АКАДЕМИЈА ТЕХНИЧКО-ВАСПИТАЧКИХ СТРУКОВНИХ СТУДИЈА

ОДСЕК НИШ

PROGRAMSKI ALATI ZA RAZVOJ SOFTVERA

Vežba 2: Pisanje testova u Javi

24.10.2024.

JUnit je jedan od najpoznatijih okvira za unit testiranje u Javi. Omogućava programerima da pišu i izvršavaju testove za male delove koda (obično metode) kako bi se osiguralo da oni rade ispravno. Testiranje pomoću JUnit-a je veoma korisno za:

- **Otkrivanje grešaka rano**: Pronalazi greške pre nego što kod dospe u produkciju.
- **Osiguranje stabilnosti**: Potvrđuje da nova funkcionalnost ili promene nisu oštetile postojeći kod.
- **Održavanje koda**: Olakšava ažuriranje i proširenje koda.

Osnovni Pojmovi u Unit Testiranju

1. **Unit test**: Testira najmanju jedinicu koda (obično jednu metodu) kako bi se osigurala njena tačnost.
2. **Test klasa**: Klasa koja sadrži skup povezanih test metoda.
3. **Test metoda**: Pojedinačna metoda u test klasi koja testira određenu funkcionalnost.
4. **Assertions (potvrde)**: Funkcije koje proveravaju očekivane rezultate; ako rezultat nije očekivan, test pada.

Instalacija JUnit Biblioteke u Eclipse-u

1. **Kreiranje Java projekta**: U Eclipse-u, otvorite File > New > Java Project i dajte ime vašem projektu.
2. **Dodavanje JUnit biblioteke**:
    - Desni klik na vaš projekat > Build Path > Add Libraries...
    - Izaberite JUnit i kliknite na **Next**. Najčešće se koristi JUnit 4, ali JUnit 5 je takođe popularan.
    - Kliknite na **Finish** da dodate JUnit biblioteke vašem projektu.
3. **Kreiranje Test Klase**:
    - Desni klik na paket gde želite da kreirate test klasu > New > JUnit Test Case.
    - Dajte ime test klasi, izaberite metode koje želite testirati, i kliknite na **Finish**.

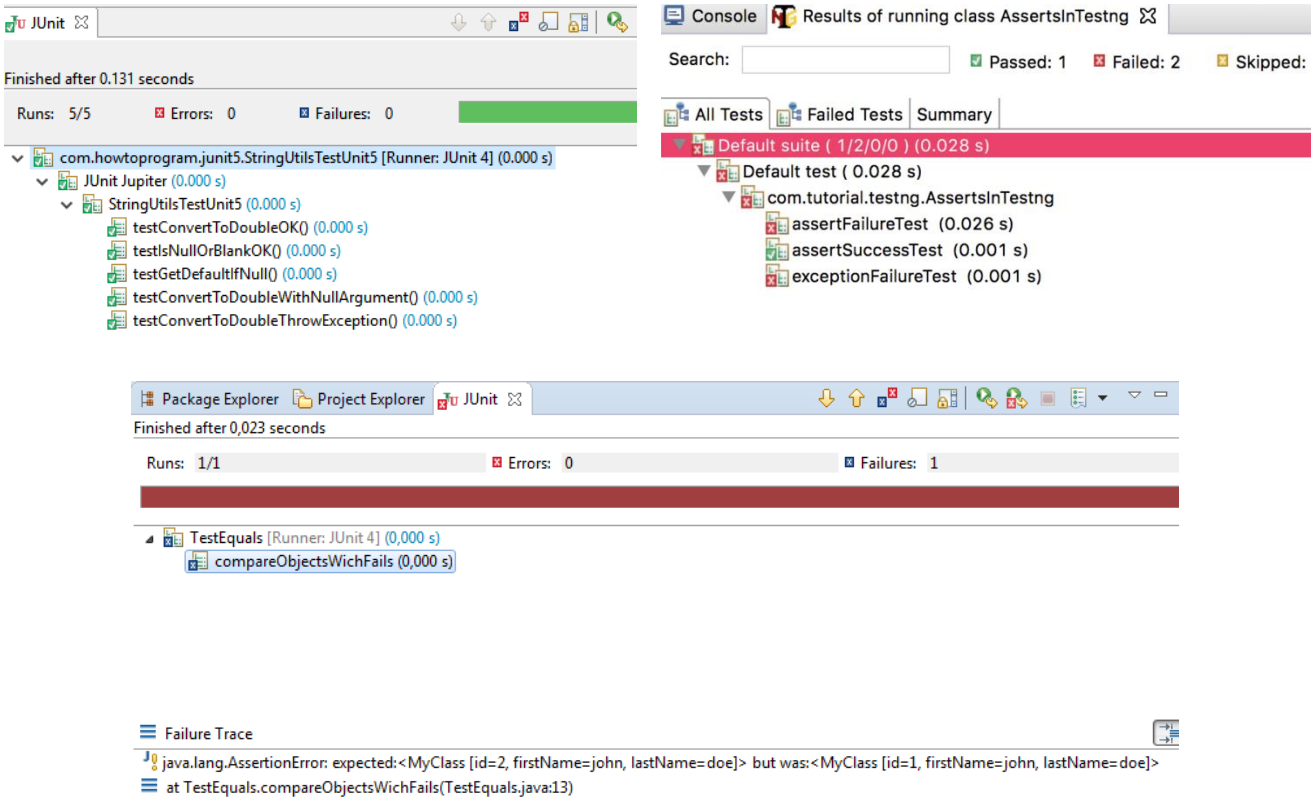
assert Metode u JUnit-u

JUnit koristi **assert** metode za poređenje očekivanih vrednosti sa stvarnim rezultatima. Svaka od njih služi za specifične tipove provera:

1. **assertEquals(expected, actual)**
   Proverava da li su dve vrednosti jednake. Ako nisu, test pada. Koristi se za poređenje numeričkih vrednosti, stringova, objekata itd.
   - **Primer**: assertEquals(5, calculator.add(2, 3));
2. **assertTrue(condition)**
   Proverava da li je uslov tačan (true). Ako je uslov false, test pada.
   - **Primer**: assertTrue(list.isEmpty());
3. **assertFalse(condition)**
   Proverava da li je uslov netačan (false). Ako je uslov true, test pada.
   - **Primer**: assertFalse(user.isLoggedIn());
4. **assertNull(object)**
   Proverava da li je objekat null. Ako objekat nije null, test pada.
   - **Primer**: assertNull(user.getEmail());
5. **assertNotNull(object)**
   Proverava da li objekat nije null. Ako je objekat null, test pada.
   - **Primer**: assertNotNull(databaseConnection);
6. **assertSame(expected, actual)**
   Proverava da li dve promenljive upućuju na isti objekat (koristi ==). Ako nije isti objekat, test pada.
   - **Primer**: assertSame(user1, user2);
7. **assertNotSame(expected, actual)**
   Proverava da li dve promenljive upućuju na različite objekte. Ako su isti objekti, test pada.
   - **Primer**: assertNotSame(product1, product2);
8. **assertArrayEquals(expectedArray, actualArray)**
   Proverava da li su dva niza jednaka. Ako nisu, test pada.
   - **Primer**: assertArrayEquals(new int[]{1, 2, 3}, myArray);

Svaka od ovih assert metoda vraća **gresku u slučaju neuspeha** i u Eclipse-u se pojavljuje crvena traka koja ukazuje da neki test nije prošao. Zelena traka svakako znači uspeh, tj. da su sve assert provere ispunjene i da kod funkcioniše kako je očekivano. Međutim, kada je traka crvena ili plava, Eclipse prikazuje listu neuspešnih testova i opis grešaka ispod prozora za JUnit. Prikazuju se:

- **Naziv testa**: Ime metode koja je pala.
- **Razlog greške**: Detaljan opis greške, uključujući očekivanu i stvarnu vrednost (ako je u pitanju assert greška) ili opis izuzetka.



Primer Koda

U ovom primeru, kreiraćemo klasu **Calculator** koja sadrži osnovne aritmetičke operacije i zatim napisati JUnit test klasu CalculatorTest koja proverava rad svake metode.

Korak 1: Kreiranje Calculator Klase

```
public class Calculator {

    // Metoda za sabiranje
    public int add(int a, int b) {
        return a + b;
    }

    // Metoda za oduzimanje
    public int subtract(int a, int b) {
        return a - b;
    }

    // Metoda za množenje
    public int multiply(int a, int b) {
        return a * b;
    }
```

```
    // Metoda za deljenje; baca grešku ako je delilac 0
    public int divide(int a, int b) {
        if (b == 0) {
            throw new IllegalArgumentException("Ne možemo deliti nulom!");
        }
        return a / b;
    }
}
```

Kada vrednost **b** iznosi nula, funkcija izaziva **IllegalArgumentException** sa jasnom porukom "Ne možemo deliti nulom" umesto da dozvoli deljenje nulom, čime sprečava grešku. Ovaj izuzetak odmah zaustavlja izvršavanje metode i pruža poruku korisniku o uzroku greške.

Korak 2: Kreiranje JUnit Test Klase

Sada kreiramo test klasu **CalculatorTest** koja sadrži testove za svaku metodu u klasi Calculator.

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

public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calc = new Calculator();
        // Provera da li je zbir 2 + 3 jednak 5
        assertEquals(5, calc.add(2, 3));
    }

    @Test
    public void testSubtract() {
        Calculator calc = new Calculator();
        // Provera da li je razlika 3 - 2 jednaka 1
        assertEquals(1, calc.subtract(3, 2));
    }

    @Test
    public void testMultiply() {
        Calculator calc = new Calculator();
        // Provera da li je proizvod 2 * 3 jednak 6
        assertEquals(6, calc.multiply(2, 3));
    }
```

```
    @Test
    public void testDivide() {
        Calculator calc = new Calculator();
        // Provera da li je rezultat deljenja 6 / 3 jednak 2
        assertEquals(2, calc.divide(6, 3));
    }

    @Test(expected = IllegalArgumentException.class)
    public void testDivideByZero() {
        Calculator calc = new Calculator();
        // Provera da li metoda baca grešku kada je delilac 0
        calc.divide(1, 0);
    }
}
```

Zadatak za samostalni rad

Kreirati klasu User koja ima nekoliko polja (npr. ime, prezime, godine, i listu omiljenih aktivnosti). Zatim napisati JUnit testove za proveru različitih metoda koristeći različite assert metode.

- U Eclipse-u, napravite novi **Java Project** i nazovite ga, na primer, **UserProject**.
- Unutar projekta, kreirajte paket (package) pod imenom **user**.
- U okviru ovog paketa kreirajte klasu User, koja ima Stringove za ime, prezime, uzrast je tipa int, a lista aktivnosti može biti definisana kao **ArrayList<String>.**
- Dodajte **Junit** u projekat.
- U paketu **user** kreirajte test klasu i ispišite sledeće obavezne testove:
    1. **Testirajte ime korisnika**:

       Koristite assertEquals da proverite da li getFullName vraća pravilno formatirano ime.
    2. **Testirajte da li je punoletan**:

       Koristite assertTrue i assertFalse za testiranje metode isAdult.
    3. **Testirajte aktivnosti**:

       Koristite assertNotNull za proveru da lista omiljenih aktivnosti nije prazna nakon što dodate aktivnost.
    4. **Testirajte razne aktivnosti korisnika**:

       Dodajte nekoliko aktivnosti i koristite assertArrayEquals za poređenje tih nizova.
    5. **Test za neinicijalizovanog korisnika**:

       Koristite assertNull da proverite ponašanje kada je objekat neinicijalizovan.

Takođe, pored ovih testova, neophodno je dodati još pet svojih testova po želji. Za odbranu vežbe priložite projekat kao .zip datoteku i screenshot koji prikazuje da su svi testovi uspešno prošli.

Korisni resursi

Odličan „poligon za vežbanje“

https://www.w3resource.com/java-exercises/unittest/index.php

Tutorijal za pisanje testova:

https://www.vogella.com/tutorials/JUnit/article.html

Još jedan solidan tutorijal:

https://codeandwork.github.io/courses/java/junit.html