TEST

MATERIALS

https://www.lynda.com/Developer-Programming-Foundations-tutorials/Welcome/124398/137955-4.html
http://junit.org/junit4/
http://junit.org/junit5/

PLAN

Terms / Concepts
- Aim of testing
- Difference between white box and black box testing
- Different types of tests (Unit / Integration / System)
- Implications of designing tests
- Determination of equivalence classes and boundary values
- Purpose of automated testing
- Performing automated testing

JUnit
- Creating JUnit tests
- Running JUnit tests

Unit testing
- Unit testing classes

Code coverage
- Improving code coverage

Jacoco
- Using Jacoco

Integration testing
System testing
- JMeter / Selenium

Database testing
Mocking

SUBJECTS

ASPECTS OF TEST / TESTING CONCEPTS

Software testing is an investigation conducted to provide stakeholders with information about the quality of the code and product giving an objective, independent view of the software.
- Demonstrate behavior / Detect problems
- Respond correctly / Find bugs / Satisfy requirements

Determine the correctness of software under the assumption of some specific hypotheses
- Inputs / Outputs / Expectations

Black box test / White box test

White-box testing focuses on internal structures or workings
- Internal structure/ design/ implementation is known to the tester

Black-box testing focuses on functionality
- Internal structure/ design/ implementation is not known to the tester

Many different types of tests

Unit tests
- Smallest testable parts
- Testing certain functions and areas
- Verification, at the level of individual units and their methods and classes, that code works and continues to work as expected
- A suite of tests can be run at any time during development to continually verify the code quality

Integration tests

- Across layers
- Integration tests are focused on groups of individually tested units and their collective interaction
- Individual modules are combined and tested as a group after unit testing

System tests

- Evaluate the whole integrated system and its compliance with specified requirements
- Performed on the entire system in the context of the functional and non-functional requirements
  - Functional requirements are concerned with WHAT a software system should do
  - Non-functional requirements revolve around HOW a software system should do something

User interface tests

- Identify the presence of defects in a product by using the graphical user interface
- Does the user interface work correctly?

User tests

- Future users try to use a preliminary version of the system to see if they can solve their tasks

Acceptance tests

- Verifying a solution works for the user
- A contract between the developers and the product owner on when a user story is implemented
- Beta testing

Test design / Constructing tests

What should be tested?
How to do testing?

Logic / Rules
Setters / Getters
Each test will reveal a potential error

To test it is needed to know what the expected behavior is

- Return values
- Thrown exceptions
- Changes to objects and external components
- Calls to other objects

Enough input data sets to make sure that:

- Methods have been called
- Both true and false branches have been executed in if statements
- Branches of switch statements have been executed
- Loops have been executed zero, one, and more times

For every input data set, the expected output must also be specified

Equivalence classes

Group test case values into classes and select a value from each class, since all values in a class are expected to behave exactly the same way

Boundary values

Test the values on the edge of each test case value class, since it is expected that problems occur around edges

Automated testing

The use of special software, separate from the software being tested, to control the execution of tests and the comparison of actual outcomes with predicted outcomes.
Some software testing tasks can be laborious and time-consuming to do manually.
Test automation can automate repetitive tasks that would be difficult to do manually.

JUnit

Simple framework to write repeatable tests
One of a family of unit testing frameworks collectively known as xUnit
JUnit API includes various classes and annotations to write test cases

Maven

Dependency
mvn test
Build

CODE COVERAGE

Test coverage

Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions
Testing cannot identify all the defects within software
Possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources

Measures to which extent code has been tested

The development community is a bit divided on automated software testing – Some people think you should have tests for 100% of all of your code, some believe that 80% is sufficient, some 50%, and some are content with 20%.

JACOCO

Netbeans plugin
Maven dependency

pom.xml

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.7.9</version>
  <executions>
    <execution>
      <id>default-prepare-agent</id>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <execution>
      <id>default-report</id>
      <phase>prepare-package</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Target/Site
Show report