Praktikum	: Pengujian Perangkat Lunak
Kode Mata Kuliah	: IF6008043
Semester/ SKS	: 4/ 1 SKS
Program Studi	: Teknik Informatika
Dosen Pengampu	: Ir. Sri Mulyono, M.Eng

MODUL 3
UNIT TESTING

A. TUJUAN

a) Mengetahui metode unit testing

b) Mengisolasi setiap bagian dari program

B. KEMAMPUAN AKHIR YANG DIHARAPKAN

a) Mahasiswa mampu menjelaskan unit testing

c) Mahasiswa mampu melakukan isolasi setiap bagian dari program

C. HARDWARE & SOFTWARE

a) Komputer

b) OS Windows

c) Microsoft Office

d) Eclipse

D. DASAR TEORI

Dalam pemrograman komputer, unit testing adalah metode dimana unit individu dari kode sumber, menetapkan satu atau lebih modul program komputer bersama-sama dengan data kontrol terkait, prosedur penggunaan, dan prosedur operasi diuji untuk menentukan apakah mereka layak digunakan. Secara intuitif, kita dapat melihat unit sebagai bagian diuji terkecil dari sebuah aplikasi.

Dalam pemrograman prosedural unit bisa menjadi modul seluruh tetapi lebih umum fungsi individu atau prosedur. Dalam pemrograman berorientasi obyek unit sering merupakan seluruh antarmuka, seperti kelas, tetapi bisa menjadi metode individu. Unit test yang dibuat oleh programmer atau kadang-kadang oleh penguji

kotak putih selama proses pembangunan. Idealnya, setiap kasus uji independen dari yang lain: pengganti seperti bertopik metode, objek tiruan, palsu dan memanfaatkan tes dapat digunakan untuk membantu pengujian modul secara terpisah.

Unit test biasanya ditulis dan dijalankan oleh pengembang perangkat lunak untuk memastikan bahwa kode memenuhi desain dan berperilaku sebagaimana dimaksud. Pelaksanaannya dapat bervariasi dari yang sangat manual (pensil dan kertas). Untuk yang diformalkan sebagai bagian dari otomatisasi membangun.

Tujuan dari unit testing adalah untuk mengisolasi setiap bagian dari program dan menunjukkan bahwa bagian-bagian individu sudah benar. Sebuah tes unit menyediakan kontrak, ketat ditulis bahwa potongan kode harus memenuhi. Akibatnya afford dapat dibagi menjadi beberapa manfaat atau keuntungan

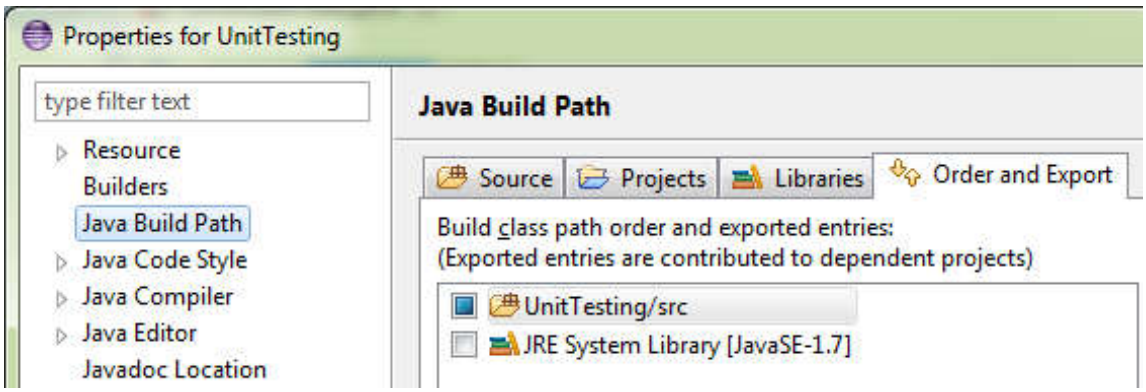
E. LANGKAH PRAKTIKUM

1. Unit Assertions

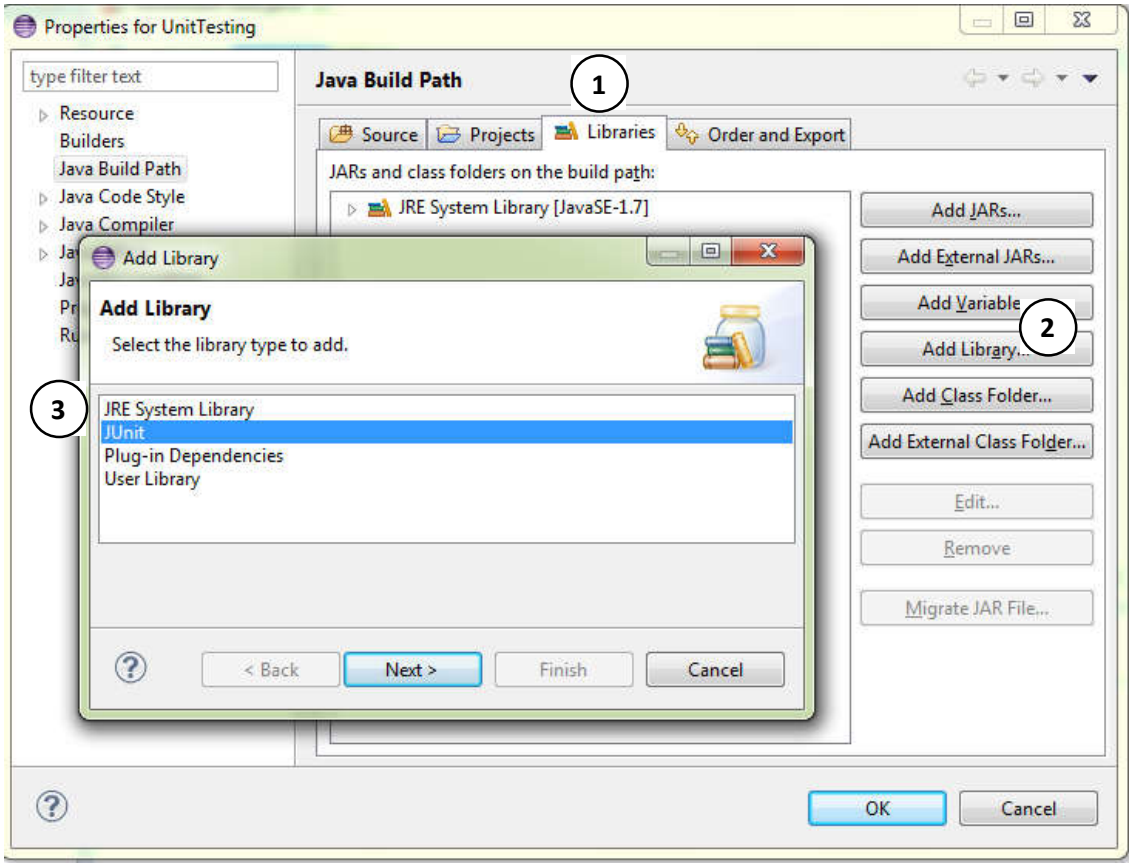
a. Buat project baru pada eclipse dengan Nama **UnitTesting**

b. Tambahkan JUnit Library pada project **UnitTesting**, klik kanan project pilih → **Properties**

c. Pilih **Java Build Path** (lihat gambar)

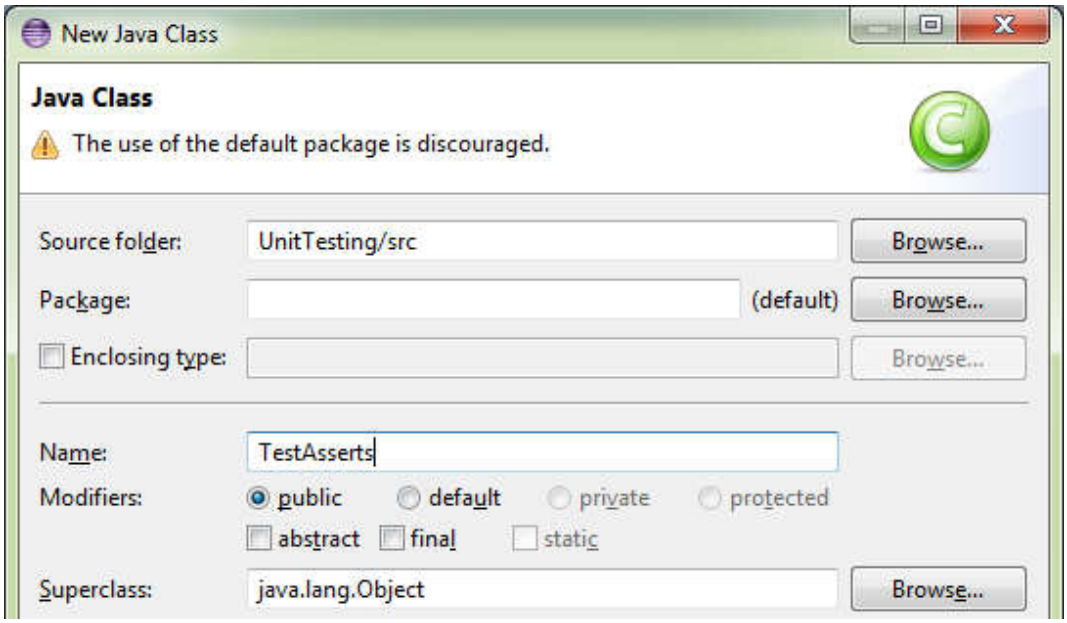


d. Pilih tab **Libraries** dan klik tombol **Add Library**. Kemudian pilih **JUnit** (lihat gambar)



e. Klik **Next**, pilih **JUnit 4** dan klik **Finish.** Klik **OK** pada Properties window

f. Buat class baru pada project dengan nama **TestAsserts** (lihat gambar dibawah)



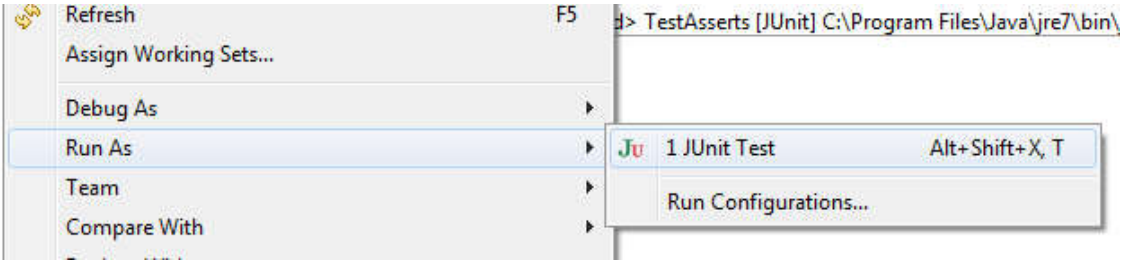
Klik **Finish**

g. Tulis kode berikut pada class **TestAsserts**

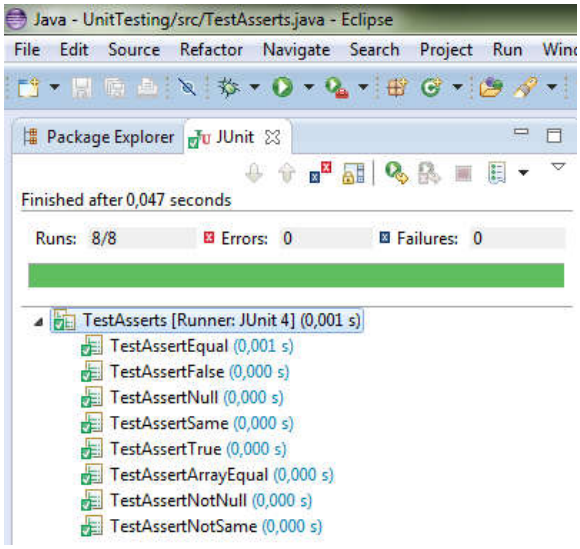
```
import org.junit.Test;
import static org.junit.Assert.*;

public class TestAsserts {

// test data
String str1 = new String("abc");
String str2 = new String("abc");
String str3 = null;
String str4 = "abc";
String str5 = "abc";
int val1 = 5;
int val2 = 6;
String[] expectedArray = { "satu", "dua", "tiga" };
String[] resultArray = { "satu", "dua", "tiga" };

@Test
public void TestAssertEqual(){
        // cek apakah dua Object sama
        assertEquals(str1, str2);
}

@Test
public void TestAssertTrue(){
        // cek apakah kondisi benar (true)
        assertTrue(val1 < val2);
}

@Test
public void TestAssertFalse(){
        // cek apakah kondisi salah (false)
        assertFalse(val1 > val2);
}


@Test
public void TestAssertNull(){
        // cek apakah Object null
        assertNull(str3);
}

@Test
public void TestAssertNotNull(){
```

```
        // cek apakah sebuah Object tidak null
        assertNotNull(str1);
    }

    @Test
    public void TestAssertSame(){
        // cek apakah dua referensi menunjuk pada Object yang sama
        assertSame(str4, str5);
    }

    @Test
    public void TestAssertNotSame(){
        // cek apakah dua referensi tidak menunjuk pada Object yang sama
        assertNotSame(str1, str3);
    }

    @Test
    public void TestAssertArrayEqual(){
        // cek apakah dua Array sama
        assertArrayEquals(expectedArray, resultArray);
    }
    }
```

h. Jalankan sebagai JUnit Test, klik kanan class **TestAsserts → Run As → 1 JUnit Test** (lihat gambar)



i. Amati Hasilnya, catat, ambil screeshot dari view JUnit



j. Ubah kode pada test data di class TestAsserts menjadi seperti berikut:

```
// test data
String str1 = new String("defgh");
String str2 = new String("abc");
String str3 = null;
String str4 = "abcdef";
String str5 = "abc";
int val1 = 56;
int val2 = 6;
String[] expectedArray = { "empat", "lima", "enam" };
String[] resultArray = { "satu", "dua", "tiga" };
```

k. Jalankan kembali sebagai JUnit Test dan amati hasilnya, ambil screenshot-nya

l. Berikan penjelasan pada hasil test-nya

2. *Unit Testing 1*

a. Buat class baru pada project yang sama, beri nama class **Kalkulator**

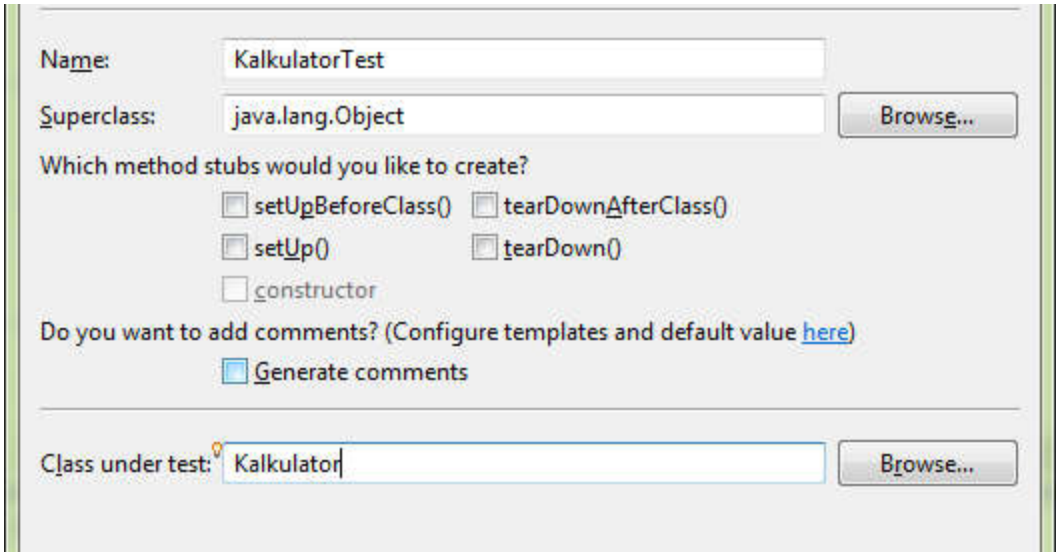
b. Tulis kode berikut pada class **Kalkulator**

```
public class Kalkulator {

        public int kali(int a, int b){
                int hasil = 0;
                return hasil;
        }

        public int bagi(int a, int b){
```

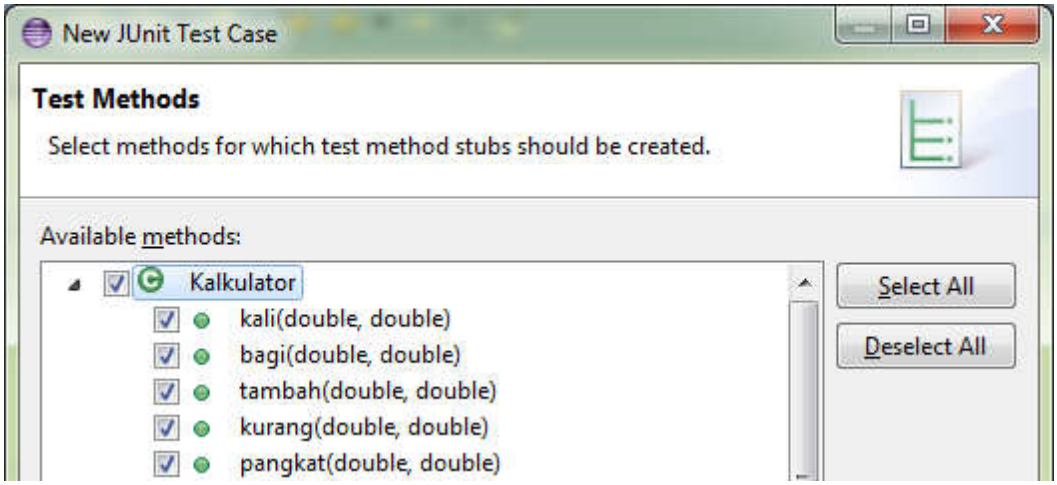
```
        int hasil = 0;
        return hasil;
    }

    public int tambah(int a, int b){
        int hasil = 0;
        return hasil;
    }

    public int kurang(int a, int b){
        int hasil = 0;
        return hasil;
    }

    public int pangkat(int a, int b){
        int hasil = 0;
        return hasil;
    }

}
```

c. Buat Test Class untuk class Kalkulator, klik kanan pada project → **New** → **JUnit Test Case**

d. Beri *nama*: KalkulatorTest, *Class under test* : **Kalkulator** (lihat gambar)



e. Klik **Next**, pilih semua method pada class Kalkulator (lihat gambar)



f. Klik Finish.

g. Edit kode untuk **KalkulatorTest** seperti berikut:

```
import static org.junit.Assert.*;
import org.junit.Test;


public class KalkulatorTest {

	Kalkulator kalkulator = new Kalkulator();

	int a = 10;
	int b = 2;

	int i = 4;
	int j = 0;

	@Test
	public void testKali() {
		assertEquals(20, kalkulator.kali(a, b));
	}


	@Test
	public void testBagi() {
		assertEquals(5, kalkulator.bagi(a, b));
	}

	@Test
	public void testTambah() {
		assertEquals(12, kalkulator.tambah(a, b));
	}
```

```
    @Test
    public void testKurang() {
        assertEquals(8, kalkulator.kurang(a, b));
    }

    @Test
    public void testPangkat() {
        assertEquals(100, kalkulator.pangkat(a, b));
    }

}
```

h. Jalankan class **KalkulatorTest** sebagai JUnit Test, amati hasilnya pada output JUnit, ambil screenshot

i. Berikan analisis terhadap unit testing diatas

3. Unit Testing 2

a. Edit class Kalkulator menjadi seperti berikut:

```
import java.util.Scanner;

public class Kalkulator {

    public int kali(int a, int b){
        int hasil = 0;
        hasil = a * b;
        return hasil;
    }



    public int bagi(int a, int b){
        int hasil = 0;
        hasil = a / b;
        return hasil;
    }

    public int tambah(int a, int b){
        int hasil = 0;
        hasil = a + b;
        return hasil;
    }
```

```
    public int kurang(int a, int b){
        int hasil = 0;
        hasil = a - b;
        return hasil;
    }

    public int pangkat(int a, int b){
        int hasil = 0;
        hasil = (int) Math.pow(a, b);
        return hasil;
    }

    public static void main(String[] args){
        Kalkulator kalkulator = new Kalkulator();

        Scanner input = new Scanner(System.in);

        int a;
        int b;
        int hasil;

        System.out.print("Tuliskan bilangan pertama (a): ");
        a = input.nextInt();

        System.out.print("Tuliskan bilangan kedua (b): ");
        b = input.nextInt();


        System.out.println("Hasil dari operasi "+a+ " dan " +b);

        System.out.println("Kali :"+ kalkulator.kali(a,b));
        System.out.println("Bagi :"+ kalkulator.bagi(a,b));
        System.out.println("Tambah :"+ kalkulator.tambah(a,b));
        System.out.println("Kurang :"+ kalkulator.kurang(a,b));
        System.out.println("Pangkat :"+ kalkulator.pangkat(a,b));

    }

}
```

b. Lakukan test dengan class KalkulatorTest yang sama, amati hasilnya
c. Apabila semua “passed” jalankan class Kalkulator sebagai Java Application, dan berikan input berikut:

    1. a = 5 , b =5

2. a = -1, b =1
3. a = -2, b = -2
4. a = 0, b = 4
5. a = 5, b = 0

catat hasil dari setiap input diatas, apakah sudah benar dan tidak ada error. Jika ada error catat dan perbaiki kode supaya tidak error (tanyakan pada asisten praktikum).

d. Beri analisa kenapa bisa terjadi error.
e. Lakukan test kembali setelah memperbaiki kode. Amati catat dan ambil screenshotnya

F. TUGAS

a. Aplikasi penghitung Penghitung Saldo dengan requirement berikut:
   1. User memasukan 3 kali pemasukan
   2. Saldo pertama adalah jumlah dari 3 kali pemasukan tersebut
   3. User diperbolehkan melakukan 3 kali penarikan
   4. Jika saldo lebih dari (>) 50.000 maka dapat melakukan penarikan
   5. Saldo yang ada di bank minimal harus tertinggal sebesar 50.000
b. Kode untuk aplikasi bank adalah sbb:

```
import java.util.Scanner;

public class Bank {
static int saldo = 0;

public static void main(String[] args) {

        Bank bank = new Bank();
        Scanner sc = new Scanner(System.in);

        // tiga kali pemasukan
        for (int i = 1; i <= 3; i++) {
                System.out.print("Masukkan setoran " + i + " : ");
                int setoran = sc.nextInt();
                if (!bank.pemasukan(setoran)) {
                        break;
                }
        }
```

```
        // tiga kali penarikan
        for (int i = 1; i <= 3; i++) {
                System.out.println("Saldo : " + bank.tampilkan_saldo());
                System.out.print("Lakukan Penarikan? (Y/N): ");
                String yes = sc.next();
                if (yes.equals("Y")) {
                        System.out.print("Jumlah penarikan: ");
                        int tarik = sc.nextInt();
                        if (!bank.penarikan(tarik)) {
System.out.println("Tidak bisa melakukan penarikan, saldo : " + saldo);
                                break;
                        }
                } else {
                        bank.tampilkan_saldo();
                        break;

                }
                bank.tampilkan_saldo();
        }

}

        public boolean pemasukan(int p1) {
                boolean b = false;
                if (p1 <= 0) {
                        b = false;
                } else {
                        saldo = p1;
                        b = true;
                }
                return b;
        }

        public boolean penarikan(int t1) {
                boolean b = false;
                if (saldo == 50000) {
                        saldo = saldo - t1;
                        b = true;
                }
                return b;
        }

        public int tampilkan_saldo() {
                return saldo;
        }

}
```

c. Lakukan unit testing pada method pemasukan() dan penarikan(), dan perbaiki kodenya agar hasilnya sesuai requirement diatas.

G. ANALISIS

Tuliskan analisis anda pada laporan

H. KESIMPULAN

Tuliskan kesimpulan anda pada laporan