# EECS 4313 Assignment 2
## - Black-box and White-box Testing with JUnit
Due: 2:29 pm, Monday, March 5, 2018

---

## 1. Assignment goals

The purpose of this assignment is to give you experience on applying the black-box and white-box testing techniques we discussed in class, as well as creating automated test code with JUnit. Your task will be to create a test suite in JUnit, produce bug reports (if any), and submit a written report describing your testing.

## 2. Getting started

### 2.1 Eclipse and JDK

Make sure you have downloaded Eclipse and have JDK 1.7 installed.

### 2.2 Importing and Building a Maven Project

Download the source code for the Borg calendar (**version 1.8.3**) and install it in into Eclipse. For those of you, who do not have much experience on working with a Maven project before, please refer to the detailed description below:

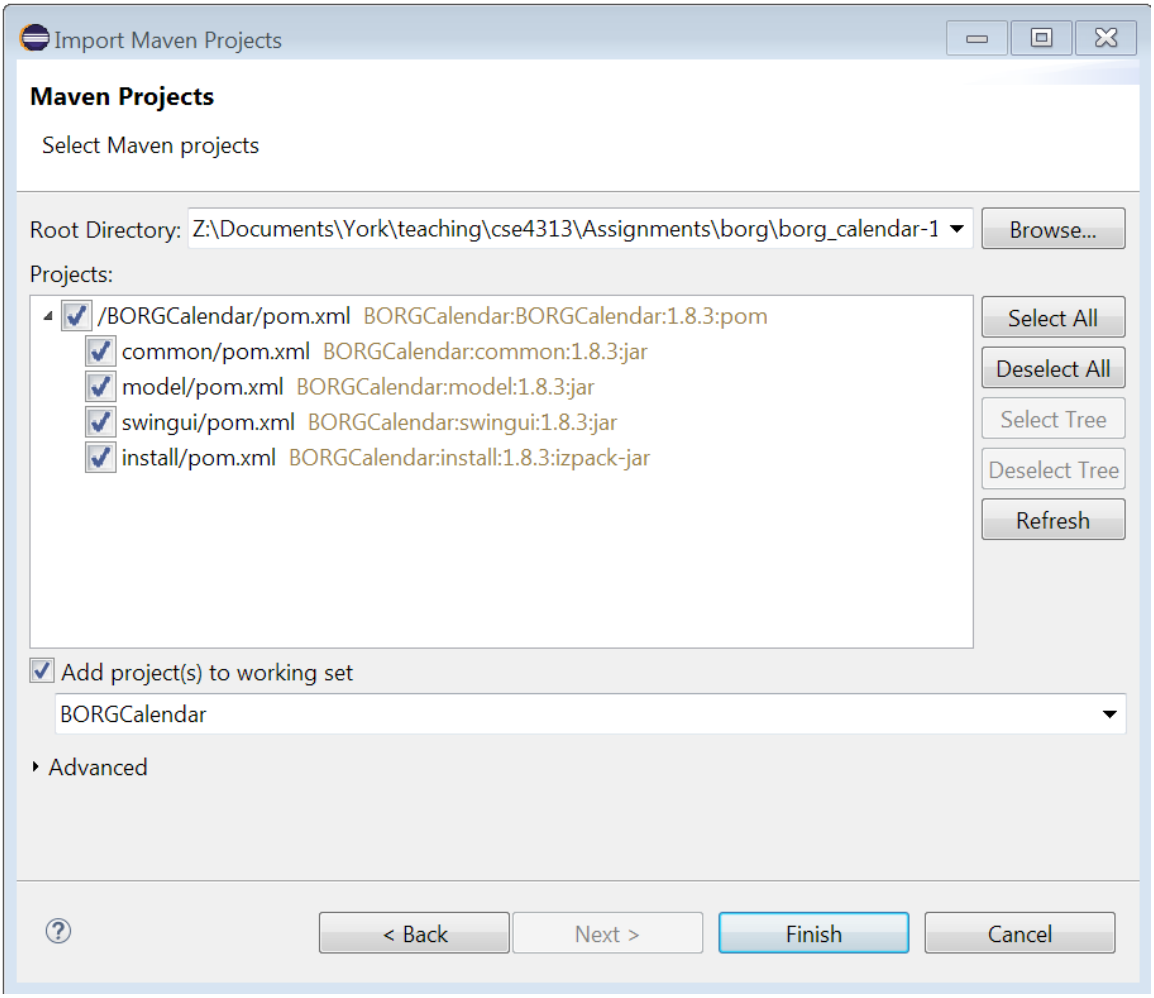
Here we describe the process of importing and building the Borg calendar into Eclipse. You may also find this link useful (https://github.com/mikeberger/borg_calendar/wiki/Building). The described process below should be similar for other projects which use Maven as their build tool.

#### 2.2.1 Install the Maven plugin

You need to install the Maven plugin in Eclipse if you have not done so. Open the Eclipse IDE, click the Help menu -> Install new software -> All Available site -> Enter the following URL http://download.eclipse.org/releases/mars, and enter "m2e" as the filtered text -> pick the items to install. Once done, close and re-open Eclipse.

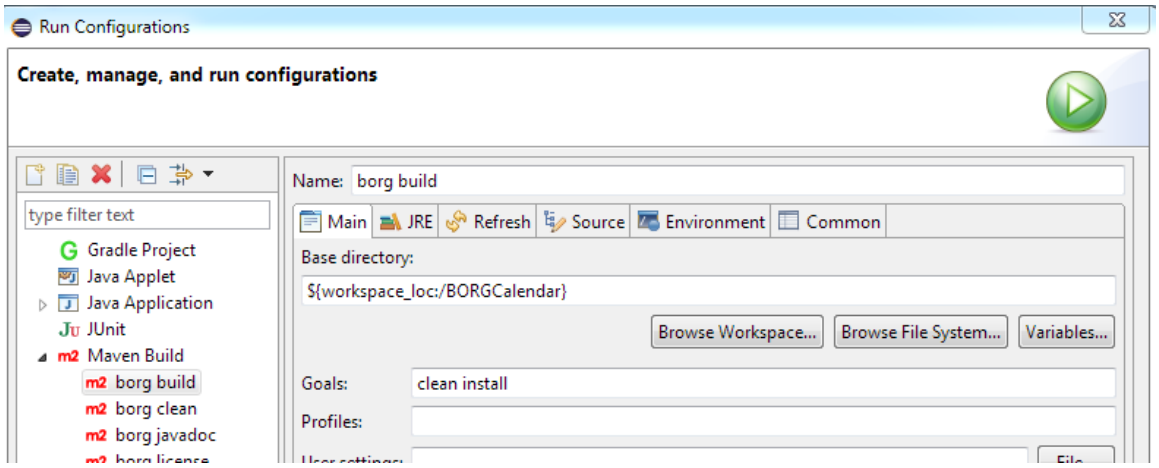
#### 2.2.2 Import the project source code

Once you have installed the Maven plugin, the next step is to import the Borg calendar source code into the Eclipse IDE. Click the File menu -> Import -> Maven -> Existing Maven Projects -> Pick the source code directory which contains the Borg source code -> Finish.



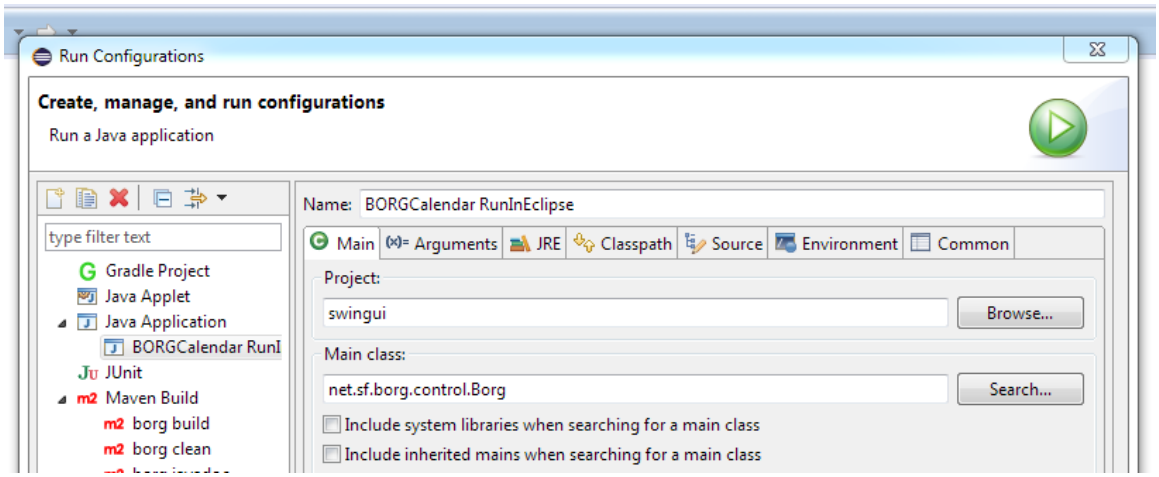

### 2.2.3 Build the Borg Calendar Application

After you have successfully imported the source code of the Borg application into Eclipse, you need to build the project. Click the Run menu -> Run Configurations -> Double click Maven build, and you will see a dialogue box, fill in the information as shown below:



### 2.2.4 Launch Borg inside Eclipse

Click the Run menu -> Run configuration -> double click Java Application -> name it as "BORGCalendar RunInEclipse -> select the swingui project -> under Main class say "search" -> pick "Borg - net.sf.borg.control" and say OK.




Once that is done, then click the Run button. If it asks whether you want to proceed with error, just click the Proceed button. You should see the Borg application launched.

**Note1:** Some students might have issues of getting the test cases run. If that is the case, you might need to download and install the latest lombok jar file. For details, please refer to this link: https://github.com/mikeberger/borg_calendar/wiki/Building

**Note2:** Some students might have issues of getting the application built using Maven. Double check your Java runtime, it needs to point to a JDK instead of a JRE.

# 3. Black-box Testing with JUnit

The purpose of this section is to give you experience on applying black-box testing techniques. Your task here will be to create a test suite in JUnit using the black-box testing techniques, produce bug reports (if any), and describe your testing in the written report.

## 3.1 Problem Description

1. Select three methods from the system you are testing. The methods must be chosen so that you can demonstrate the application of the three testing techniques discussed in class: boundary value testing, equivalence class testing, and decision table testing. It is okay to use a combination of testing techniques, but each technique must be used at least once.
2. Define a detailed specification for every selected method based on your understanding of its purpose.

Following is an example of a specification for a Java class from a sample system:

**Sample Specification**

public generateGraph (int numberOfNode, int maxDegree,boolean isConnected, String outputFileName)

This method generates a random graph and outputs it in the designated file in XML format.

- The first argument is the number of nodes that the generated graph should have.
- The second argument is the maximum degree for any node.
- If the boolean argument is true, then the produced graph must be connected in the non-directed sense. If the boolean argument is false, then the graph may or may not be connected.
- Finally, the String argument is the name of the file that will contain the produced output.

If the number of nodes or the maximum degree is less than 1, then a NegativeInputException is thrown.

## 3.2 Code Structure and Automated Testing

Other than your own testing code, you should not modify any of the existing feature code in the application. We will conduct automated testing to mark your testing code. Make sure you place all the black-box testing Java code that you have created in a folder called eecs4313a2b. *There should be no sub-folders inside eecs4313a2b!*

You can use the checks below to test your code. Failure to do so will result in loss of marks!

- The code package should be called "*eecs4313a2b*"
- Name the test suite as *EECS4313A2AllBlackBoxTests.java*
- To compile and run the black-box test suites:
  - Download all the jar files at: http://www.cse.yorku.ca/~zmjiang/teaching/eecs4313/assignments/BorgJarFiles/ and place them in a folder (e.g., AutoTest)
  - Copy your eecs4313a2b folder under the AutoTest folder
  - Open a command prompt, and go to the location where AutoTest is
    - Type the following command to compile:
      - javac -classpath .;common-1.8.2.jar;install-1.8.2.jar;junit-4.10.jar;model-1.8.2.jar;swingui-1.8.2.jar eecs4313a2b/*.java
    - Type the following command to run the test suite:
      - java -cp ".;common-1.8.2.jar;install-1.8.2.jar;junit-4.10.jar;model-1.8.2.jar;swingui-1.8.2.jar;." org.junit.runner.JUnitCore eecs4313a2b.EECS4313A2AllBlackBoxTests
- **Note3:** Make sure "eecs4313a2b.EECS4313A2AllBlackBoxTests" is one word.

# 4. White-box Testing with JUnit

The purpose of this section is to give you experience on applying path testing approaches and measuring coverage. Your task here will be to create **another** test suite in JUnit using the white-box testing techniques, produce bug reports (if any), and describe your testing in the written report.

## 4.1 Problem Description

1. Copy the test cases that you have created for the black-box testing approaches and make these test cases as your starting point for your white-box testing cases.
2. Install the **EclEmma** plugin for Eclipse from the Eclipse Marketplace (Help -> Eclipse Marketplace). Measure coverage for the test cases you created for the black-box testing approaches (a.k.a., your "original" white-box testing cases) above. Make note of the coverage measurements as you will need them for your report.
3. Based on the coverage results, add more test cases to your test suite to bring the code coverage of your selected methods as close as possible to 100%.
4. Derive the Control Flow Graph for one of the methods you have tested. Calculate the number of paths in the method (use Boundary Value Analysis in the case of loops). Estimate the percentage of paths that your test suite covers. What other test cases that would not improve statement coverage would you like to add based on this? Implement as many test cases as you can.
5. If you believe that the additional test cases have revealed new bugs, attach bug reports in the written report (see section below for details).

## 4.2 Code Structure and Testing

We will conduct automated testing to mark your testing code. Make sure you place all the white-box testing Java code that you have created in a folder called eecs4313a2w. *There should be no sub-folders inside eecs4313a2w!*

The compilation and the running processes for the white-box test suites are similar as the black-box test suites. Please refer to the steps mentioned in Section 3.2. You should use the following naming conventions for developing and packing your white-box testing code:

- The code package should be called "*eecs4313a2w*"
- Name the test suite as *EECS4313A2AllWhiteBoxTests.java*

# 5. Written Report

For this assignment, you should create a written report (a2.pdf). The report need include one section about the black-box testing and another section about the white-box testing you did above. These two sections must include the following information:

1) For the black-box testing section:
   - Specification of the selected Java methods.
   - Justification of the testing technique chosen, i.e., why is it appropriate for this method.
   - Description of your application of the three testing strategies. Be clear about which test cases you implemented.
   - Evaluation of the test cases derived by the testing technique. Include the screenshots of the test running results. If you had to complement the derived test cases with special value

testing, describe that as well. The marker will not read your code in order to see what you tested. You have to describe it.

- Attaching bug reports if bugs are discovered using your testing methods. You should use the same bug report format as in Assignment 1. **Do not** file these bug reports to the project's bug report system.

2) For the white-box testing section:
   - The statement coverage measurements for your Assignment 2 test suite.
   - A description of the test cases that you added in this assignment to improve statement coverage. The marker will not read your code in order to see what you tested. You have to describe it.
   - The statement coverage measurements for your final submission. Include the screenshots of the test running results and the screenshots of the coverage measurement. If your coverage is not 100%, include a discussion on why that is.
   - The Control Flow Graph you created. Indicate the segments clearly (you will probably need to include the code for this).
   - The path coverage discussion described in section 2 above.
   - Attaching bug reports if bugs are discovered using your testing methods. You should use the same bug report format as in Assignment 1. Do not file these bug reports to the project's bug report system.
   - An appendix with the specification of the methods you are testing (if there are new ones).

Presenting your thought processes in English is an important skill for a software engineer. If you have trouble in writing English, ask someone to proof-read and correct your writing. You can also consult the English as a Second Language Open Learning Centre (http://www.yorku.ca/eslolc) at the York University.

# 6. Submission

You must work in a team with **at least three and at most five** partners. *Failure to form a team or a team with the right sizes will result in loss of marks!* Submit a PDF of your report (a2.pdf) and two test code packages (i.e., eecs4313a2b and eecs4313a2w) electronically. Your report must include the names and the student numbers of all the team members. **Only 1 submission per group, please!** You should also bring a hardcopy of the report before the class.

Navigate to the directory where it contains the code packages and the report. Use the following commands to submit:

- submit 4313 a2 eecs4313a2b
- submit 4313 a2 eecs4313a2w
- submit 4313 a2 a2.pdf