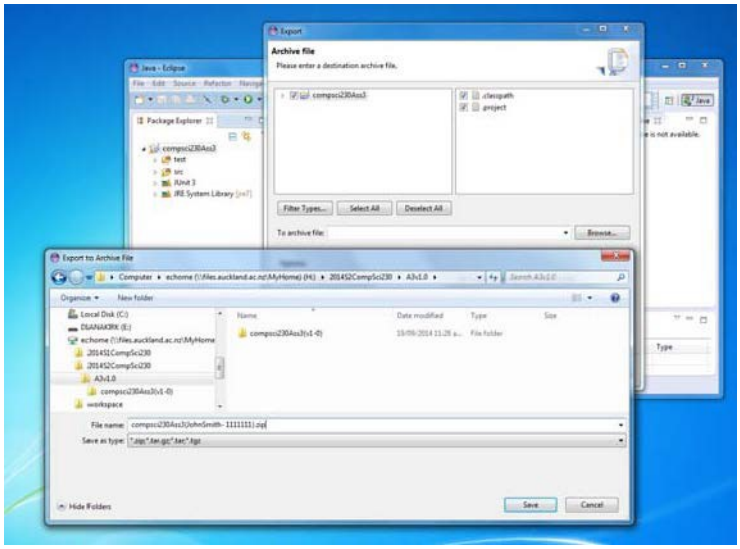
4. After import, you should see the 'test' and 'src' structure as below.



5. When you are ready to export your updated codebase, select File:Export and General:Archive File.



6. Browse to yor working directory and save the exported project. You should include some identification in the file name.



7. Compress the exported project along with your Test Design document and deliver for marking.