MODUL 3 - jUnit

Testing Tools

TIM RKPPL
TEKNIK INFORMATIKA

UNIVERSITAS PASUNDAN
BANDUNG 2016

Pengertian Pengujian

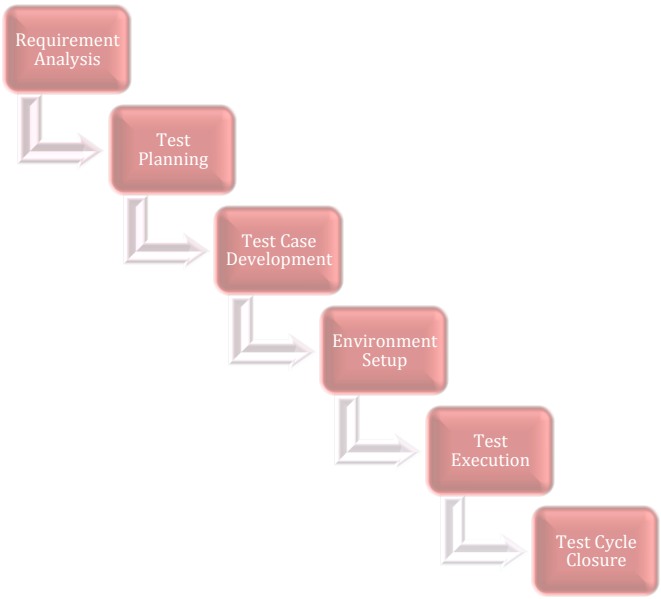
Pengujian perangkat lunak (bahasa Inggris: *software testing*) merupakan suatu investigasi yang dilakukan untuk mendapatkan informasi mengenai kualitas dari produk atau layanan yang sedang diuji (*under test)*.[1] Pengujian perangkat lunak juga memberikan pandangan mengenai perangkat lunak secara obyektif dan independen, yang bermanfaat dalam operasional bisnis untuk memahami tingkat risiko pada implementasinya. Tidak hanya itu pengujian perangkat lunak digunakan untuk mendeteksi perbedaan antara kondisi yang ada dengan kondisi yang diinginkan (*defects/errors/bugs*) dan mengevaluasi fitur-fitur dari entitas perangkat lunak.

Pengujian perangkat lunak dapat dinyatakan sebagai proses validasi dan verifikasi bahwa sebuah program / aplikasi / produk:

1. Memenuhi kebutuhan (*requirement*) yang mendasari perancangan dan pengembangan perangkat lunak tersebut;
2. Berjalan sesuai dengan yang diharapkan;
3. Dapat diterapkan menggunakan karakteristik yang sama;
4. Memenuhi kebutuhan semua pihak yang berkepentingan

[Sumber : wikipedia]

Dibawah ini adalah siklus hidup pengujian perangkat lunak (STLC) :



Level Testing Untuk Perangkat Lunak yang Konvensional

Unit Testing

- Metode verifikasi dan validasi software yang dilakukan oleh developer untuk memastikan bahwa unit individual dari suatu source code berfungsi sebagaimana dimaksudkan

Integration Testing

- Suatu tahapan proses pengujian aplikasi setelah fase unit testing dan sebelum system testing.
- Berfokus pada desain dan konstruksi arsitektur perangkat lunak.

Validation Testing

- Mengecek kebenaran perangkat lunak, apakah proses yang telah ditulis dalam SRS sudah terpenuhi ? atau belum ?

System Testing

- Pengujian perilaku produk perangkat lunak yang lengkap dan terintegrasi berdasarkan dokumen Software Requirement Specification (SRS)
- Fokus utama dari pengujian ini adalah untuk mengevaluasi kebutuhan Bisnis / Fungsional / End User

Pada modul ini kita akan membahas tentang Unit Testing !

Unit Testing

- Fokus pengujian pada fungsi atau modul perangkat lunak,
- Berfokus pada logika pengolahan dan struktur data internal,
- Menyederhanakan ketika modul-modul dirancang dengan kohesi tinggi
    - Mengurangi jumlah kasus uji,
    - Memungkinkan kesalahan akan lebih mudah diprediksi dan ditemukan,
- Berfokus pada modul-modul kritis dan kompleksitas cyclomatic tinggi, ketika melakukan pengujian dengan sumber daya terbatas.

JUnit

Junit merupakan framework pengujian untuk bahasa pemrograman Java yang bersifat open source. Framework pengujian seperti Junit dapat digunakan untuk pengujian modul program berulang kali, khususnya setelah ada perubahan pada bagian modul tersebut.

Junit memiliki beberapa fitur, diantaranya :

- Assertion, dapat digunakan untuk pengujian berdasarkan hasil yang diharapkan.
- Test Fixtures, untuk sharing data yang akan diuji.
- Test Suite, untuk memudahkan pengaturan dan running test.
- Dapat menampilkan grafik dan textual untuk test runner.

JUnit code constructs

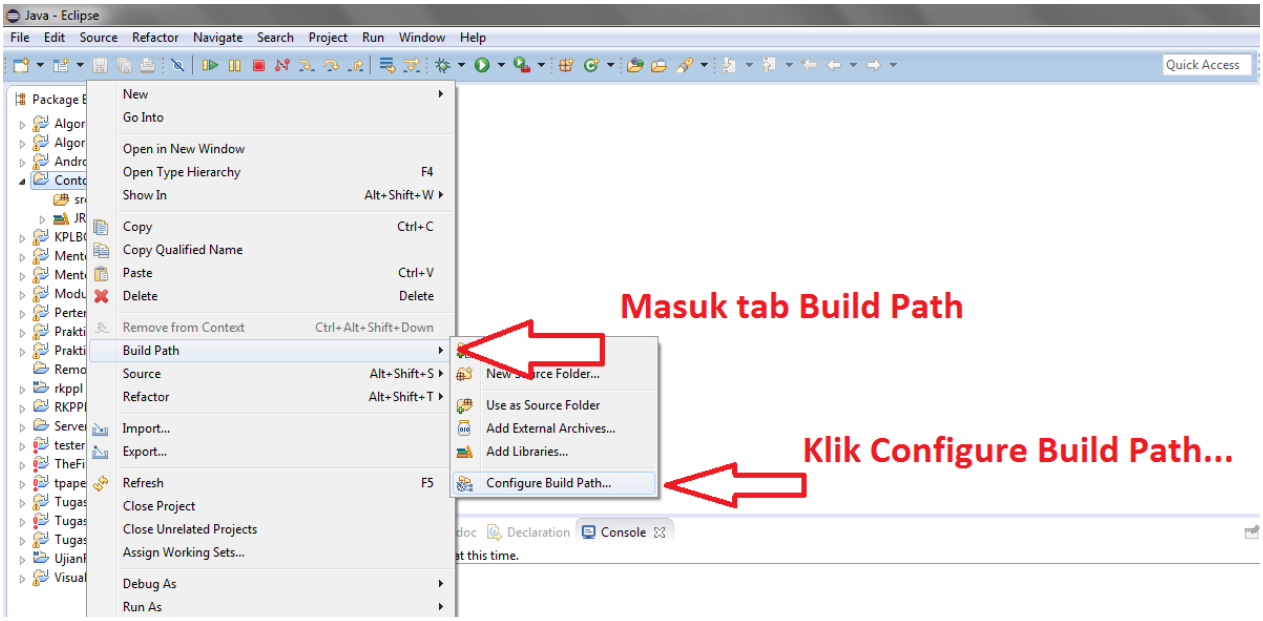
Annotation	Description
@Test public void method()	The @Test annotation identifies a method as a test method.
@Test (expected = Exception.class)	Fails if the method does not throw the named exception.
@Test(timeout=100)	Fails if the method takes longer than 100 milliseconds.
@Before public void method()	This method is executed before each test. It is used to prepare the test environment (e.g., read input data, initialize the class).
@After public void method()	This method is executed after each test. It is used to cleanup the test environment (e.g., delete temporary data, restore defaults). It can also save memory by cleaning up expensive memory structures.
@BeforeClass public static void method()	This method is executed once, before the start of all tests. It is used to perform time intensive activities, for example, to connect to a database. Methods marked with this annotation need to be defined as static to work with JUnit.
@AfterClass public static void method()	This method is executed once, after all tests have been finished. It is used to perform clean-up activities, for example, to disconnect from a database. Methods annotated with this annotation need to be defined as static to work with JUnit.
@Ignore	Ignores the test method. This is useful when the underlying code has been changed and the test case has not yet been adapted. Or if the execution time of this test is too long to be included.

Assert Statements

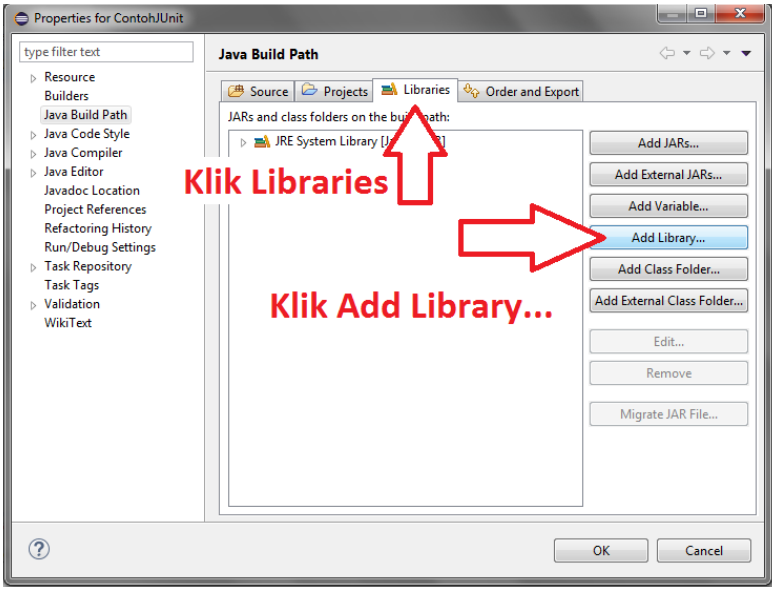
Statement	Description
fail(String)	Let the method fail. Might be used to check that a certain part of the code is not reached or to have a failing test before the test code is implemented. The String parameter is optional.
assertTrue([message], boolean condition)	Checks that the boolean condition is 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.

Cara menggunakan JUnit

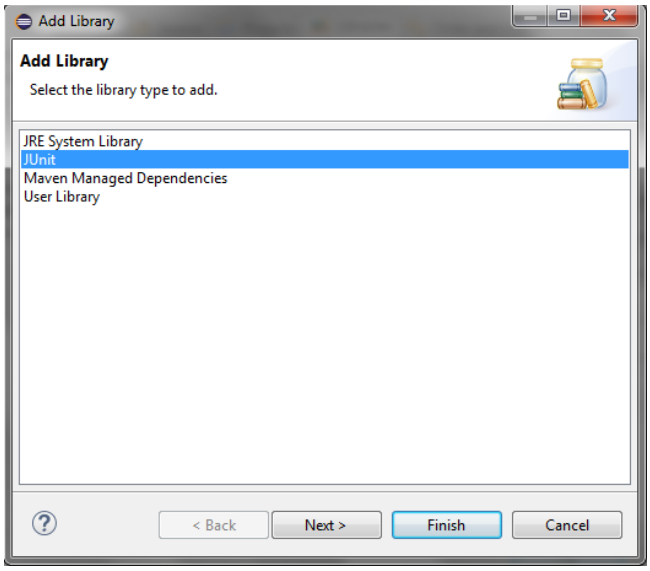
1. Jika menggunakan Eclipse : Kepler (Atau versi terbaru)
    - Buat project baru
    - Klik kanan project kemudian sorot Build Path lalu klik Configure Build Path



- Klik tab library dan klik Add Library



- Pilih JUnit lalu klik Next



- Pilih JUnit versi 4 lalu klik Finish
- Lihat pada properties for [nama_project] yang kalian buat, library JUnit sudah berhasil ditambahkan.

2. Jika menggunakan Eclipse : Juno (Atau versi lama)
    - Download library JUnit di : https://github.com/junit-team/junit/wiki/Download-and-Install
    - Lakukan langkah yang sama seperti diatas lalu hingga masuk ke menu Configure Build Path
    - Klik Library lalu klik Add External JARs...
    - Pilih lokasi dimana library JUnit didownload (disimpan)
    - Klik open, lalu Finish

SIMULASI

Buatlah kelas mahasiswa seperti berikut :

```
package org.ifunpas.praktikum.rkppl.percobaan1;

public class Mahasiswa {

    private String nama;
    private String nrp;
    private int umur;

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNrp() {
        return nrp;
    }

    public void setNrp(String nrp) {
        this.nrp = nrp;
    }

    public int getUmur() {
        return umur;
    }

    public void setUmur(int umur) {
        this.umur = umur;
    }

}
```

Setelah itu kita mulai pengujian dengan menggunakan JUnit !

- Buat kelas MahasiswaTest seperti berikut :

```
package org.ifunpas.praktikum.rkppl.percobaan1test;

import org.ifunpas.praktikum.rkppl.percobaan1.Mahasiswa;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class MahasiswaTest {

    private static Mahasiswa m;

    @Before
    public void awalTest() {
        m = new Mahasiswa();
        System.out.println("Mengawali Testing");
    }

    @Test
    public void test1() {
        m.setNrp("123040001");
        assertNotNull("Seharusnya Tidak Null", m.getNrp());
    }

    @Test
    public void test2() {
        m.setNama(null);
        assertNull("Seharusnya Null", m.getNama());
    }

    @After
    public void akhirTest() {
        System.out.println("Mengakhiri Testing");
    }

}
```

- **Coba Running kelas tersebut sebagai JUnit (Alt+Shift+X, T), dan amati apa yang terjadi.**
- **Lakukan perubahan pada method test2 tambahkan value "Nama kalian" pada setNama, Amati apa yang terjadi.**

Tambahkan kode ini kedalam kelas Mahasiswa :

```
public boolean cekUmur(int umur){
    boolean toReturn = false;
    if (umur >= 0 && umur <= 100) {
        toReturn = true;
    }

    return toReturn;
}
```

Tambahkan pula kode ini ke kelas MahasiswaTest :

```
@Test
public void test3() {
    assertTrue("Seharusnya True", m.cekUmur(21));
}

@Test
public void test4() {
    assertFalse("Seharusnya False", m.cekUmur(-1));
}

@Test
public void test5() {
    assertTrue("Seharusnya False", m.cekUmur(101));
}
```

- **Coba Running kembali kelas MahasiswaTest, amati apa yang terjadi !**
- **Coba ubah nilai parameter method test3 dengan nilai -21, amati apa yang terjadi !**

Tambahkan kode ini kedalam kelas Mahasiswa :

```
public String getIndex(double ipk) {
    String toReturn = null;
    if (ipk == 4.0) {
        toReturn = "A";
    } else if (ipk == 3.0) {
        toReturn = "B";
    } else if (ipk == 2.0) {
        toReturn = "C";
    } else if (ipk == 1.0) {
        toReturn = "D";
    } else if (ipk < 1.0) {
        toReturn = "E";
    } else {
        toReturn = "NONE";
    }
    return toReturn;
}
```

Tambahkan pula kode ini pada kelas MahasiswaTest :

```
@Test
public void test6() {
   assertSame("Harusnya ", "A", m.getIndex(4.0));
}

@Test
public void test7() {
   assertSame("Harusnya ", "B", m.getIndex(3.5));
}

@Test
public void test8() {
   assertSame("Harusnya ", "C", m.getIndex(2.3));
}

@Test
public void test9() {
   assertNotSame("Harusnya ", "A", m.getIndex(5.0));
}

@Test
public void test10() {
   assertNotSame("Harusnya ", "D", m.getIndex(-1.0));
}
```

- **Coba ganti nilai expectide method test6 menjadi “B”, amati apa yang terjadi !**
- **Coba ganti method Assert pada test10 menjadi assertSame, amati apa yang terjadi !**

Untuk lebih mengenal Annotation pada JUnit buatlah kelas seperti dibawah ini :

```
public class JunitAnnotation {

    // mengeksekusi before class
    @BeforeClass
    public static void beforeClass() {
        System.out.println("in before class");
    }

    // mengeksekusi after class
    @AfterClass
    public static void afterClass() {
        System.out.println("in after class");
    }

    // mengeksekusi before test
    @Before
    public void before() {
        System.out.println("in before");
    }

    // mengeksekusi after test
    @After
    public void after() {
        System.out.println("in after");
    }

    // mengeksekusi test case
    @Test
    public void test() {
        System.out.println("in test");
    }

    // mengeksekusi test case ignore and will not execute
    @Ignore
    public void ignoreTest() {
        System.out.println("in ignore test");
    }
}
```

Kemudian buat kelas TestRunner untuk menjalankan program diatas seperti ini :

```
public class TestRunner {

    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(JunitAnnotation.class);
        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }
        System.out.println(result.wasSuccessful());

    }

}
```

Running pada TestRunner menggunakan java application dan pada JunitAnnotation dengan menggunakan JUnit Test. Amati perbedaannya !

TUGAS

- Buatlah makalah mengenai Unit Testing, didalamnya terdapat spesifikasi sebagai berikut :
  - Pengertian dari Unit Testing
  - Tujuan dari Unit Testing
  - Framework Unit Testing (minimal 3 & Selain JUnit)
- Tambahkan pula kedalam makalah tersebut :
  - Penggunaan Annotation (Cari anotasi lain yang bisa digunakan pada pada JUnit) dan berikan penjelasannya dan contoh programnya
  - Cari minimal 10 method pengujian Assert, berikan penjelasan kegunaanya dan contoh programnya !
- Format :.PDF