

MODUL & GUIDED

Praktikum Pemrograman Berorientasi Objek 2023



MODUL 14

OBJECT ORIENTED TESTING

TUJUAN

- Praktikan mampu memahami konsep object oriented testing pada java.
- Praktikan mampu mengimplementasikan pemahaman tersebut ke dalam penyelesaian kasus tertentu menggunakan bahasa Java.

PENJELASAN

Level dalam testing dibagi menjadi:

- Method testing (Unit testing).
- Class testing (Unit testing).
- Inter-class testing (Integration testing).
- System testing.

Unit Testing adalah pengujian sebuah kelas tunggal (*single class*), baik pengujian method atau pengujian kelas. Dalam melakukan pengujian, haruslah disusun test casenya terlebih dahulu. Test case berguna sebagai rencana pengujian berupa daftar skenario pengujian untuk menentukan apakah suatu unit code sudah bekerja dengan baik atau tidak. Test case biasanya berupa tabel yang berisi sekumpulan input (masukan) dan output (keluaran) yang diharapkan. Salah satu cara untuk menyusun test case adalah dari pre condition dan post condition dari method yang akan diuji. Pre condition adalah keadaan yang harus terpenuhi sebelum method dieksekusi, sedangkan post condition adalah keadaan setelah method dieksekusi.

Beberapa terminologi:

- Test fixture: set up data yang dibutuhkan untuk menjalankan pengujian.
- Test case: pengujian terhadap respons sebuah fungsi terhadap sekumpulan masukan.
- Unit test: pengujian sebuah kelas tunggal (*single class*), baik method atau class testing.
- Test suite: kumpulan unit test
- Test runner : perangkat lunak yang menjalankan pengujian serta memberikan hasil pengujian.

Unit testing yang akan dilakukan menggunakan library JUnit. Pada library ini terdapat fungsi yang memudahkan kita untuk mengecek suatu kondisi apakah benar atau tidak menggunakan **assert**. Berikut beberapa contoh assert yang ada pada JUnit:

```

assertTrue(boolean condition)
assertFalse(boolean condition)

assertEquals(Object expected, Object actual)
    • Uses equals() comparison
    • Overloaded for all primitive types

assertSame(Object expected, Object actual)
assertNotSame(Object expected, Object actual)
    • Uses == comparison

assertNull(Object o)
assertNotNull(Object o)

fail(String message)

```

Pada test class yang terbuat otomatis, jangan lupa menambah **extends junit.framework.TestCase**. Dengan menambah extends tersebut, test class akan mewarisi method:

- `protected void setUp()`
Dipanggil sebelum pemanggilan setiap fungsi yang diuji (test methods). Sifat opsional.
- `protected void tearDown()`
Dipanggil setelah pemanggilan setiap fungsi yang diuji (test methods). Sifat opsional.

Setiap test method yang ada di test class, jangan lupa menambahkan “**test**” di depan nama fungsinya. Jika “**test**” tidak ditambahkan, maka fungsi tersebut tidak akan dipanggil secara otomatis untuk pengujian. Contoh `public void testHasil()`.

Test suite adalah kumpulan unit test sehingga dapat menjalankan beberapa unit test sebagai satu grup. Untuk menggunakan test suite harus mengimport library `RunWith` dan `Suite`.

Contoh penggunaannya:

```

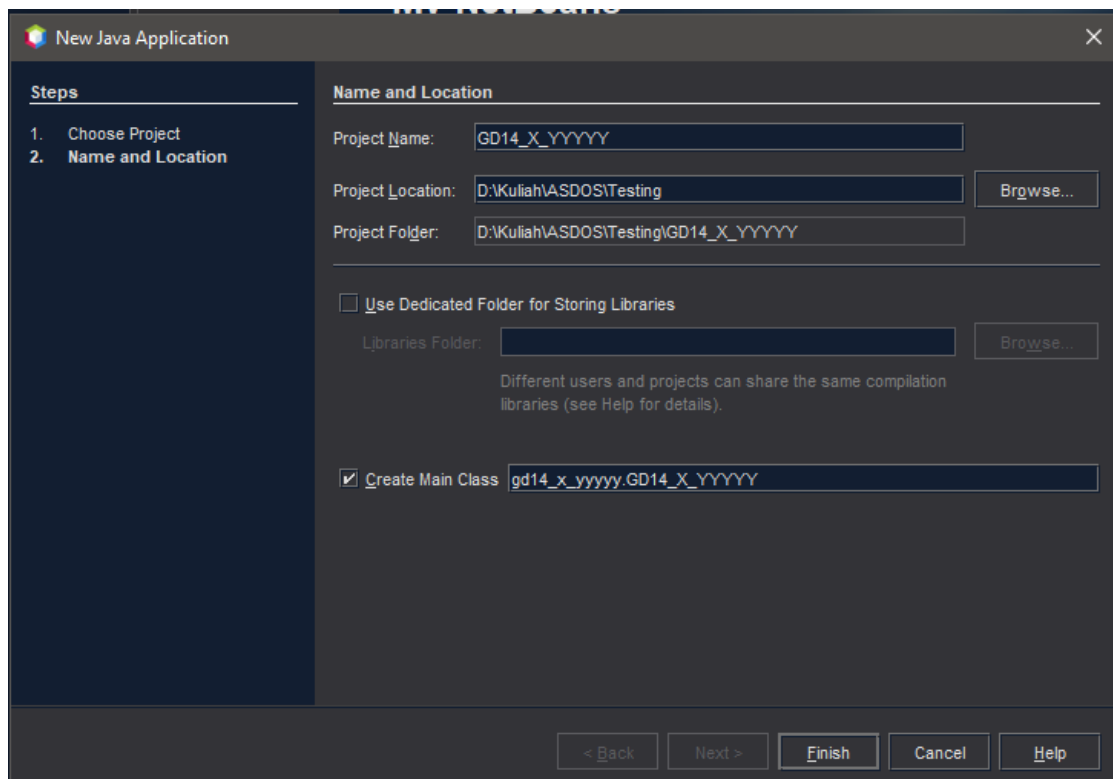
import org.junit.runner.RunWith; import
org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({Kumpulan Unit Test})

```

GUIDED

1. Buatlah project di Apache Netbeans dengan format nama GD14_X_YYYYY (X = Kelas dan YYYYYY = 5 digit terakhir NPM praktikan).



2. Buatlah kelas dengan nama Kalkulator lalu tambahkan kode di bawah ini.

```
public class Kalkulator {
    public static int tambah(int angka1, int angka2){
        if(angka1 > 100)
            throw new IllegalArgumentException("Input invalid");
        return angka1+angka2;
    }

    public static int kurang(int angka1, int angka2){
        if(angka2 > angka1)
            throw new IllegalArgumentException("Input invalid");
        return angka1 - angka2;
    }

    public static int kali(int angka1, int angka2){
        return angka1 * angka2;
    }

    public static int bagi(int angka1, int angka2){
        if(angka2 == 0)
            throw new IllegalArgumentException("Cannot divide by 0!");
        return angka1 / angka2;
    }
}
```

3. Berikut adalah pre dan post condition pada masing-masing method, beserta dengan test case nya.

Method tambah:

- **Pre Condition:** angka1 lebih dari 100, **Post Condition:** Muncul exception “Input Invalid”.
- **Pre Condition:** angka1 harus kurang dari sama dengan 100, **Post Condition:** mereturnkan hasil angka1 ditambah angka2.

Condition	Test angka1	Test angka2	Expected Output
angka1>100	101	0	Input Invalid
angka1<=100	50	10	60

Method kurang:

- **Pre Condition:** angka1 lebih kecil dari angka2, **Post Condition:** Muncul exception “Input Invalid”.
- **Pre Condition:** angka2 harus lebih kecil sama dengan angka1, **Post Condition:** mereturnkan hasil angka1 dikurangi angka2.

Condition	Test angka1	Test angka2	Expected Output
angka2>angka1	7	8	Input Invalid
angka2<=angka1	10	5	5

Method kali:

- **Pre Condition:** -, **Post Condition:** mereturnkan hasil angka1 dikali angka2.

Condition	Test angka1	Test angka2	Expected Output
-	0	0	0
-	5	7	35

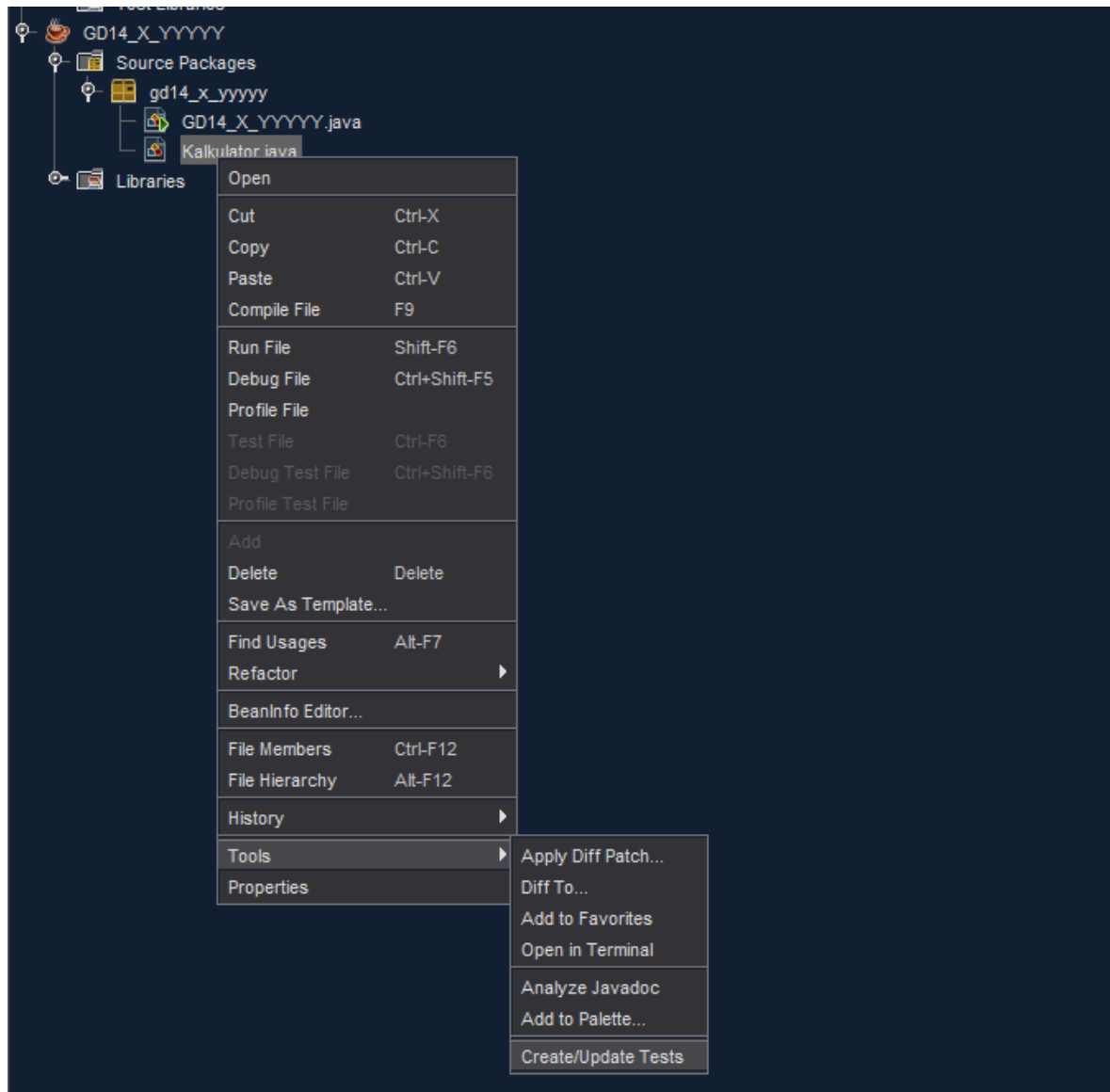
Method bagi:

- **Pre Condition:** angka2 sama dengan 0, **Post Condition:** Muncul exception “Cannot divide by 0”.
- **Pre Condition:** angka2 bukan 0, **Post Condition:** mereturnkan hasil angka1 dibagi angka2.

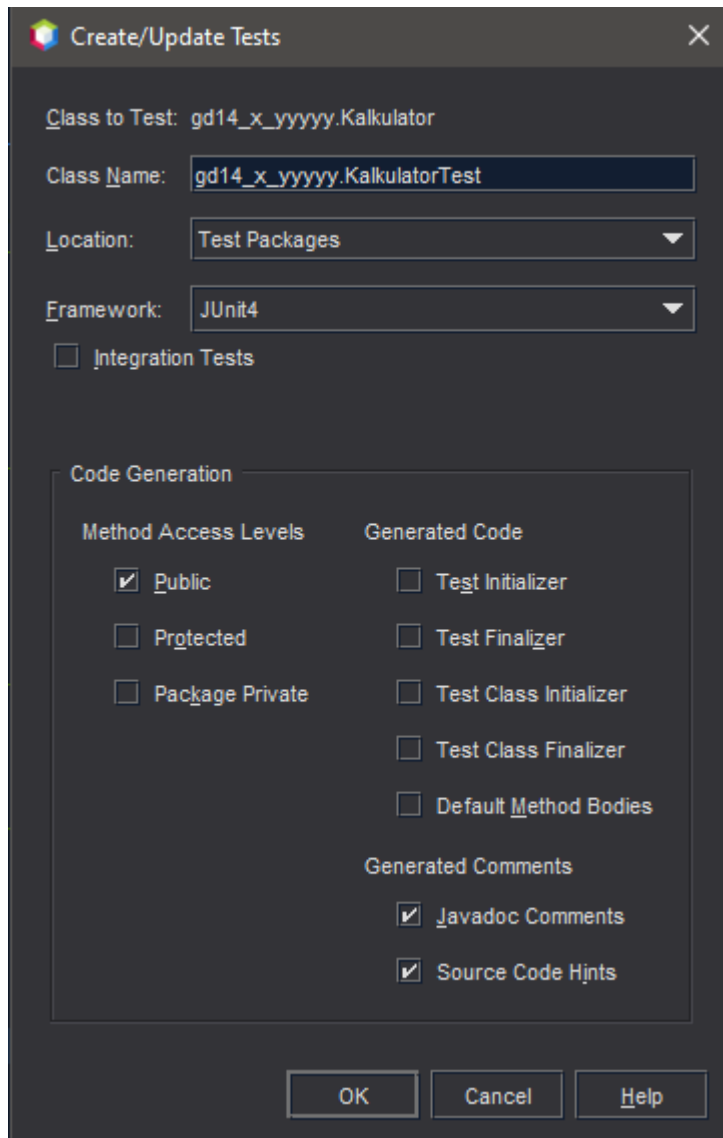
Condition	Test angka1	Test angka 2	Expected Output
angka2==0	10	0	Cannot divide by 0!

angka2!=0	10	2	5
-----------	----	---	---

- Selanjutnya kita akan membuat package Test Package dan Test Libraries yang otomatis generate oleh netbeans. Pada kelas Kalkulator, klik kanan dan ke bagian Tools lalu tekan Create/Update Tests.



5. Isi dan centang sesuai berikut. Lalu klik ok. (Pada Framework jangan lupa ubah ke JUnit4 agar langsung terisi Hamcrest 1.3, serta JUnit 4.13.2 pada package Test Libraries).



Create/Update Tests

Class to Test: gd14_x_yyyy.Kalkulator

Class Name: gd14_x_yyyy.KalkulatorTest

Location: Test Packages

Framework: JUnit4

☐ Integration Tests

Code Generation

Method Access Levels	Generated Code
<input checked="" type="checkbox"/> Public	<input type="checkbox"/> Test Initializer
<input type="checkbox"/> Protected	<input type="checkbox"/> Test Finalizer
<input type="checkbox"/> Package Private	<input type="checkbox"/> Test Class Initializer
	<input type="checkbox"/> Test Class Finalizer
	<input type="checkbox"/> Default Method Bodies

Generated Comments

☒ Javadoc Comments

☒ Source Code Hints

OK Cancel Help

6. Kelas KalkulatorTest berada di package TestPackage. Jangan lupa menambah extends junit.framework.TestCase pada KalkulatorTest.

```
public class KalkulatorTest extends junit.framework.TestCase {
```

7. Pada kelas KalkulatorTest, tambahkan kode di bawah ini.

```
public class KalkulatorTest extends junit.framework.TestCase{

    public KalkulatorTest() {
    }

    @Test
    public void testTambah() {
        System.out.println("tambah");
        int angka1 = 101;
        int angka2 = 0;
        int expectedResult = 101;
        int result = Kalkulator.tambah(angka1, angka2);
        assertEquals(expResult,result);
    }

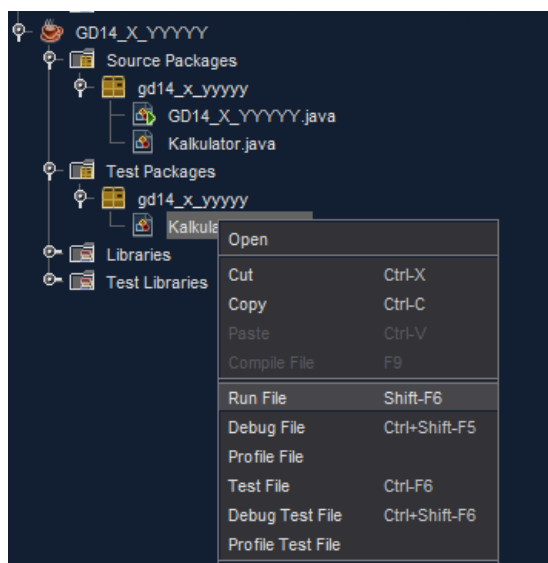
    @Test
    public void testKurang() {
        System.out.println("kurang");
        int angka1= 7;
        int angka2=8;
        int expectedResult=-1;
        int result=Kalkulator.kurang(angka1, angka2);
        assertEquals(expResult,result);
    }

    @Test
    public void testKali(){
        System.out.println("kali");
        int angka1=0;
        int angka2=0;
        int expectedResult=0;
        int result=Kalkulator.kali(angka1, angka2);
        assertEquals(expResult,result);
    }

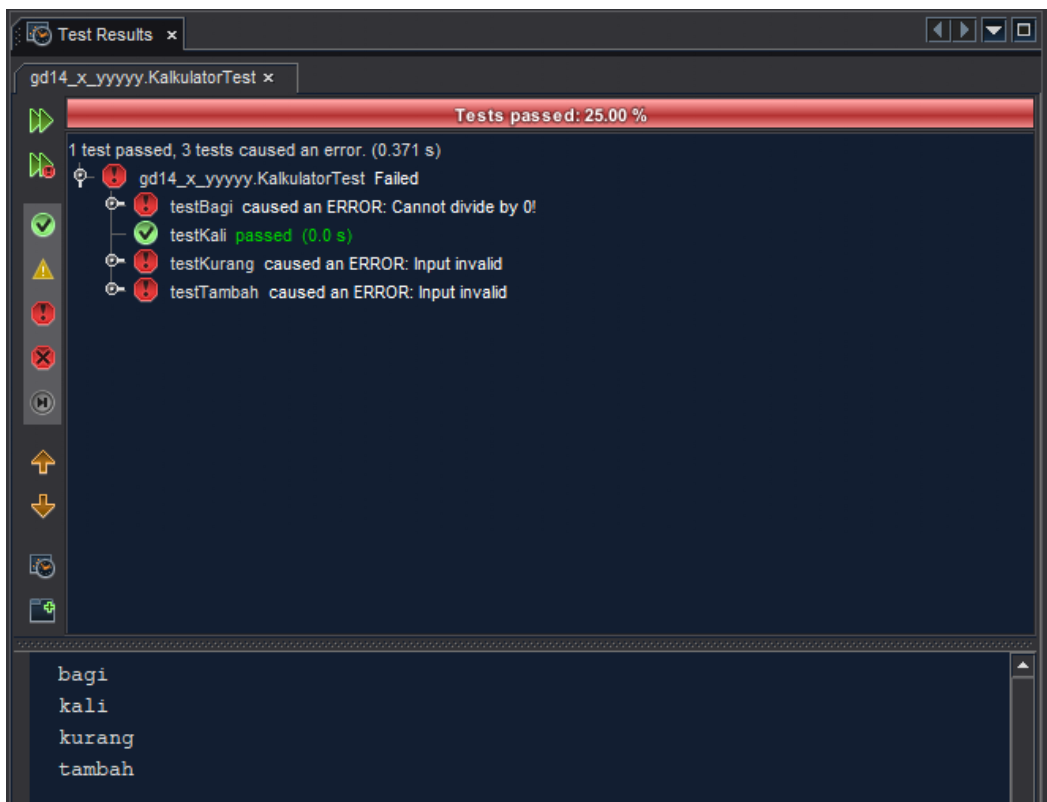
    @Test
    public void testBagi(){
        System.out.println("bagi");
        int angka1=10;
        int angka2=0;
        int expectedResult=0;
        int result = Kalkulator.bagi(angka1, angka2);
        assertEquals(expResult,result);
    }

}
```

8. Run kan KalkulatorTest saja dengan klik kanan pada kelas tersebut dan klik run file.



9. Hasilnya akan sebagai berikut. (Ada yang error bukan berarti salah tapi benar karena sesuai output yang ada di test case).



Penjelasan:

- Sesuai pada tabel test case method tambah, pada testTambah hasilnya error dan muncul exception. angka1 diisi dengan 101, kondisi $angka1 > 100$ ($101 > 100$), sehingga result akan menghasilkan exception, sedangkan expectedResult diisi dengan 101. Maka ketika dibandingkan menggunakan assertEquals, result dan expectedResult tidaklah equal.
- Sesuai pada tabel test case method kurang, pada testKurang hasilnya error dan muncul exception. angka1 diisi dengan 7, angka2 diisi dengan 8, kondisi $angka2 > angka1$ ($8 > 7$), sehingga result akan menghasilkan exception, sedangkan expectedResult diisi dengan -1. Maka ketika dibandingkan menggunakan assertEquals, result dan expectedResult tidaklah equal.
- Sesuai pada tabel test case method kali, pada testKali hasilnya passed. angka1 diisi dengan 0, angka2 diisi dengan 0, sehingga result akan menghasilkan 0 (0×0), sedangkan expectedResult diisi dengan 0. Maka ketika dibandingkan menggunakan assertEquals, result dan expectedResult equal.
- Sesuai pada tabel test case method bagi, pada testBagi hasilnya error dan muncul exception. angka2 diisi 0, sehingga result akan menghasilkan exception, sedangkan

expResults diisi 0. Maka ketika dibandingkan menggunakan assertEquals, result dan expResults tidaklah equal.

10. Ubah hasil kode yang ada di kelas KalkulatorTest sebagai berikut.

```
public class KalkulatorTest extends junit.framework.TestCase{

    public KalkulatorTest() {
    }

    @Test
    public void testTambah() {
        System.out.println("tambah");
        int angka1 = 50;
        int angka2 = 10;
        int expResult = 60;
        int result = Kalkulator.tambah(angka1, angka2);
        assertTrue(expResult == result);
    }

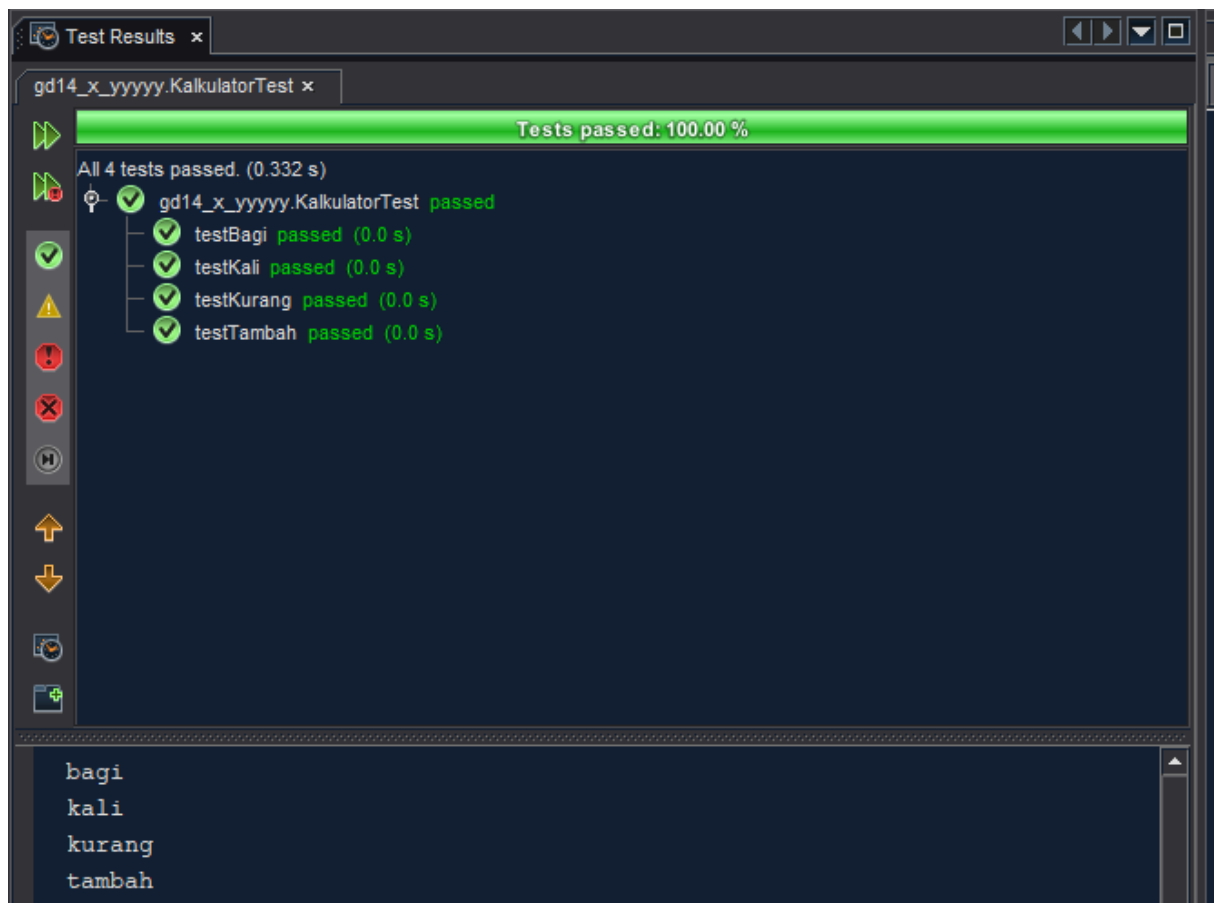
    @Test
    public void testKurang() {
        System.out.println("kurang");
        int angka1= 10;
        int angka2=5;
        int expResult=5;
        int result=Kalkulator.kurang(angka1, angka2);
        assertEquals(expResult, result);
    }

    @Test
    public void testKali() {
        System.out.println("kali");
        int angka1=5;
        int angka2=7;
        int expResult=30;
        int result=Kalkulator.kali(angka1, angka2);
        assertFalse(expResult == result);
    }

    @Test
    public void testBagi() {
        System.out.println("bagi");
        int angka1=10;
        int angka2=2;
        int expResult=5;
        int result = Kalkulator.bagi(angka1, angka2);
        assertEquals(expResult, result);
    }

}
```

Hasilnya akan sebagai berikut.



Penjelasan:

- Sesuai pada tabel test case method tambah, pada testTambah hasilnya passed. angka1 diisi dengan 50, angka2 diisi dengan 10, kondisi $\text{angka1} \leq 100$ ($50 \leq 100$), sehingga result akan menghasilkan 60 ($50+10$), sedangkan expResults diisi dengan 60. Maka ketika dibandingkan menggunakan `assertTrue`, result dan expResults sama sehingga True.
- Sesuai pada tabel test case method kurang, pada testKurang hasilnya passed. angka1 diisi dengan 10, angka2 diisi dengan 5, kondisi $\text{angka2} \leq \text{angka1}$ ($5 \leq 10$), sehingga result akan menghasilkan 5 ($10-5$), sedangkan expResults diisi dengan 5. Maka ketika dibandingkan menggunakan `assertEquals`, result dan expResults equal.
- Sesuai pada tabel test case method kali, pada testKali hasilnya passed. angka1 diisi dengan 5, angka2 diisi dengan 7, sehingga result akan menghasilkan 35 ($5*7$), sedangkan expResults diisi dengan 30. Maka ketika dibandingkan menggunakan `assertFalse`, result dan expResults tidak sama sehingga False.

- Sesuai pada tabel test case method bagi, pada testBagi hasilnya passed. angka1 diisi dengan 10, angka2 diisi dengan 5, kondisi $\text{angka2} \neq 0$ ($5 \neq 0$), sehingga result akan menghasilkan 2 ($10/5$), sedangkan expResults diisi 2. Maka ketika dibandingkan menggunakan assertEquals, result dan expResults equal.
11. Dikumpulkan dalam bentuk zip dengan nama GD14_X_YYYYYY.zip (X = Kelas, YYYYYY = 5 digit NPM terakhir).
 12. Selamat, Guided anda telah berhasil. Jangan lupa untuk dikumpulkan yaa.
 13. Pelajari Kembali materi tentang manipulasi string karena itu akan membantu dalam pengerjaan Unguided nanti.