Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

**Igor Rožanc**

# Osnove algoritmov in podatkovnih struktur I

# (OAPS I)

**2. letnik, VSP Računalništvo in informatika, vse smeri**

**PROSOJNICE ZA 10. PREDAVANJA (7.12.2006)**

Študijsko leto 2006/07

## Sklad (razred `Stack<E>`)



**Dva primera:**

**a) Računanje aritmetičnih izrazov v postfiksni obliki (brez oklepajev)**

- najprej zapišemo operanda, nato operator: `3 4 +` pomeni `3 + 4`
- **Postopek:**
  - beremo od leve proti desni
  - če naletimo na operand, ga postavimo na sklad
  - če naletimo na operator, vzamemo dva operanda s sklada, izvedemo operacijo in rezultat zapišemo na sklad
- **Prikaz izračuna ...**
- **Realizacija metode v Javi: razred `Kalkulator`**
- **Primer: Kalkulator.java**

**b) Odprava rekurzije s pomočjo sklada (hanojski stolpiči)**

- **Opis problema:** n obročev različnih velikosti in 3 palice (A, B, C)
  - **Začetek:** - vsi obroči so na palici A
    - zloženi so od največjega do najmanjšega
  - **Cilj:** prestaviti obroče na palico C s pomočjo pomožne palice B
  - **Omejitvi:** - naenkrat lahko prestavimo en obroč
    - nikoli ne smemo prestaviti večji obroč na manjšega
- **Narava problema je rekurzivna:**
  - prestavi (n-1) obročev s palice A na palico B
  - prestavi en obroč s palice A na palico C
  - prestavi (n-1) obročev s palice B na palico C





**Rekurzivna rešitev:**

- **Prikaz postopka ...**
- **Rekurzivna rešitev v Javi: razred `HanoiRek`**
- **Primer: HanoiRek.java**

**Iterativna rešitev s skladom:**

- **Postopek:**
  - na sklad postavimo začetni zahtevek
  - s sklada jemljemo zahtevke in jih obdelujemo
    (pri tem se na sklad dodajajo novi zahtevki)
  - postopek ponavljamo, dokler sklad ni prazen

## Sklad (razred `Stack<E>`) 58

- **Grob opis postopka:**

```
postavi začetni zahtevek na sklad;
dokler sklad ni prazen
{   odvzemi zahtevek s sklada;
    obdelaj zahtevek; }
```

- **Prikaz postopka ...**
- **Predstavitev zahtevka:** razred `Zahtevek`

  **Štiri atributi:**

  n – število obročev, ki jih je treba prestaviti

  a – izvor

  b – pomožna palica

  c - ponor
- **Iterativna rešitev v Javi: razred `HanoiIte`**
- **Primer: HanoiIte.java**



## Ostale značilnosti Jave (1.)5.0 59

**Avtomatska pretvorba ovojnih tipov - "autoboxing" in "unboxing":**

```
Integer I1 = new Integer(5); // pred Javo 5.0
Integer I2 = 5;    // autoboxing
Number N = 2.2f;
int i1 = I1;       // unboxing
Integer I3 = null;
int i2 = I3;       // NAPAKA!!
```

**Krajši zapis zanke for (za tabele in zbirke):**

```
int[] t = {1,2,3,4,5,6,7,8,9,10};
for (int i : t) {          // zadaj se skriva iterator!!
    System.out.println(i);}
List sez = Arrays.asList(t);
for (Object i : sez) {...}
```

V svojem razredu moramo definirati ustrezen generični iterator ...

### Naštevni tipi:

```
public enum Ocena = {5,6,7,8,9,10,BREZ};
class Student
{   ...
    Ocena oc1=Ocena.BREZ;
    Ocena oc2=Ocena.8;
    ... }
```

### Nedoločeno število argumentov - “varargs”:

```
public void izpis(String s1, String...ostali);
{
    System.out.println(s1);
    for (String i:ostali)
            System.out.println(i);
    ... }
```

### Oblikovanje izpisa – `printf(format,args)`:

```
System.out.printf("%4d %5.3f %-6s",15,3.14159,"Haha");
```





### Statični import stavek:

```
import static java.lang.System.out;
public class Test {
    public static void main(String[] args) {
            println("Dober dan!");
    }
}
```

### Dopolnjen razred `Arrays`:

```
int[][] t1={{1,2,3},{4,5,6},{7,8,9}};
int[][] t2={{1,2,3},{4,5,6},{7,8,9}};
System.out.println(Arrays.deepToString(t1));
if(Arrays.deepEquals(t1,t2))
    {...}
System.out.println(Arrays.deepHashCode(t1));
```