Testno voden razvoj

Testiranje in kakovost

Kje smo?

- **Testiranje enot**
- Spoznali smo podatkovno strukturo sklad
  - Glavni operaciji na strukturi sta push in pop
  - Dodali smo še metodo isEmpty
- Ustvarili smo projekt Seznami v razvojnem okolju NetBeans
- Spoznali smo delo z JUnit
  - @Test označuje metodo, ki se mora pognati kot samostojen test v okviru orodja
  - @Test(expected = *„ime razreda“*.class) označuje pričakovano izjemo pri izvedbi
  - Delo z NetBeans
  - Delo z Junit

Današnje vaje

- Spoznali bomo nekaj dodatnih JUnit zmogljivosti
  - Označbe @Before, @After, @BeforeClass in @AfterClass
  - Označba @Ignore za izklopljene teste
  - Označba @Test(timeout=XX) za časovno omejitev testa
- Dodali bomo novo enoto v naš program
  - Uporabniški vmesnik
  - Zahteve zanj določijo uporabniki
  - Uporabniški vmesnik = izvajanje ukazne vrstice
- Testno voden razvoj programskih rešitev

Zahteve za uporabniški vmesnik

- **Dodajanje elementa**
  - Dve obliki ukaza: ***push niz*** ali **push *„niz z več besedami“***
  - ob uspešnem vnosu elementa naj se izpiše beseda "**OK**"
  - ob neuspehu naj se izpiše razlog za neuspeh
- **Odvzem vrhnjega elementa**
  - ukaz **pop**
  - ob uspešnem odvzemu elementa s sklada naj se element tudi izpiše
  - ob neuspehu naj se izpiše razlog za neuspeh
- **Popolna izpraznitev sklada**
  - je na voljo preko ukaza **reset**
  - po uspešni operaciji naj se izpiše beseda "**OK**"
  - ob neuspehu naj se izpiše razlog za neuspeh
- **Število elementov na skladu**
  - je na voljo preko ukaza **count**
  - po uspešni operaciji naj se izpiše število elementov na skladu
  - ob neuspehu naj se izpiše razlog za neuspeh

Nov razred SeznamiUV

- V programu NetBeans dodamo nov Javanski razred
- Načrt zanj določimo predvsem z željo lažjega testiranja

```
public class SeznamiUV
{
    public String processInput(String input)
    {
        return "";
    }
}
```

- Po vnosu novega razreda zanj še avtomatsko ustvarimo teste

Priprava na izvedbo testov

- Metoda setUp (označba @Before)
  - Kliče se pred vsakim testom
- Metoda tearDown (označba @After)
  - Kliče se za vsakim testom
- Metoda setUpClass (označba @BeforeClass)
  - Kliče se pred izvedbo prvega testa v razredu
- Metoda tearDownClass (označba @AfterClass)
  - Kliče se po izvedbi zadnjega testa v tem razredu

Osnovni test za push

- V testno vodenem razvoju najprej poskušamo preučiti možne vnose in njihove rezultate

```
@Test
public void testPushBasic() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testPushBasic");
    assertEquals("OK", uv.processInput("push Test1"));
    assertEquals("OK", uv.processInput("push Test2"));
}
```

Test za push več besed

- Teste lahko začasno onemogočimo preko označbe @Ignore

```
@Ignore("To be implemented at a later stage")
@Test
public void testPushMultipleWords() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testPushMultipleWords");
    assertEquals("OK",
        uv.processInput("push \"Test with multiple words\""));
    assertEquals("1", uv.processInput("count"));
    assertEquals("OK", uv.processInput(
                "push \"Another test with multiple words\""));
    assertEquals("2", uv.processInput("count"));
}
```

Test za push brez parametra

- Test „odkrijemo“ med razmišljanjem, kako naj bi se program obnašal

```
@Test
public void testPushNothing() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testPushNothing");
    assertEquals("Error: please specify a string",
                 uv.processInput("push"));
}
```

Preprost test za pop

```
@Test
public void testPopBasic() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testPopBasic");
    assertEquals("OK", uv.processInput("push Test1"));
    assertEquals("OK", uv.processInput("push Test2"));
    assertEquals("Test2", uv.processInput("pop"));
    assertEquals("Test1", uv.processInput("pop"));
}
```

Test za pop več besed

```
@Ignore("To be implemented at a later stage")
@Test
public void testPopMultipleWords() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testPopMultipleWords");
    assertEquals("OK", uv.processInput(
        "push \"Test with multiple words\""));
    assertEquals("OK", uv.processInput(
        "push \"Another test with multiple words\""));
    assertEquals("2", uv.processInput("count"));

    assertEquals("Another test with multiple words",
                 uv.processInput("pop"));
    assertEquals("1", uv.processInput("count"));

    assertEquals("Test with multiple words",
                 uv.processInput("pop"));
    assertEquals("0", uv.processInput("count"));
}
```

Test za pop na praznem skladu

```
@Test
public void testPopNothing() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testPopNothing");
    assertEquals("Error: stack is empty",
                 uv.processInput("pop"));
    assertEquals("Error: please specify a string",
                 uv.processInput("push"));
    assertEquals("Error: stack is empty",
                 uv.processInput("pop"));
}
```

Testi za vnos reset

```
@Test
public void testResetOnEmpty() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testResetOnEmpty");
    assertEquals("OK", uv.processInput("reset"));
}

@Test
public void testResetOnFull() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testResetOnFull");
    assertEquals("OK", uv.processInput("push Test"));
    assertEquals("OK", uv.processInput("reset"));
    assertEquals("Error: stack is empty",
                 uv.processInput("pop"));
    assertEquals("0", uv.processInput("count"));
}
```

Test za count na praznem skladu

```
@Test
public void testCountOnEmpty() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testCountOnEmpty");
    assertEquals("0", uv.processInput("count"));
}
```

Test za count z enim elementom

```
@Test
public void testCountOne() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testCountOne");
    assertEquals("OK", uv.processInput("push Test"));
    assertEquals("1", uv.processInput("count"));
}
```

Test za count dveh elementov

```
@Test
public void testCountTwo() {
    SeznamiUV uv = new SeznamiUV();
    System.out.println("testCountTwo");
    assertEquals("OK", uv.processInput("push Test1"));
    assertEquals("OK", uv.processInput("push Test2"));
    assertEquals("2", uv.processInput("count"));
}
```

Implementacija rešitve

- Po implementaciji testov implementiramo se rešitev

```
import java.util.Scanner;

public class SeznamiUV {
    Sklad<String> sklad;

    public SeznamiUV()  {
        sklad = new Sklad<String>();
    }

    public String processInput(String input) {
        ...
    }
}
```

Implementacija rešitve (nad.)

- Naloga prve rešitve je predvsem pokriti osnovne teste

```
public String processInput(String input) {
    Scanner sc = new Scanner(input);
    String token = sc.next();
    String result = "OK";
    switch (token) {
        case "push":
            sklad.push(sc.next());
            break;
        case "pop":
            result = sklad.pop();
            break;
        case "reset":
            while (!sklad.isEmpty())  sklad.pop();
    }
    return result;
}
```

Razrešitev napak pri testih

- Pri izvedbi so se pripetile napake v testih:
  - testPushNothing
  - testPopNothing
  - testResetOnFull
- Problem leži v (ne)obravnavanju izjeme NoSuchElementException
  - Metoda pop() je njen vir v naši programski kodi
  - En primer izjeme pa se zgodi tudi med uporabo Scannerja

Razrešitev napak pri testih (nad.)

```
switch (token) {
    case "push":
        if (sc.hasNext())
            sklad.push(sc.next());
        else
            result = "Error: please specify a string";
        break;
    case "pop":
         if (!sklad.isEmpty())
            result = sklad.pop();
         else
            result = "Error: stack is empty";
         break;
    case "reset":
        while (!sklad.isEmpty())
            sklad.pop();
        break;
}
```

Razrešitev neuspešnih testov

- Razlog za preostale neuspesne izvedbe testov je v manjkajoči metodi count
- Najprej popravimo uporabniški vmesnik SeznamiUV

```
...
case "count":
    result = String.format("%d", sklad.count());
    break;
...
```

Razrešitev neuspešnih testov (nad.)

- Napišemo še dejansko metodo count v razredu Sklad (prejšnji izziv)
- ... in poženemo teste

```
public int count()  {
    int result = 0;
    Element<Tip> temp = vrh;
    while (null != temp)  {
        ++result;
    }
    return result;
}
```

Neskončna zanka

- Če naredimo v metodi count napako, se naši testi nikoli ne končajo
- Preden resnično napako odpravimo, moramo popraviti tudi testne primere

```
@Test(timeout=100)
public void testCountOne() {
    ...
}

@Test(timeout=100)
public void testCountTwo() {
    ...
}
```

- @Test(timeout=X) lahko vedno nadomesti @Test

Popravek v metodi count

Sklad.java

```
public int count()
{
    int result = 0;
    Element<Tip> temp = vrh;
    while (null != temp)  {
        ++result;
        temp = temp.vezava;
    }
    return result;
}
```

Polepšava naših testov

- Ob pregledu testov opazimo, da se v vsakem testu pojavi vrstica
  - SeznamiUV uv = new SeznamiUV();
- V takih primerih lahko uporabimo metodo setUp

```
@Before
public void setUp() {
    uv = new SeznamiUV();
}
```

- Prednost je predvsem v tem, da sedaj test ne vsebuje predpriprave podatkov za izvedbo testa
- Za zaključek počistimo še testne izpise
- **Alternativa:** sklad naredimo le enkrat (@BeforeClass), pred izvedbo vsakega testa (@Before) ali po izvedbi vsakega testa (@After) ga izpraznemo

Izziv 3

- Dopolnite primer z uporabniškim vmesnikom za ukaza **top niz** in **search niz**, seveda skupaj z ustreznimi testi (vmesnika in sklada).

  Metoda **top** preveri, ali se podani element **niz** nahaja na vrhu sklada; metoda vrne „**OK**“ ali ustrezno napako:„**Error: wrong element**“, „**Error: stack is empty**“, “**Error: please specify a string**“

  Metoda **search** podani element **niz** poišče v skladu in vrne njegovo mesto oziroma -1, če podanega elementa ni v skladu. Če metoda ni ustrezno klicana vrne “**Error: please specify a string**“

- Dopolnite primer tako, da se bosta testa **testPushMultipleWords()** in **testPopMultipleWords()** uspešno izvedla