Testiranje

Testiranje predstavlja veoma važan proces u razvoju softvera koji podrazumeva ispitivanje ponašanja softvera i detektovanje odstupanja u odnosu na predviđene zahteve. Testiranje u širem smislu predstavlja sistem za proveru kvaliteta ne samo softvera nego i njegovih pratećih komponenti i karakteristika.

Pre samog testiranja se definiše vrsta testova i način na koji će se vršiti proces testiranja. Postoje automatski/poluautomatski testovi koji proveravaju funkcionalnost/sigurnost/opterećenje aplikacije po strategiji bele/crne kutije. Iz perspektive inžinjera-programera, tokom pisanja aplikacije ujedno se pišu i jedinični testovi koji proveravaju funkcionalnost manjih delova aplikacije.

JUnit je Javina biblioteka za jedinično testiranje koja je integrisana u *Eclipse* razvojno okruženje. Omogućuje poluautomatsko testiranje – testovi se ipak moraju napisati. *JUnit* testiranje se bazira na pisanju test klasa (engl. *TestCase*) koje se mogu grupisati u test grupe (engl. *TestSuite*). Jedna test klasa odgovara jednoj klasi čije funkcionalnosti se ispituju. Test klase se organizuju u dodatnom direktorijumu izvornog koda (npr. test) čija struktura paketa odgovara *src* direktorijumu.

Test klasa i grupa testova se kreira sledom sledećih akcija: *New ->Java ->JUnit*. U slučaju kreiranja test klase, otvara se forma za odabir klase koja se testira, kao i metode čija funkcionalnost se testira. Dodatno se generišu metode koje se pozivaju pre i posle testiranja unutar kojih se vrši priprema okruženja za test i dovođenje sistema u prvobitno stanje pre testiranja. Naziv test klasa se zadaje poštujući konvenciju davanja imena (obrazložena na prvim vežbama) sa tim što se na naziv klase koja se testira uglavnom dodaje naziv *Test*.

Primer 1. Test klasa sa generisanim okvirima metoda koje se pozivaju pre, tokom I nakon izvršavanja testiranja.

```
public class MainTest {
    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        //izvršava se na početku izvršavanja test klase
    }
    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        //izvršava se na kraju izvršavanja test klase
    }
    @Test
    public void testMainMethod1() {
        //test metode mainMethod1
        fail("Not yet implemented");
    }
    @Test
    public void testMainMethod2() {
        //test metode mainMethod2
        fail("Not yet implemented");
    }
}
```

Testiranje

U generisane okvire metoda se pišu testovi i propratni kod za pripremu scenarija testiranja. Procedura testiranja se bazira na instanciranju objekta klase koji se testira, pozivanju određenih metoda i proveri dobijenih rezultata u odnosu na očekivane. Provera rezultata se vrši uz pomoć *assert* metoda koje potvrđuju ili obaraju test, u zavisnosti od toga da li se dobijeni rezultat poklapa sa očekivanim. Neke od *assert* metoda iz biblioteke *org.junit.Assert* su date u nastavku:

- *assertTrue([message], boolean condition)*
- *assertEquals([String message], expected, actual)*
- *assertEquals([String message], expected, actual, delta)* - za double tip sa određenom tačnošću
- *assertNull([message], object)*
- *assertNotNull([message], object)*
- *assertSame([String], expected, actual)* – expected i actual bi trebalo da ukazuju na isti objekat
- *assertThat(object , Matcher matcher)* – potreban statički import *org.hamcrest.CoreMatchers.**

Za verziju 3 *JUnit*-a, neophodno je da testna klasa nasledi klasu *TestCase* kako bi koristila *assert* metode iz bibliotke *org.framework.Assert*. Od verzije 4 *JUnit*-a se koriste anotacije za pravljenje test klasa i nije neophodno nasleđivanje *TestCase* klase. A za korišćenje *assert* metoda bez prefiksa klase, potrebno je napisati statički import biblioteke *org.junit.Assert.**.

Primer 2. Grupa testova (engl. *TestSuite*) čiji ishod zavisi od rezultata više testnih klasa.

```
@RunWith(Suite.class)
@SuiteClasses({ MainTest1.class, MainTest2.class })

public class AllTests {  }
```

Napomena: Svaka funkcija bi trebala bar da ima *boolean* povratnu vrednost kao znak da je uspešno/neuspešno izvršena. Funkcije koje samo ispisuju podatke u konzolu nije potrebno testirati (bar na ovom kursu). Zato je neophodno modularno organizovati projekat kako bi npr. funkcije za prikaz (konzola, GUI) samo vršile ispis, ne i izmenu podataka.

Više detalja o JUnit testiranju se može naći u zvaničnoj dokumentaciji:

https://junit.org/junit4/javadoc/latest/

Primer 3. Jedan primer test klase za klasu *KorisnikManager*. Potrebno je testirati sve funkcije koje sadrži klasa.

```java
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertSame;
import static org.junit.Assert.assertTrue;

import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;

import models.ErrorCode;
import models.Korisnik;
import models.TipKorisnika;

public class KorisnikManagerTest {
    public static KorisnikManager km = KorisnikManager.getInstance();
    public static TipKorisnikaManager tm = TipKorisnikaManager.getInstance();

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        System.out.println("KorisnikManager test start.");

        tm.addTipKorisnika("tip1", true);
        tm.addTipKorisnika("tip2", true);

        km.addKorisnik("imenko", "prezimenic", "0123", true, "iprez", "pass123", "tip1");
        km.addKorisnik("pera", "peric", "4567", false, "test", "123pass", "tip2");
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        // oslobadjanje resursa i vracanje u zateceno stanje pre testa
        System.out.println("KorisnikManager test end.");
    }

    public void testAddKorisnik() {

    public void testReadFromFileString() {

    @Test
    public void testLogin() {
        Korisnik user = km.tryLogin("iprez", "pass123");
        assertTrue(user != null);

        user = km.tryLogin("iprez", "153");
        assertTrue(user == null);

        user = km.tryLogin("iprezaa", "pass123");
        assertTrue(user == null);
    }


    public void testEditKorisnik() {


    public void testGetKorisnik() {

    // ostale funkcije

}
```