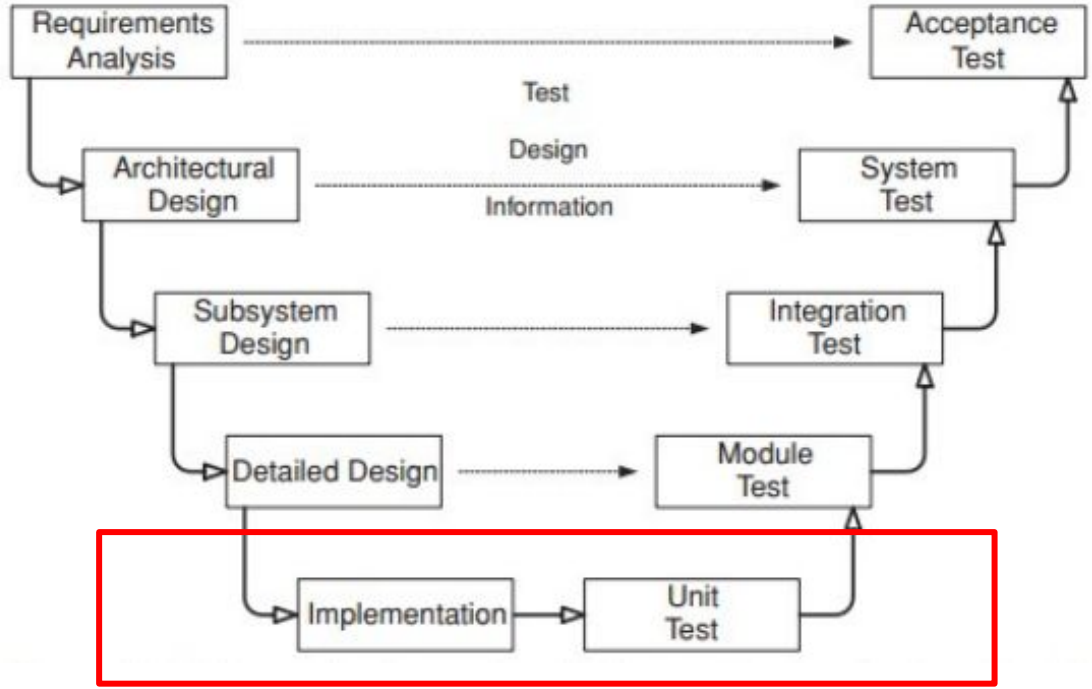
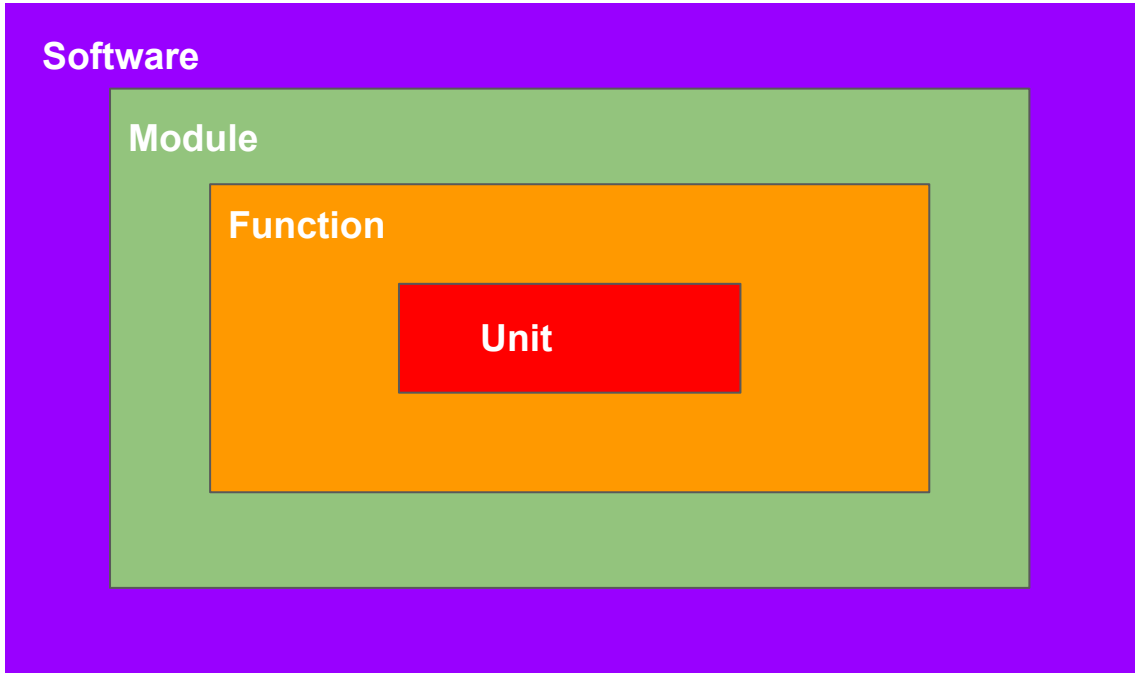
Introduction to Unit Testing Using JUnit

STT Terpadu Nurul Fikri 2019

V-Model Testing Activities



Unit Testing



Unit Testing (2)

1. Sebuah aktifitas testing yang dilakukan pada level terkecil software (unit).
2. Dilakukan oleh programmer / developer (bukan software tester).
3. Termasuk ke dalam **White-box Testing** & **Dynamic Testing.**
4. Fokus hanya pada menguji kode yang kita buat.
5. Bersifat isolated (tidak ada hubungan external code / library).
6. Pengujian dilakukan dengan cara membandingkan antara **expected result** dengan **actual result.**
7. Juga berperan sebagai **living documentation.**

Unit Testing (3)

```
public class MySetTest
{
    @Test
     public void testAddElement()
     {
         MySet set = new MySet();
         set.add("Budi");
         assertEquals(1, set.size());
     }
}
```

- **“Assertion”** dalam unit testing ditujukan untuk mengecek nilai **expected result** dengan **actual result**.
- Pada contoh kode di samping, assertEquals(1, set.size()) merupakan sebuah assertion yang ditujukan untuk **memastikan bahwa size dari MySet bernilai 1**.

Unit Testing (4)

```java
public class MySet
{
    private String[] items = new String[5];
    private int index = 0;
    private int size = 0;

    public int getSize()
    {
        return size;
    }

    public void add(String item)
    {
        if(this.isExist(item)) return;
        if(this.isFull()) this.expand();

        items[index++] = item;
        size++;
    }
}
```

```java
public class MySetTest
{
    @Test
    public void testAddNewElement()
    {
        MySet mySet = new MySet();
        mySet.add("Uwais");
        assertEquals(1, mySet.getSize());
    }
}
```

JUnit

- JUnit merupakan sebuah library unit testing berbasis JAVA
- Versi terbaru: JUnit 5
- JUnit 5 sangat jauh berbeda dengan JUnit 4
- Untuk keperluan kuliah ini, kita akan menggunakan JUnit 4
- Dapat diunduh melalui website resmi JUnit: **https://junit.org/junit4/**

JUnit (2)

Download dan Instalasi:

1. Menggunakan File Jar
   a. JUnit: https://search.maven.org/search?q=g:junit%20AND%20a:junit
   b. Hamcrest: https://search.maven.org/artifact/org.hamcrest/hamcrest-core/1.3/jar
2. Menggunakan Maven

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

JUnit (3)

Cara Menggunakan:

```
//compile MySet.java
javac MySet.java

//compile MySetTest.java
javac -cp .;junit-4.12.jar MySetTest.java

//menjalankan unit testing
java -cp .;junit-4.12.jar;hamcrest-core-1.3.jar
org.junit.runner.JUnitCore MySetTest
```

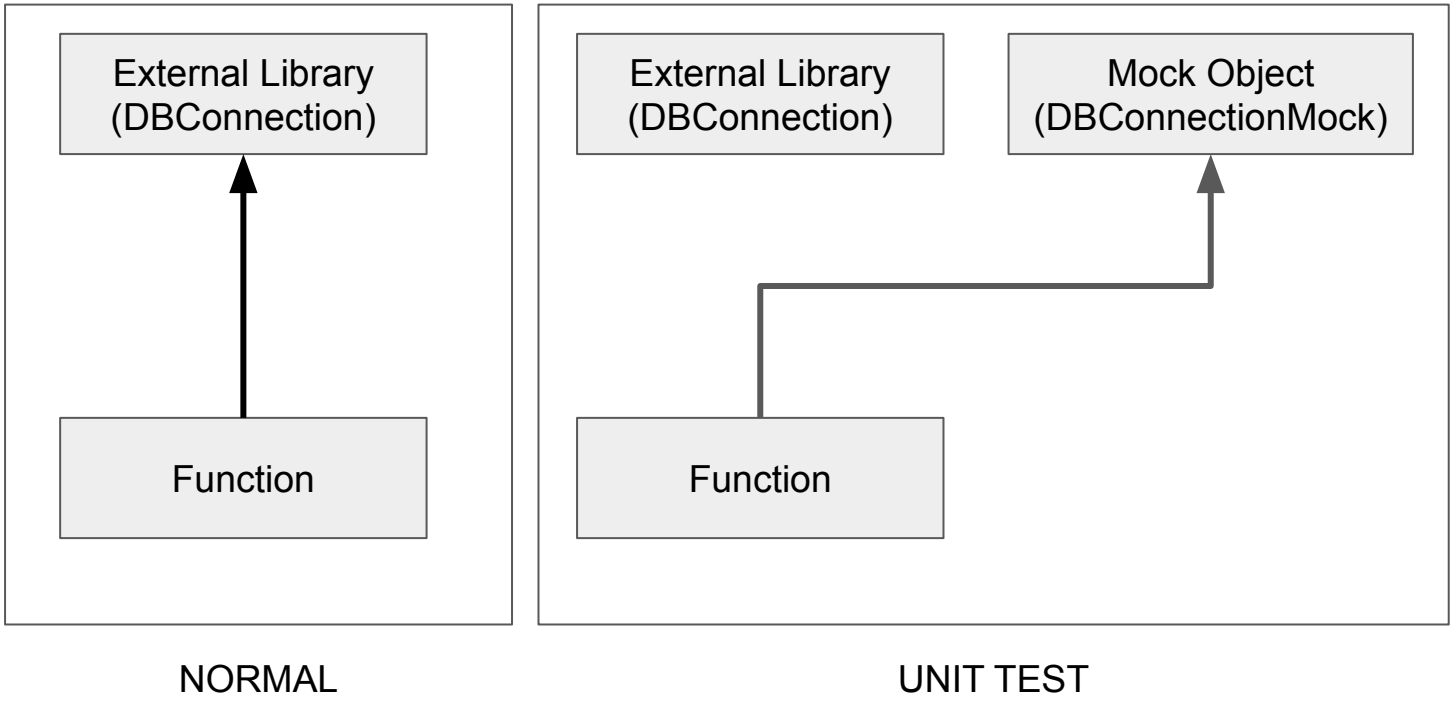
Assertion

```
assertEquals("failure - strings are not equal", "text", "text");

assertFalse("failure - should be false", false);

assertNull("should be null", null);

assertNotNull("should not be null", new Object());

assertNotSame("should not be same Object", new Object(), new Object());

assertArrayEquals("failure - byte arrays not same", expected, actual);
```

Mock Object

- Sebuah software umumnya berkaitan erat dengan software yang lain seperti misalnya penggunaan library untuk Database, Http / Network, String manipulation, Logging, dll.
- Dikarenakan Unit Testing bersifat isolated, maka seharusnya komunikasi dengan external library harus dibatasi dikarenakan kita tidak control penuh terhadap kode library tersebut sehingga tidak bisa kita masukan sebagai bagian dari unit testing yang kita buat.
- **Mock Object** merupakan object pengganti dari object sesungguhnya yang behaviour kita control sesuai dengan yang kita inginkan.

Mock Object (2)



Mock Object (3)

```
public class MockExample
{
    private DBConnection conn;

    public MockExample(DBConnection conn)
    {
        this.conn = conn;
    }

    public User getUser(Integer id)
    {
        User user = conn.findUserById(id);
        return user;
    }
}
```

Mock Object (4)

```
public class MockExample
{

    public void updateUser(Integer id, String name)
    {
        User user = this.getUser(id);
        user.setName(name);
        user.save();
    }
}
```

Normal Call

Mock Object (5)

```
public class TestMockExample
{
     @Test
     public testGivenIdShouldReturnUserWithCorrectId()
     {
          DBConnection mockDB = mock(DBConnection.class);
          when(mockDB.findUserById(1)).thenReturn(new User(1, "User Object"));

          MockExample app = new MockExample(mockDB);
          User user = app.getUser(1);
          assertEquals(1, user.getId());
     }
}
```

Using Mock Object

Mock Object (6)

Library yang dapat digunakan:

Mockito (JAVA) : https://site.mockito.org/

Using gradle:

```
repositories { jcenter() }

dependencies { testCompile "org.mockito:mockito-core:2.+" }
```

Mock Object (7)

```java
import static org.mockito.Mockito.*;

// mock creation
List mockedList = mock(List.class);

// using mock object - it does not throw any "unexpected interaction" exception
mockedList.add("one");
mockedList.clear();

// selective, explicit, highly readable verification
verify(mockedList).add("one");
verify(mockedList).clear();
```