Programiranje I – RIN
Računalništvo I – MA

Dogodki

12. predavanje

Dogodek

- Kaj so dogodki?
  - vrata se odprejo
  - luč se prižge
  - prične se nova minuta
  - miška se premakne
  - čoln je prišel na drugi breg
  - prišlo je do napake pri deljenju
- Kaj lahko povemo o teh dogodkih?
  - kdaj se lahko zgodijo v toku programa? kadarkoli?
  - kaj pa zadnji dogodek?

Prekinitve

- Prekinitve (*interrupt*) so posebne vrste dogodkov v računalništvu
- Prožijo se nenapovedano (asinhrono) in sicer takrat, ko se je nekaj zgodilo:
  - v računalniku - *zmanjkalo je elektrike*,
  - v operacijskem sistemu - *končalo se je neko opravilo* ali
  - v okolju izvajanja programa - *nekdo je premaknil miško*

Dogodki v računalništvu

- Poznamo prekinitve, ki jih povzroči **strojna oprema** in prekinitve, ki jih povzroči **programska oprema** (*hardware* in *software interrupts*)
- Strojeve prekinitve: premik miške; pritisk na tipko; minila je nova milisekunda; po omrežju je pripotoval nov podatek (paket) itd.

Dogodki v računalništvu

- Programske prekinitve: zgodilo se je deljenje z nič, zmanjkalo je pomnilnika; nekdo je poskušal v polju dostopati do elementa, ki ne obstaja; itd.
- Posebna vrsta programskih prekinitev so sporočila o posebnih (izjemnih) situacijah - *exceptions*

Kaj narediti ob dogodku

- Ko se dogodek zgodi, se je potrebno nanj **odzvati**
- Pravimo, da je potrebno dogodek obdelati, oziroma z njim **rokovati**
- Zato pravimo funkcijam, objektom, itd., ki obdelajo dogodek **rokovalniki**

Rokovanje dogodkov

- Vsak dogodek, lahko rokuje en ali več rokovalnikov - pravimo, da se njihovo izvajanje **veriži**
- Rokovalnik najprej **priklopimo** na dogodek (namestimo - *install*) in odslej se bo vsakič, ko se bo dogodek zgodil, pognal rokovalnik zanj
- Rokovalnik lahko tudi **odklopimo** (odstranimo - *uninstall*)

Kako izgleda rokovalnik

- Rokovalnik je ponovno nič drugega kot **predmet**, ki ima posebne lastnosti (*metode*)
- Osnovna lastnost je, da ima metodo, ki zna rokovati z dogodkom - *obdelaj*
- Dogodku prijavimo predmet rokovalnik in, ko se dogodek zgodi, se pokliče metoda *obdelaj*

Rokovalnik

```
public interface rokovalnik {
  public void obdelaj(void);
} // rokovalnik
                              ...
public class rokovDogodka implements rokovalnik{
  public void obdelaj(void) {
    System.out.println("zgodil se je dogodek");
  } // obdelaj
} // rokovDogodka
                              ...
rokovDogodka rokov= new rokovDogodka();
dogodek.install(rokov);
```

Rokovalnik izjemne situacije

```
try {
   ...;
} catch (Sporocilo sporocilo) {
   System.out.println("zgodil se je " +
                      sporocilo);
};
```

Primer - tok vhodnih podatkov

- Imeli bomo razred *bralec*, ki bo bral črke z vhodnega toka
- Naj bodo črke iz neke znane množice *A*, ki jo imenujemo **abeceda**
- Dogodki, ki jih razred *bralec* razlikuje so posamezne prebrane črke
  - naj bo *A* = {*0, 1, 2*}
  - potem so dogodki: prebral 0, prebral 1, ...
- Za vsakega od dogodkov lahko namestimo rokovalnik

Razred *bralec* in dogodki

- Da poenostavimo nameščanje rokovalnikov ob različnih dogodkih, imamo samo eno namestitveno metodo:

```
void install(rokovalnik rokov, char crka);
```

  - ki namesti rokovalnik *rokov* na dogodek, da je bila prebrana črka *crka*
- Razred ima še metodo *beri*, ki sproži branje toka vhodnih podatkov
- Ob tvorbi lahko navedemo ime vhodnega toka podatkov (datoteke)
  - če je ta *null*, se bere s standardnega toka

Razred *bralec*

```
public interface bralec {

  public bralec (String ime);
  public void install(rokovalnik rokov, char crka);
  public void beri();

}; // bralec
```

Preštejmo *a*-je

- S pomočjo razreda *bralec* želimo prešteti število črk *a* v vhodnem toku podatkov
- Definiramo razred *stejA*, ki:
  - je udejanjitev vmesnika *rokovalnik*
  - rokovalniška metoda *obdelaj* ob vsakem klicu poveča števec *stev* za 1
    - metoda bo klicana, ko bo bralec prebral črko *a*
  - ima dodatno metodo *stevilo*, ki vrne trenutno vrednost števca *stev*

Razred *stejA*

```
public class stejA implements rokovalnik {

  private int stev= 0;

  public      stejA()    { stev= 0;      }
  public void obdelaj()  { stev++;       };
  public int  stevilo()  { return stev; };
} // stejA
```

Uporaba *stejA*

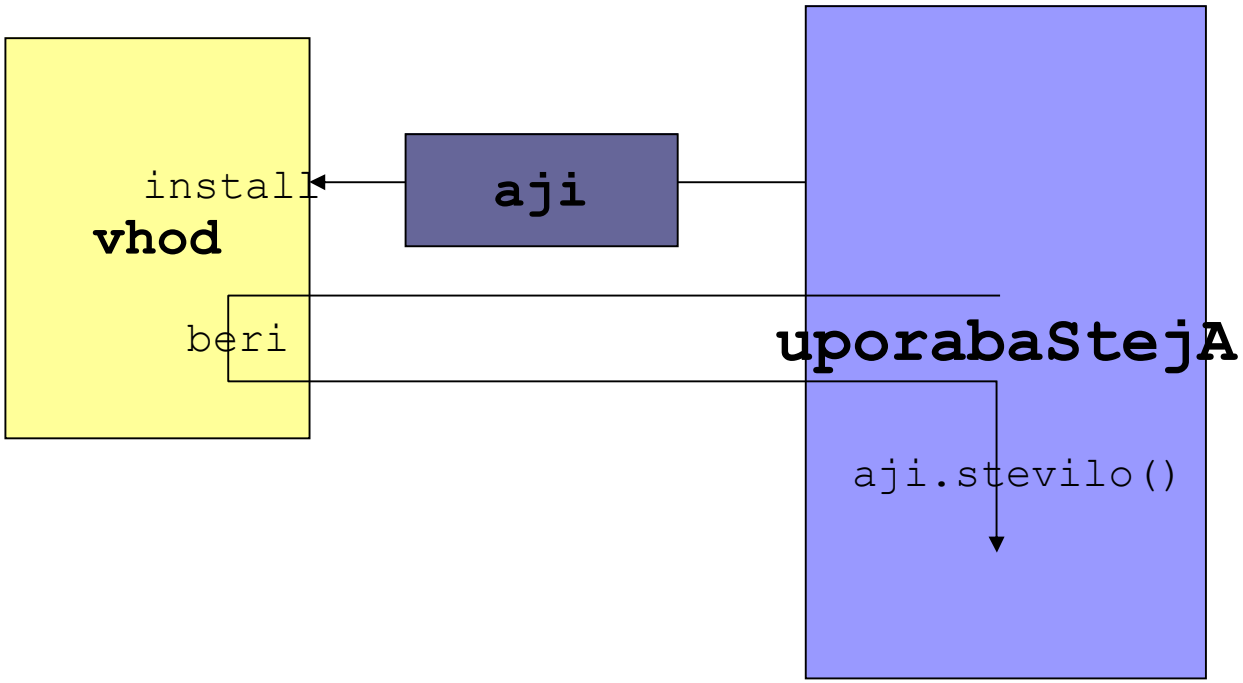
```
public class uporabastejA {

  public static void main (...) {
    stejA  aji=  new stejA();
    bralec vhod= new bralec(null);

    vhod.install(aji, 'a');
    vhod.beri();
    System.out.print("Stevilo a-jev je: ");
    System.out.print( aji.stevilo() );
    System.out.println();
  } // main

} // uporabastejA
```

Arhitektura sistema



Izrazoslovje

- Rokovalniku se reče tudi *handler* ali *call-back routine*
- Rokovalnik namestimo na dogodke in vsak dogodek ima svojo ročko/kljuko (*handle*)
- Metoda (funkcija) rokovalnika, ki se pokliče (sproži) ob prekinitvi je posluževalnik
- Če poslužujemo strojno opremo, je rokovalnik del programja, ki se imenuje gonilnik (*driver*)

Zaključek

- Tehniki, ki smo si jo ogledali, rečemo ***dogodkovno gnano programiranje***.
- Je osnova programiranju sistemov, ki delujejo v realnem času:
- Ko se nekaj zgodi, se (takoj) odzovemo na dogodek.
- Dogodek povzroči reakcijo
- In reakcija je lahko ponovno dogodek za naslednjo reakcijo …
- **Izziv:** kako se lotiti tega, da bosta dva rokovalnika lahko sodelovala?

Povzetek

- Dogodki
- Prekinitve
  - strojeve
  - programske
- Odzivanje na dogodke = rokovanje
  - rokovalnik
  - primer rokovalnika – analiza toka vhodnih podatkov