Assignment 3

CompSci 230 S1 2015
Diana Kirk

Submission deadline: 4p.m. Friday 22 May 2015

Version 1.1 of 14 May 2015

Total marks on this assignment: 40. This assignment counts 5% 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 **A3v0.9** – this is available for download at https://www.cs.auckland.ac.nz/courses/compsci230s1c/assignments/A3v0.9.zip. This contains:
    - The Controller codebase (includes a skeleton test class).
    - JavaDocs for the Controller class.
    - A skeleton Test Design document.

Submission instructions. You must submit electronically, using the Assignment Drop Box (https://adb.auckland.ac.nz/). Your submission will be **two files**:

- a single .pdf 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 for 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 software company as Test Engineer. The company produces software for control systems. The software interfaces with physical sensors that indicate changes in e.g. temperature and pressure and with control units that respond to changes by e.g. opening and closing valves. The role of Test Engineer involves creating and designing tests for the control software. It is expected that Test Engineers have some programming knowledge and are able to troubleshoot failed tests and locate the defects responsible.

The company is working on a major upgrade for an existing control application. As the codebase is undergoing major redesign, the Quality Manager has decided to put in place an exhaustive set of test cases for the main controller functions before upgrade activity commences. These functions include adding a device to the set of devices controlled by an individual controller, removing a device from the controller's set of controlled devices and activating control activity on a device by sending a reading from the controlled environment.

Your first task is to design and create a set of JUnit tests to test the methods ***addDevice()*** and ***removeDevice().*** 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. On the basis of your performance, the Quality Manager will decide whether you will be assigned as head Test Engineer for the project.

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 or GitHub.

Part 1: Designing your tests
(15 marks)

Study the javadocs for the ***Controller*** class (see document ***compsci230Ass3(ControllerJavadocs)).*** The API exposes 5 methods :

- addDevice() — This is a method you will test.
- removeDevice() — This is a method you will test.
- doControl() — Tests for this method will be written at a later date.
- isDeviceInList() — You may use this to help you test.
- getNumDevices() — You may use this to help you test.

Your task is to design a set of tests aimed at exposing defects in the methods ***addDevice()*** and ***removeDevice().*** As the code-under-test is undergoing significant rework, you should test these methods from the perspective of a user application. This means you must take a black-box approach. You should apply partitioning and boundary value techniques. You will be awarded 1 mark for each test you define, up to a maximum of 15 in total, as long as the tests focus on different usage aspects, represent different partitions and involve boundary values. 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 ***addDevice()*** with a valid device and valid device parameters, and defines three tests that cover the expected outcomes of the method call i.e. the add method should return 'true', the controlled list should include the device and the size of the controlled list should be 1.

Part 2: Implementing your tests
(15 marks)

Import version 1.1 of the CONTROLLER codebase into Eclipse. You will see a 'src:main' package 'Controller', with 7 classes. The class 'Controller' contains the API under test. You will also see a 'src:test' package with the class 'ControllerTest' – 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 'ControllerTest' class. When you run your tests, you should find that many have errors – this is good as it means your tests are successfully identifying bugs in the code. YOU SHOULD NOT CHANGE ANYTHING IN 'main'. For Part 2, it does not matter whether or not your tests succeed.

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 do not appear 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 5 defects in the CONTROLLER CLASS. 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 marks for each defect you successfully fix. For each, you will get:

- 1 mark for fixing the defect
- .5 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:

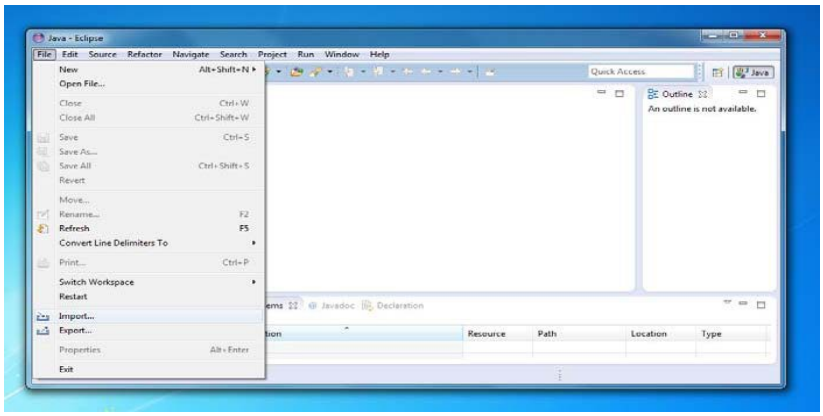
- a single .pdf document that is your submission for Part 1 below (Test Design)
- a compressed file containing the codebase as given to you, and with your JUnit and source code additions for Parts 2 and 3.

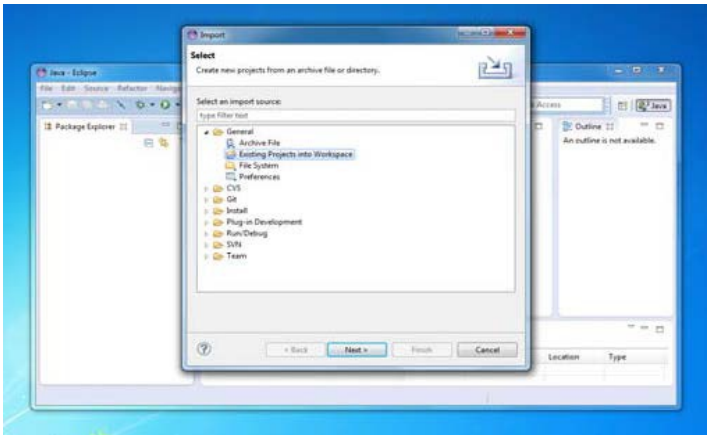
Appendix

Instructions for importing and exporting with Eclipse

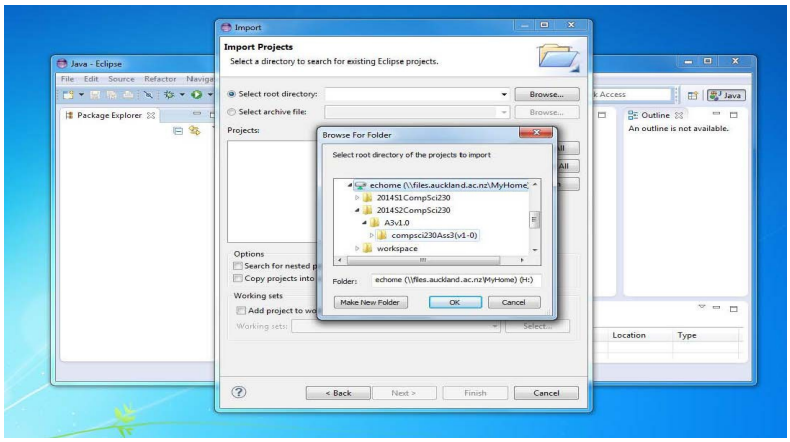
1. Unzip the compressed file **A3v1.0.zip** into the directory you have set up for this assignment. In the figure below, I have set up a directory in my Home drive. There are 4 documents:
    - The codebase, which you will import into your development environment and update in Parts 2 and 3.
    - Data 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.

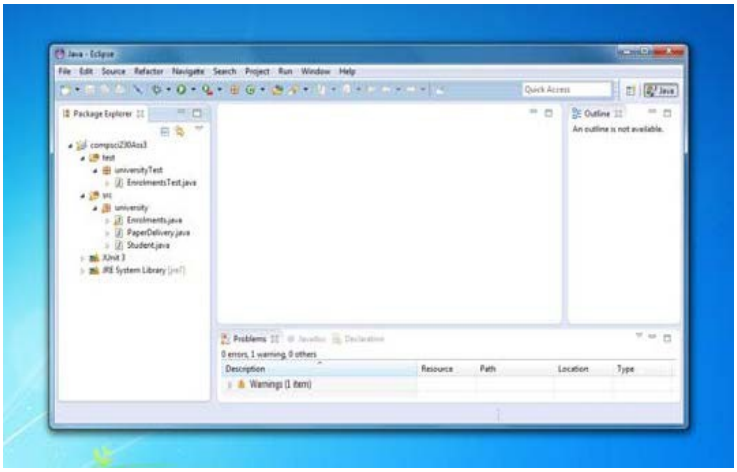




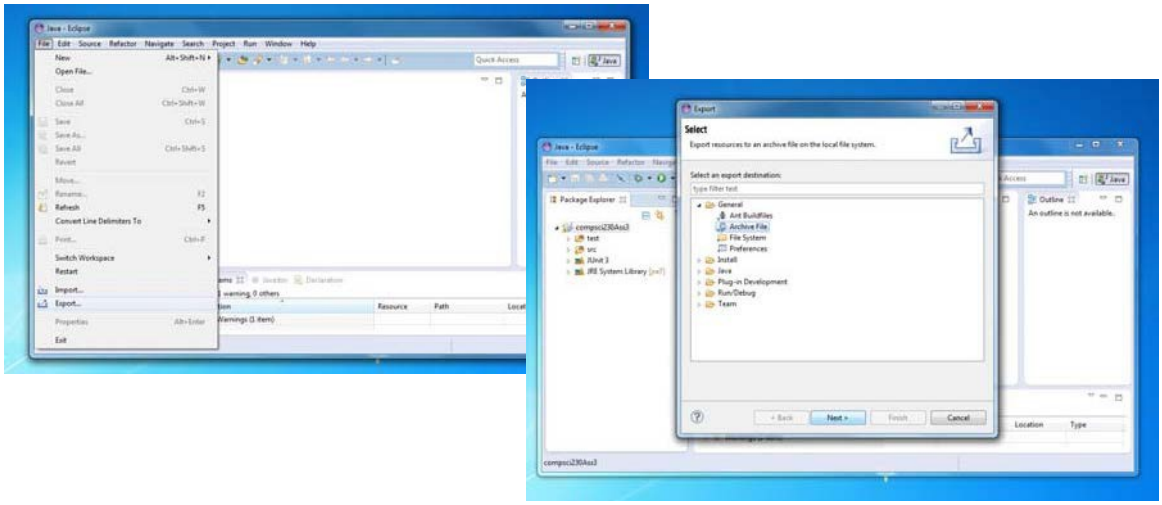
3. Select the codebase from your working directory.



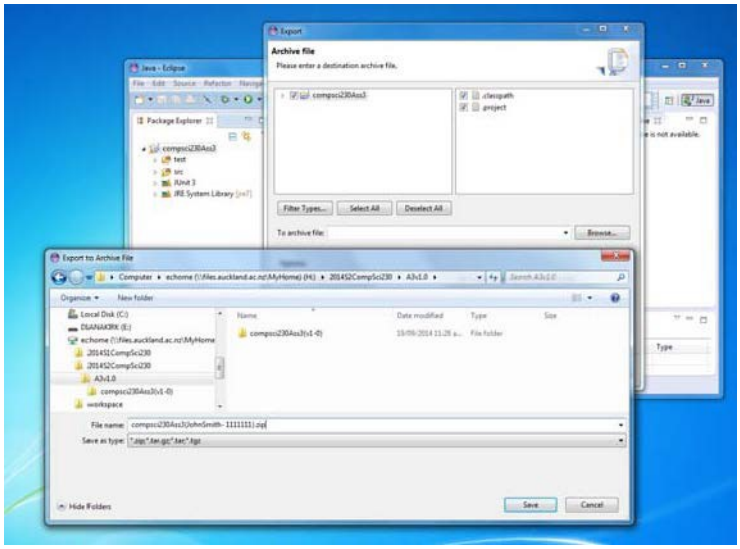
4. After import, you should see the 'test' and 'src' structure as below.



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.