Topics for this Lecture

- Unit testing with JUnit
- How to use Maven

UNIT Testing

- A ***unit test*** is a piece of a code (usually a method) that invokes another piece of code and checks the correctness of some assumptions afterward. If the assumptions turn out to be wrong, the unit test has failed.
  - A "unit" is a method or function.
  - Done during the development (coding) of a program
- Why it is important?
  - Proper unit testing done during the development stage leads to less defect fixing costs and save both time and money in the end.
- How to generate Unit Test Cases
  - Automated and Manually

The JUnit unit testing tool for Java

- JUnit is a framework to unit tests for Java programs
  - Writing test cases in Java
  - Executing test cases (i.e., pass/fail)
- Home page: http://junit.org/junit4/
- Test framework:
  - Test Case = “sequence of operations and inputs + assertions (i.e., expected output)”, and it is written in Java.
  - Test Suite: grouping several test cases together

Features of JUnit

- Easy, convinced, and allows you to write code faster
- Runs in most of IDE's
    - IntelliJ
    - Eclipse
- Can be run and used through command-line
- Automation of tests saves time increases quality

Basic JUnit code constructs

- JUnit annotations:
  - Junit uses annotations to configure methods as test methods

@**BeforeClass** public static void method()	This method is executed once, before the start of all tests.
@**AfterClass** public static void method()	This method is executed once, after all tests have been finished.
@**Before** public void method()	This method is executed before each test.
@**After** public void method()	This method is executed after each test.
@**Test** public void method()	This is the test method to run

Basic JUnit code constructs

- Assert statements:
  - Junit uses asserts to allow you to specify the error message, the expected and the actual result.
  -

fail(message)	This method is executed once, before the start of all tests.
assertTrue([message,] boolean condition)	Checks that the boolean condition is true.
assertFalse([message,] boolean condition)	Checks that the boolean condition is false.
assertEquals([message,] expected, actual)	Tests that two values are the same.
assertNull([message,] object)	Checks that the object is null.

- For example:
  -
  - `int val1 = 5; int val2 = 6; assertFalse(val1 > val2); assertEqual(val1,val2);`
  - `String str4 = "abc"; assertNull(str4);`

Stack Example

```
import java.util.NoSuchElementException;
public class Stack {
     private static int MAX_ELEMENTS=10 ;
     private int[] values= new int[MAX_ELEMENTS];
     private int size=0;
     public Stack() {
     }
     public void push(int x) {
          if (isFull())
               throw new IllegalStateException("Cannot add to full stack");
          else
               values[size++] = (Integer) x;
     }
     private boolean isFull() {
          if (size >= MAX_ELEMENTS)
               return true;
          else
               return false;
     }
     public int pop() {
          if (isEmpty())
               throw new NoSuchElementException("Cannot pop from empty stack");
           else
               return values[--size];
     }
     private boolean isEmpty() {
          if (size == 0)
               return true;
          else
               return false;
     }
     public int top() {
          if (isEmpty())
               throw new NoSuchElementException("Cannot pop from empty stack");
          else
               return values[size - 1];
     }
}
```

```
import org.junit.Test;
import static org.junit.Assert.*;
import java.util.EmptyStackException;

public class StackTest {
  @Test
  public void test0()  throws Throwable  {
      Stack stack0 = new Stack();
      stack0.push((-16));
      stack0.push((-16));
      stack0.push((-16));
      Try {

        stack0.push((-16));

      } catch(EmptyStackException e) {
       }
  }
  @Test
  public void test1()  throws Throwable  {
      Stack stack0 = new Stack();
      stack0.push((-16));
      assertEquals(-16, stack0.top());
      int int0 = stack0.pop();
      assertEquals((-16), int0);
  }
  @Test
   public void test2()  throws Throwable  {
      Stack stack0 = new Stack();
      try {
        stack0.pop();
        fail("Expecting
exception:EmptyStackException");
      } catch(EmptyStackException e) {
       }
   }

   @Test
   public void test3()  throws Throwable  {
      Stack stack0 = new Stack();
      try {
        stack0.top();
      } catch(EmptyStackException e) {
        }
   }
}
```

Using Maven in CS362

- Apache Maven is a software-project-management tool that:
  - Includes the functionality of a build system (that compiles a complicated multi-file sources)
  - Manages dependencies nicely (i.e., a complex software system will frequently depend on the installation of various other libraries)
  - A ***Project Object Model*** or ***POM*** is the fundamental unit of work in Maven. It is an XML file (pom.xml) that contains information about the project and configuration details used by Maven to build the project.
  - Maven has become widely used for open-source software projects.
- The Maven tool is available for download (https://maven.apache.org/). It is recommend that you add the <untarred maven directory>/bin to your PATH variable. If you don't, you'll need to substitute mvn with <untarred maven directory>/bin/mvn

Maven - Command Line Options

- Maven is a command-line tool, so it is run from the Linux (or even Windows) command prompt:
  - Create a Project from Maven Template
    - mvn archetype:generate
- With a project that has been set up using Maven, you might use the following commands (you must run mvn command from the directory containing the pom.xml) :
  - mvn compile - to compile your code
  - mvn test - to run your unit tests
  - mvn package - to create a JAR file that you can run
  - mvn eclipse:eclipse -to generate the following eclipse configuration files: .project and .classpath files

Maven + how to setting it up

- The Maven tool is available for download https://maven.apache.org/
- After opining the page choose download link (left side), and download of the files, you can chhose the binary Zip file Binary zip apache-maven-3.3.9-bin.zip
- After download the zip file and extract it you will see the **apache-maven-3.3.9** folder that contains all the required files to run Maven
- You need to setup and add the **apache-maven-3.3.9**/bin to your PATH variable where you extreacted Maven. If you don't, you'll need to substitute mvn with **apache-maven-3.3.9**/bin/mvn
- To test out that Maven is working fine, you need to execute the simplest command for Maven, c:>**mvn -version**. If you correctly setup Maven in you

```
$> or c:\>  mvn -version (in windows/linux)
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T08:41:47-08:00)
Maven home: /usr/local/apps/apache-maven-3.3.9
Java version: 1.8.0_121, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.121-0.b13.el7_3.x86_64/jre
Default locale: en_US, platform encoding: UTF-8
```

Maven + how to use Maven to your project

- Create a directory called dominion and go to the directory dominion
- While your in the directory dominion:
  - > **mvn archetype:generate**
  - Choose a number or apply filter …. : press enter
  - Choose a number: 6: enter 6 and press enter (Maven version)
  - Define value for property 'groupId': **edu.osu.cs362** < this is the package name you are going to use>
  - Define value for property 'artifactId': : **hw1** <this is the name of the jar file>
  - Define value for property 'version': 1.0-SNAPSHOT: : press enter
  - package: edu.osu.cs362

    Y: : enter Y and press enter
  - Maven is finished and when to see what Maven has done so far!

Oregon State UNIVERSITY

Maven + how to use Maven to your project

- $>ls or dir to see the folder **hw1** that was created by Maven
- $> **cd hw1**
- You should see a src folder and pom.xml file
- The src folder you find main and test folders
  - The main folder is used to keep/store Java files for your project
  - The test folder is used to keep/store all the Junit test cases
- The pom.xml file contains all the information and dependencies that you project is required.
  - There is only one dependency is already defined by Maven which is the **Junit**
  - We need to change the version of Junit from 3.8.1 to 4.12

Maven + how to use Maven to your project

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>cs362.001</groupId>
  <artifactId>hw1</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>Dominion</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Maven + code coverage example

- **Note**: **we always run Maven commands in the hw1 directory , i.e., where the pol.xml file is**
- You need to compile your project
  - **mvn compile**
    - Note: if you run this for the first time, it might take a while to finish!
- Build the Project
  - **mvn package**
  - Note: if you want to run the main file
    - java -cp target/hw1-1.0-SNAPSHOT.jar org.cs362.YourMainFileName
    - Java -cp ./target/classes/ edu.osu.cs362.YourMainFileName
- **Run your JUnit test cases**
  - **mvn test**
- **To import the project to Eclipse**
  - **mvn eclipse:eclipse**
    - **Then open eclipse and select File, Import and General, Existing projects to workspace, go to the Dominion folder and press OK**

Oregon State UNIVERSITY

Useful youtube links

- How to install Java JDK on Windows 10 (with JAVA_HOME)
- Getting Started With Eclipse
- Java - JUnit testing in Eclipse

References:
http://www.vogella.com/tutorials/JUnit/article.html
The Art of Unit Testing: With Examples in C# Book by Roy Osherove