Programski jezik Java

Interno gradivo za predmet

Algoritmi in programski jeziki (4. letnik)

Izjeme – razlaga A

(neprečiščeno besedilo)

bostjan.vouk@tsc.si

Izjeme

- Prestrezanje in obravnava izjem
- Izjema predstavlja dogodek, kateri lahko nastopi med izvajanjem programa in zmoti predviden tok izvajanja.
- Nek dogodek lahko proži vrsto izjem zato je v določenih primerih pomemben tudi vrstni red prestrezanja izjeme.
  - Na primer psevdokod v nadaljevanju obravnava odpiranje datoteke.

bostjan.vouk@tsc.si

Kaj so izjeme?

- Izjeme so posebni dogodki, ki se sprožijo ob izvajanju programa, kadar v programu pride do napake. Izvajanje programa se ob tem prekine, kar pa ponavadi ni zaželeno.
- ***Kako izjeme prestrežemo?***
- Obravnavanja izjem se lahko lotimo na dva načina.
  a) V metodi, ki lahko sproži izjemo, v deklaraciji to »povemo« z določilom **throws** in s tem prenesemo odgovornost naprej.
  b) S konstruktom:

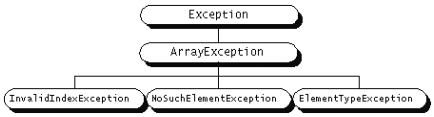
```
try{
  A;
}

catch(Imeizjeme i){
  B;
}
```

V konstruktu se zgodi naslednje: Izvede se stavek A in če se med izvajanjem stavka A sproži izjema Imeizjeme, se izjema ujame in se izvede stavek B.

bostjan.vouk@tsc.si

Izjeme

- Izjeme so razporejene v hierarhijo. V sami dokumentaciji posamezne operacije oziroma razreda moramo biti pozorni na opis izjem, ki jih posamezna operacijo lahko proži. Take izjeme moramo prestrezati in ustrezno obdelati.



Primer uporabe izjeme

- Primer branja tekstovne datoteke
- public static void main(String[] args) {

```
try {
        FileReader f = new FileReader("test.txt");
        BufferedReader out = new BufferedReader(f);
        String vrstica = out.readLine();
        while ( vrstica != null ) {
                System.out.println(vrstica);
                vrstica = out.readLine();
        }
} catch (FileNotFoundException e) {
        System.out.println("Datoteke ni bilo moc odpreti");
        e.printStackTrace();
} catch (IOException e) {
        System.out.println("Iz datoteke ni moc brati");
        e.printStackTrace();
}
}
```


Lastnosti

- Datoteko z lastnosti lahko ustvarimo tudi sami. V naslednjem primeru naredimo preprosto tekstovno datoteko v katero smo vpisali pare (atribut, vrednost). Primer v nadaljevanju prikazuje primer uporabe take datoteke lastnosti:
- vsebina datoteke mojelastnosti.txt
- user=janez
- barva.ozadje=bela
- barva.tekst=crna

Primer

- aplikacija v javi, ki prebere in izpiše lastnosti:
- try {
  FileInputStream f = new
  FileInputStream("mojelastnosti.txt");
  Properties p = new Properties();
  p.load(f);
  System.out.println(p.getProperty("user"));
  System.out.println(p.getProperty("barva.ozadje"));
  System.out.println(p.getProperty("barva.tekst"));
  } catch (Exception e) {
  e.printStackTrace();
  }

Primer

- readFile {
  try {
  open the file;
  determine its size;
  allocate that much memory;
  read the file into memory;
  close the file;
  } catch (fileOpenFailed) {
  doSomething;
  } catch (sizeDeterminationFailed) {
  doSomething;
  } catch (memoryAllocationFailed) {
  doSomething;
  } catch (readFailed) {
  doSomething;
  } catch (fileCloseFailed) {
  doSomething;
  }
  }

Obravnava izjem

- V programih lahko pride do napak
  - resne okvare strojne opreme
  - preproste programerske napake
  - (indeks izven meja)
- dober program je pripravljen na izjemne dogodke
- jezik lahko delo poenostavi

Nekaj primerov napak – izjemnih dogodkov

- SW napake
  - deljenje z nič
  - poskus dostopa do elementa izven indeksa
  - odpiranje neobstoječe zbirke
  - branje po EOF
  - prekinitev read/write operacije
  - proženje metode neobstoječega razreda
  - nepravilna pretvorba med tipi pri razredih
  - suspend ali resume že zaključene niti
  - klic metode objekta z null referenco
  - posredovanje nenumeričnega niza objektu Integer/Float
  - določitev negativne velikosti polja
- HW napake
  - disk crash
  - out of memory

bostjan.vouk@tsc.si

Tipi izjem

- Napake (Error)
  - za Java VM in HW probleme kot so
  - InternalError, OutOfMemoryError ipd.
  - teh napak običajno ne prožimo, niti jih ne lovimo
- Izjeme (Exception)
  - so “pričakovane” oz. najavljene izjeme
  - oz. morebitne težave, ki jih pričakujemo,
  - običajno ni izpolnjen kakšen predpogoj
  - (ni zbirke, ki jo želim odpreti)
- RT izjeme (RuntimeException)
  - tipično programske napake, ki običajno kažejo
  - na “hrošče” npr. indeks polja izven ...

bostjan.vouk@tsc.si

Proženje izjeme

- Ko se pojavi izjema znotraj metode, se kreira ustrezni objekt (exception object), ki vsebuje informacijo o izjemi, vključno s tipom in stanjem programa, ko je izjema nastopila.
- XYZException
  - FileNotFoundException obj
  - ArrayIndexOutOfBoundsException obj

bostjan.vouk@tsc.si

Izjeme

- Ko se pojavi izjema, metoda kreira objekt izjemo in ga preda RunTime sistemu.
- Objekt izjema vsebuje informacije o izjemi, tipu in stanju programa ob pojavu izjeme.
  - throwing
- RT sistem je odgovoren, da najde kodo, ki bo izjemo obdelala.
  - handling
- Sledenje v skladu s klici.
- Če ne najde, se izvajanje RT in posledično Java programa zaključi

Stavki za obvladovanje izjem

- "exception" je izjemni dogodek, ki prekine normalni tok izvajanja
- **try -** *definira blok kode, za katerega želimo obvladovati izjeme*
- **catch -** *definira izjemo, ki jo želimo obvladati in definira blok kode, ki za to poskrbi*
- **throw -** *preklopi izvajanje na ustrezni "handler" izjem*
- **finally -** *definira blok kode, ki se izvrši ne glede na to, ali se je izjema pojavila ali ne*

Princip obravnave izjem

- **try {**
  // V splošnem se ta del programa izvede normalno. Lahko pa pride do izjeme ali
  // prekinitve s stavkom break, continue ali return.
  }
  catch (Tip_izjeme Izjema_1) {
  // Obravnava izjeme Izjema_1 tipa Tip_izjeme ali podrazreda tega tipa
  }
  catch (Tip_izjeme Izjema_2) {
  // Obravnava izjeme Izjema_2 tipa Tip_izjeme ali podrazreda tega tipa
  }
  finally {
  // Koda se izvede vsakič, ko zapustimo del označen s try ne glede na //način.
  // Ločimo več vrst zaključitev izvajanja:
  // normalno, ko dosežemo konec bloka
  // z izjemo, ki smo jo ulovili s stavkom catch
  // z izjemo, ki je nismo ujeli
  // zaradi stavka break, continue ali return
  }

vsak try ima vsaj en catch ali finally blok

Obravnavanje izjem

- Načeloma lahko obravnavamo več izjem naenkrat (ni pa pametno - hitro namreč "obvladamo" še kaj drugega)
- try {
  ...
  } catch (Exception e) {...}

- Bolje:
- try {
  ...
  } catch (IOException e1) {...}
  catch (ArrayIndexOutOfBounds e2) {...}

Pomembne metode

- catch (Throwable ime)
- Throwable ima dva podrazreda
  - Exception in Error
- getMessage() → *java.lang.ArrayIndexOutOfBoundsException: at Prepisi2.main(Prepisi2.java:14)*
  - Sporočilo
  - Razred Exception jo podeduje iz razreda Throwable. Ta metoda nam vrne podroben opis izjeme.
- printStackTrace()
  - nit klicev
- toString()
  - kratko sporočilo
  - ki je tudi podedovana iz razreda Throwable in nam vrne kratek opis izjeme.

Izjema je objekt

- Izjema v Javi je objekt
- Vse izjeme so potomke izjeme java.lang.Throwable
- Nekatere skupne metode: Primer: int i=3/0;
  - public String **getMessage()**
    - / by zero
  - public String **toString()**
    - java.lang.ArithmeticException: / by zero
  - public void **printStackTrace()**
    - java.lang.ArithmeticException: / by zero at Try.main(Try.java:5)

Najava proženja izjem

- Vse izjeme, razen podrazredov ***RuntimeException in Error*** imenujemo preverjane/označene/ najavljene izjeme (checked exceptions). Metoda, ki jo proži, mora to izjemo deklarirati s pomočjo **throws** v deklaraciji metode.
- RTE in Error ni potrebno loviti in niso dokumentirane v vmesniku metod
- Klicatelj mora vedeti, katere izjeme lahko pričakuje in mora biti sposoben reagirati - imam dve možnosti: (***try, catch***) ali pa metoda deklarira proženje te izjeme (prenese odgovornost naprej).

RTE exception - primer

```
public class ExceptionalClass {
public void method1() throws CheckedException {
        // ... some code
        throw new CheckedException( "something bad happened" );
}
    public void method2( String arg ) {
        if( arg == null ) {
            throw new NullPointerException( "arg passed to method2 is null" );
}
}
public void method3() throws CheckedException {
    method1();
}
}
```


RTE exception - primer

- ***(Nepravilen) klic metode method2(String x)***

```
public static void main( String [] args ) {
    ExceptionalClass example = new ExceptionalClass();
    try
    {
        example.method1();
        example.method3();
    }
    catch( CheckedException ex ) {
    }
    example.method2( null );
}
```

Razlika med throw in throws

throw

- označuje del, ki sproži izjemo
- **sintaksa:** *throw izraz*
- izraz je primerek razreda *Throwable* ali njegovega podrazreda

throws

- za metodo označi možne izjeme
- ne vključuje napak (Error) in izjem v času izvajanja (RuntimeException-RTE)

Primer z določilom throws

```
public class PrelaganjeOdgovornosti{
  //odgovornost se preloži na Java virtual machine
 public static void main(String[] args) throws Exception{
   metodaKiSprozilzjemo();
 }
  //prelozimo odgovornost tja, kjer bo metoda klicana.
 public static void metodaKiSprozilzjemo()throws Exception{
  String niz = "pomlad";
  char znak = niz.charAt(6);          //šesti znak v nizu ne obstaja, sproži se izjema
 }
}
```

Izpis:

```
> java PrelaganjeOdgovornosti
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 6
  at java.lang.String.charAt(Unknown Source)
  at PrelaganjeOdgovornosti.metodaKiSprozilzjemo(PrelaganjeOdgovornosti.java:11)
  at PrelaganjeOdgovornosti.main(PrelaganjeOdgovornosti.java:5)
>
```


Primer

- Primer izjeme ***NullPointerException***.
-

```
public class NullPointerException{
  public static int[] tabela;
  public static void main(String[] args)throws Exception{
    izjema();
  }
  public static void izjema() throws Exception{
    int stevilo = tabela[0];
  }
}
```