JUNIT – Java Unit Testing

Outline

1) What is JUNIT?
2) Refactoring code
    a. BadRobot.java
    b. GoodRobot.java
3) Writing JUNIT tests
    a. TestGoodRobot.java
4) Installing JUNIT into Eclipse

What is JUNIT?

1) JUNIT is a plugin for Eclipse
    a. JUNIT stands for Java Unit Testing
    b. JUNIT enables you to write tests to test your methods.
2) A "unit" is a chunk of code you want to test, that is – **a method**
3) Assuming you have a class named "Robot", you create a "test class" call TestRobot
    a. Each test starts with annotation of @Test

```
class Robot {
   public int calculateAngle() {
        return(50);
   }
} // end of class Robot


class TestRobot {
   Robot myRobot = new Robot();

   @Test
   Public void testAngle() {
        assertEquals(50, myRobot.calcualateAngle());
   }
} // end of class TestRobot
```

Refactoring code

Refactor code – restructure existing computer code, WITHOUT changing the external behavior!

**** The goal is to improve the CODE ITSELF, without changing what the program does at all**

This must be done in tiny steps. Change code, test to ensure no changes, REPEAT!

Advantages

1. Improve code **readability**
2. Reduce code **complexity**
3. Improve source code **maintainability** (easier to make changes, without fear of breaking things!)
4. Improve **extensibility** (the ability to easily add more code, and extend functionality) without breaking things!)

Checklist of things to refactor

1. Add more and better comments
2. Remove "magic numbers" from code by using CONSTANTS
    a. Change: if (xbox.getButton() == 3) to
        if (xbox.getButton() == LEFT_SHOULDER_BUTTON)
3. Break code into small methods (strive for methods about one screen long) – makes code simpler
    a. Each method should do **something simple**, that is very well defined.
    b. Don't keep adding code to methods (which makes them more complicated). Instead, add more methods!
4. Improve complicated logical expressions
5. Avoid having deeply nested structures

```
if (x < 6) {
   if (x ==7) {
       while (sensorReading < 50) {
           if (level == 7) {
```

Instead have:

```
if (x < 6) {
   if (x == 7) {
       processLightLevel(sensorReading)
   }
}

public static processLightLevel(int sensorReading) {
   while (sensorReading < 50) {
       if (level == 7) {
           …
}
```

Refactoring session

BEFORE: BadRobot.java
AFTER: GoodRobot.java

Writing JUNIT Tests- TestGoodRobot.java

JUNIT is a framework that allow you to easily create tests (independent from your project code)
Each test starts with the annotation: @Test

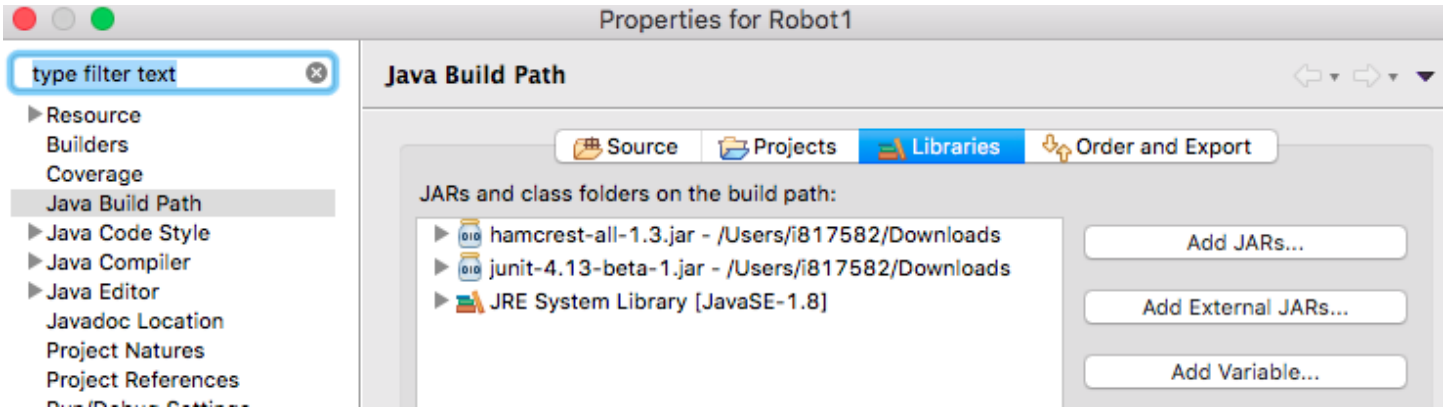
Tips on writing methods:

1) JUNIT assumes you have methods to test!!
    a. A method should be simple (fewer lines of code is best)
    b. A method should perform a single task (keep it simple)
    c. It is best for methods to return a value – so you can JUNIT test them. **Avoid "void" methods**
2) Method names should start with a verb. Examples: turnWheel(), getTurnValue(), calculateAngle(), etc.
3) JUNIT enabled TDD – Test Driven Development
    a. This means you can write the test FIRST, implement later!!
    b. See example of testTurnWheelReallySlow()

```
@Test
public void testLightDesc() {
  assertEquals("Dark", goodRobot.getLevelDescription(1));
}
```

Installing JUNIT into Eclipse

1) High level steps to install JUNIT (use may get slightly different versions)
    a. Download junit-4-13-beta-1.jar
        i. https://search.maven.org/search?q=junit
    b. Download hamcrest-all-1.3.jar
        i. https://search.maven.org/search?q=a:hamcrest-all
2) Use the links below for exact instructions
    a. Right-click Project | Properties | Java Build Path | Libraries | Add External JARs…
    b.



References

Primary JUNIT website: https://junit.org/junit5/
Install JUNIT link: https://www.guru99.com/download-installation-junit.html