**LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK**

**JOBSHEET 15: UNIT TEST**

****

oleh :

Halim Teguh Saputro

2E

2141762122

**PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

Jl. Soekarno Hatta No .9, Jatimulyo, Kec. Lowokwaru, Kota Malang,

JawaTimur 65141

|  |
| --- |
| KOMPETENSI |

Setelah melakukan percobaan pada modul ini, mahasiswa memahami konsep:

1. Memahami konsep dan fungsi unit testing
2. Menerapkan unit testing dengan Junit pada program sederhana
3. Menerapkan unit testing dengan Junit pada program terkoneksi database

|  |
| --- |
| PRAKTIKUM 1. DASAR UNIT TESTING |

1. Buatlah sebuah project baru di NetBeans dengan nama LearnUnitTesting. Pada package unittest, di tempat yang sama dengan class mainnya, buatlah 1 file class baru bernama MessageProcessor.java.
2. Bukalah file MessageProcessor.java, ketikkan baris-baris kode berikut yang terdiri dari beberapa properties. Setelah itu generate-lah setter untuk semua variabel-nya.

package unittest;

public class MessageProcessorTest {

    private String sender, recipient, message;

    public String getMessage() {

        return message;

    }

    public String getRecipient() {

        return recipient;

    }

    public String getSender() {

        return sender;

    }

    public void setMessage(String message) {

        this.message = message;

    }

    public void setRecipient(String recipient) {

        this.recipient = recipient;

    }

    public void setSender(String sender) {

        this.sender = sender;

    }

1. Berikutnya, tambahkan 3 method berikut pada class MessageProcessor, pastikan semua access modifier-nya public.

    public String messageFormat() {

        String message = String.format("Hai %s, you have message from %s\nThe message as follows : %s",

                this.recipient, this.sender, this.message);

        return message;

    }

    public void showMessage() {

        System.out.println(messageFormat());

    }

}

Ketiga method yang dibuat ini adalah method yang sangat sederhana dan semua kegunaannya sesuai dengan nama methodnya. Method messageFormat(), hanya menambahkan beberapa kata pembuka yang berisi variabel sender dan recipient di awal dan message kemudian mengembalikannya sebagai sebuah String.

1. Berikutnya bukalah file class LearnUnitTesting.java yang telah dibuat pada langkah ke-1. Pada class tersebut tambahkan kode untuk menjalankan class MessageProcessor seperti berikut.

package unittest;

public class LearnUnitTesting {

    public static void main(String[] args) {

        MessageProcessorTest mProcessor = new MessageProcessorTest();

        mProcessor.setSender("Halim");

        mProcessor.setRecipient("Rafael");

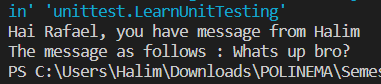
        mProcessor.setMessage("Whats up bro?");

        mProcessor.showMessage();

    }

}

1. Eksekusi program seperti biasa lewat NetBeans untuk melihat jalannya program MessageProcessor yang telah anda buat.



1. Sampai disini, aplikasi telah selesai dibuat. Berikutnya akan kita buat pengetesan unit sederhana untuk program kita tersebut. Untuk melakukannya, pertama kali buatlah sebuah package bernama unittesting (sama dengan nama package project), di folder Test Packages. Pada package tersebut, tambahkan 2 buah file class bernama MessageProcessorTest.java dan TestRunner.java. Perhatikan penamaan class testing disini! Untuk menamai class untuk pengetesan maka konvensinya adalah NamaClass + Test = NamaClassTest.
2. Pertama-tama bukalah terlebih dahulu file class MessageProcessorTest.java pada class tersebut tambahkan baris-baris kode berikut.

Pada class ini terdapat objek dari class MessageProcessor dengan nama mProcessor, dimana variabel tersebut nantinya menjadi subjek uji dari class MessageProcessorTest. Data-data dari objek uji tersebut diisi manual pada konstruktor class test. Pada konstruktor tersebut diisi data-data yang diketahui agar dapat dibandingkan dengan data yang dihasilkan pada saat tes dijalankan.

1. Berikutnya tambahkan kode berikut pada method uji testShowMessage(). Kode pada method tersebut alurnya hampir sama dengan method pada langkah sebelumnya yaitu membandingkan ‘hasil yang diharapkan’, yang diisi manual berdasarkan nilai properti pesan dari objek mProcessor yang diisikan pada konstruktor, dengan hasil aktual dari method messageFormat() pada objek tersebut.
2. Kemudian bukalah file class TestRunner.java yang telah dibuat sebelumya, lalu isikan didalamnya baris-baris kode berikut.

Class ini adalah class Main dari Test Packages, class ini yang nantinya dieksekusi ketika akan menjalankan tes secara keseluruhan. Pada class tersebut dilakukan eksekusi class MessageProcessorTest, dimana hasilnya kemudian akan diambil informasi mengenai unit-unit tes yang gagal dan kemudian ditampilkan pesan gagalnya karena apa.

1. Klik kanan pada file class tersebut lalu pilih Run. Maka pada output window akan dapat dilihat hasil seperti berikut.
2. Hasil unit testing juga dapat dilihat melalui Test Results Window jika kita menjalankan unit testing lewat menu Run  Test Project (Nama Project). Pada window ini akan ditampilkan bar warna hijau jika semua tes berhasil dijalankan, dan akan menampilkan warna merah jika sebaliknya. Disamping bar juga ditampilkan pesan-pesan yang kita kirim lewat System.out.println().

|  |
| --- |
| PRAKTIKUM 2. UNIT TESTING DENGAN TEST CASE |

Pada percobaan kedua kita akan mencoba membuat test case dari salah satu percobaan pada jobsheet minggu ke 14. Struktur file dari percobaan ke satu terdiri dari 4 file class yaitu DBHelper.java, FormKategori.java, Kategori.java dan TestBackend.java.

Hasil akhir dari percobaan satu jobsheet ke 14 adalah seperti pada gambar berikut. Aplikasi dapat melakukan insert, update, select dan delete ke database dbperpus tabel kategori.

1. Bukalah project yang telah Anda buat pastikan dapat berjalan. Setelah project tersebut dapat dijalankan, bukalah folder Test Packages pada project explorer. Kemudian buat package dengan nama database, pilih New  Other…
2. Selanjutnya pilih Test for Existing Class.
3. Di jendela dialog yang muncul berikutnya, klik Browse… lalu pilih class Kategori.java, karena class ini lah yang akan kita buatkan unit testnya. Di jendela dialog yang muncul berikutnya klik Finish.
4. Setelah selesai langkah sebelumnya, perhatikan di folder Test Packages, sudah dibuatkan secara otomatis sebuah class test bernama KategoriTest. Bukalah class tersebut dan perhatikan kode-kode yang terdapat didalamnya!

Pada bagian atas class terdapat beberapa baris kode yang terdiri dari konstruktor, static setUpClass(), static tearDownClass(), setup(), dan tearDown(). Masing- masing method tersebut diatasnya terdapat anotasi yang berbeda-beda. Maksud dari masing-masing anotasi tersebut adalah bahwa pada saat tes dijalankan maka method dengan anotasi @BeforeClass akan dijalankan sekali pada saat awal pertama kali class test akan dijalankan. @AfterClass dijalankan sekali pada saat seluruh class test dijalankan. @Before dijalankan setiap kali masing-masing method test akan dijalankan, dan sebaliknya @After dijalankan setiap kali masing-masing method test selesai dijalankan.

1. Selain itu jika Anda perhatikan di bagian bawah class yang sama, terdapat beberapa method yang sesuai dengan yang ada pada class Kategori. Bukalah class Kategori tersebut, perhatikan disana terdapat method testGetIdkategori(), testSetIdkategori(), testSearch(), testSave(), dan testDelete(). Expand-lah semua method-method yang otomatis dibuatkan tersebut, lalu hapus method get test, set test, dan sisakan method testSave dan testSearch . Seperti pada screenshoot di bawah ini!
2. Selanjutnya, kita buat class KategoriTest tersebut menjadi turunan dari class milik JUnit yaitu class TestCase.

Dengan meng-extends class TestCase tersebut kita dapat dengan mudah memperoleh informasi tentang test yang saat ini sedang dijalankan. Contoh dari informasi-informasi ini salah satunya yang terpenting adalah tentang nama method unit test yang sedang dijalankan, cukup dipanggil dengan syntax: this.getName().

1. Selanjutnya tambahkan beberapa properties pada class KategoriTest yang kita buat sebagaimana ditunjukkan pada kode yang di-highlight pada gambar di bawah.
2. Lalu pada method setUp() dan teardown(), tambahkan baris-baris kode berikut:

Pada method setUp() ini kita mempersiapkan segala sesuatu yang dibutuhkan untuk semua method test pada class KategoriTest. Diantaranya adalah mengisi variabel secara manual sehingga kita bisa memprediksi hasil eksekusi method insert dan search.

1. Selanjutnya adalah salah satu yang terpenting yaitu menulis kode untuk mengetes method testSave() dan testSearch()!

Karena save atau insert ke database tidak memiliki return type maka salah satu cara yang valid untuk melihat insert ke database sukses atau tidak adalah dengan membuat query untuk me-select bedasarkan nama dan keterangan yang diinsert. Maka tambahkan method getByNamaDanKeterangan(), di Kategori.java

1. Run class Kategori.java bagaimana hasilnya?
2. Edit file bernama TestRunner pada percobaan sebelumnya menjadi seperti berikut.
3. Kode-kode pada class ini hampir sama dengan kode class TestRunner yang ada pada percobaan sebelmnya. Bedanya hanya terletak pada class yang akan diuji di parameter method JUnitCore.runClasses() saja. Jika test Runner di run hasilnya adalah sebagai berikut:

|  |
| --- |
| TUGAS |

1. Buat test case untuk percobaan kedua pada jobsheet ke 14.