Exercice

Develop a class with a method that counts the lines that are not empty nor just contain comments, creating tests for each new code increment.

As soon as you have a idea to start the problem, create a test, implement this first part and test it. For each new increment, do the same.

Don't think too much beforehand, just try and test, but **keep all tests** and improve/refactor your code

Use pair programming

```
/*****
* This is a test program with 5 lines of code
//*****//***/// Slightly pathological comment ending...

public class Hello {
    public static void main(String [] args) { // comment
    // Say hello
    System./*wait*/out./*for*/println/*it*/(“Hello/*”);
    }
}
```

TDD: Test Driven Development

– Write a test before any real code writing

On Eclipse

– JUnit 1.3.8: Java 1.4

– JUnit 4: Java 5.0 with annotations

TDD: Test Driven Development

Documentation Eclipse:

- Help **>** Help Contents **>** Java Development User Guide **>** Getting Started **>** Basic Tutorial **>** Writing and running JUnit tests (1.3.8)
- Help **>** Help Contents **>** Java Development User Guide **>** What's new **>** JUnit Toolings (4)
- (http://www.junit.org) → JUnit 4.0 in 10 minutes

TDD

- Create a Java project as usual
- Project **>** Build Path **>** Add Libraries… **>** JUnit 4
- Right-click the class that will be under test (possibly with an empty method) **>** New **>** JUnit Test Case (*creates a case of the version entered into the build path*)
- Define a package that is going to contain the tests

TDD

Edit the test class:

- Create an instantiation of the Class under test in the @BeforeClass method (once for all methods) or in the @Before method (once for every method)
- Put one test (CTRL-space) per test method. If you put several tests, they will not be identified separately when there is an error.
- The message in an assertion is displayed if there is an error
- It is possible to check the kind of Exception returned by the method under test (see doc)

TDD

```
package unitTests;

import static org.junit.*;

public class ComputationTest1 {
        static example.Computation computation = null;


/**
 * This method is executed only once before all tests
 *  Here, it creates an environment to test the modules
 */
        @BeforeClass
        public static void setUp() throws Exception {
                computation = new example.Computation();
                computation.setResult(0);
        }
```

TDD

```
package unitTests;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.junit.BeforeClass;

public class ComputationTest1 {
    static example.Computation computation = null;

/**
 * The following method is repeated before every test
 */
    @Before
    public void setUpRepeated() throws Exception {
        computation = new example.Computation();
        computation.setResult(0);
    }
 . . .
```

TDD

```
. . .
@After
public void tearDown() throws Exception {
}
@Test
public void testComputation() {
}
@Test
public void testAdd1() {
        computation.add(7);
        // To show an error in a test, the next check is incorrect !
        assertTrue("Error 1 of computation", computation.getResult()==6);
}
@Test
public void testAdd2() {
        computation.add(-12);
        assertTrue("Error 2 of computation", computation.getResult()==-5);
}
}
```

TDD: running the tests

- Select some test classes or the test package
- Right click the selection > Run As > JUnit Test
- The result is displayed in the view JUnit (same window as the Package Explorer)