Java Day 4

1. Unit Testing

- A unit test is a piece of code that executes a specific functionality in the code that is being tested

- The percentage of code which is tested by unit tests is called test coverage

- A test fixture is a fixed state of the software under test, used as a baseline for running tests.

2. Functional and integration tests

- An integration test has the target to test the behavior of a component or the integration between a set of components.

- Functional test is used as a synonym for integration test.

- These tests allow you to translate user stories into a test suite, i.e. the test would resemble a expected user interaction with the application

3. Creating a unit test case

- Go to File -> New -> JUnit Test case

and create the following test

```
@Test
public void testMultiply()
{
   MyClass tester = new MyClass();
   assertEquals(“Testing multiply”, 50, tester.multiply(10,5));
}
```

7. Running the test

- Go to Run -> Run As -> JUnit Test to run

- JUnit assumes that all test methods can be executed in an arbitrary order

- Therefore, tests should not depend on other tests

8. Assert Statements

fail(String)

Let the method fail. Might be used to check that a certain part of the code is not reached. Or to have a failing test before the test code is implemented. The String parameter is optional.

assertTrue([message], boolean condition) Checks that the boolean condition is true.

assertFalse([message], boolean condition) Checks that the boolean condition is false.

assertEquals([String message], expected, actual) Tests that two values are the same. Note: for arrays the reference is checked not the content of the arrays.

assertEquals([String message], expected, actual, tolerance) Test that float or double values match. The tolerance is the number of decimals which must be the same.

assertNull([message], object) Checks that the object is null.

assertNotNull([message], object) Checks that the object is not null.

assertSame([String], expected, actual) Checks that both variables refer to the same object.

assertNotSame([String], expected, actual) Checks that both variables refer to different objects.

9. Creating a JUnit test suite

- Use the following code to create a test suite

```
@RunWith(Suite.class)
@SuiteClasses({MathEngineTest.class,
PhysicsEngineTest.class})
public class AllTests
{


}
```