Approach for
Unit testing with the help of JUnit ...

Satish Mishra

mishra@informatik.hu-berlin.de

This session

- Testing concepts
  - Unit testing
- Testing tools
  - JUnit
- Practical use of tools
  - Examples
- Writing unit test cases
  - With the help of you all
- Discussions
  - Always welcome

Status of assignments

- **Doubts ?**
- **Progress ! !**
- **Results ??**

Unit vs Functional tests

- **Unit tests::**
  - Unit tests are written from a programmer's perspective. They ensure that a particular method of a class successfully performs a set of specific tasks. Each test confirms that a method produces the expected output when given a known input.

- **Functional tests::**
  - Functional tests are written from a user's perspective. These tests confirm that the system does what users are expecting it to .

Boundries between Unit and Functional tests

- If a unit test crosses class boundaries, it might be a functional test.
- If a unit test is becoming very complicated, it might be a functional test.
- If a unit test is a valid test but it has to change continually to handle different user permutations, it might be a functional test.
- If a unit test is harder to write than the code it is testing, it might be a functional test.

Top reasons to write unit tests

- Tests reduce bugs in new features
- Tests reduce bugs in existing features
- Tests are good documentation
- Tests reduce the cost of change
- Tests improve design
- Tests allow refactoring
- Tests constrain features
- Testing forces you to slow down and think
- Testing makes development faster
- Tests reduce fear

Example: 1. Hello World Program

- Write test case for a ‘Hello World’ program which is written in java.

Hello world Unit Test

- Instance of HelloWorld.java should not be null
- Output should be “Hello World“.

Hello world java code

```
class HelloWorld {
 /**     * Print "Hello World"     */
 void sayHello() {
    System.out.println("Hello World");
 }
 /**     * Test     */
  public static void main( String[] args ) {       HelloWorld world =
    new HelloWorld();       world.sayHello();
 }
}
```

*JUnit tests for **HelloWorld***

```
public class HelloWorldTest extends junit.framework.TestCase {
   Hello HelloWorld ;

   public HelloWorldTest () { }   // default constructor

   protected void setUp() {   // creates a (simple) test fixture
       Hello = new Helloworld();
   }

   protected void tearDown() { } // no resources to release
```

*JUnit tests for **HelloWorld** …*

```
  testSayHello() {
    assert( Hello!=null );
   assertEquals("Hello World",Hello.sayHello());    }
}   // End from last slide
```

Example:2

Write a java program for addition and multiplication of square Matrix.

Detail:

A + **B** = **C** (Simple addition)

A x **B** = **C**

[**AB**]ij = **SIGMA** [**A**]ik[**B**]kj

where A and B are square Matrix and C is resultant Matrix

Matrix Program Unit Tests::

We all will write together ..

Matrix Program Java Code::

- Suppose i wrote this java program according to suggested test cases.
- Now we will add the test case in my program
- Try with JUnit

Result

- We will see with the help of tool

When ??

!! Now !!

Example 3 Employee Database

- Write a java class for reading a Employee database which is in text file with specific format and then storing the text file data into the database. Program should use odbc for the connetion.

Detail:: Text file contains the data in the sequence of

Employee Id, Department Id,Name,Date of birth, Date of Joining, Salary, Position, Extension, Location

Employee Database Program Unit Tests::

We all will write together ....

Employee Database Program Java Code::

- Suppose java code we got from our colleague who is also working with us
- Now we will add the test case in this program
- Try with JUnit

Result

- We will see with the help of tool

When ??

!! Now !!

The End

Thank You