Programski jezik Java

Interno gradivo za predmet

Algoritmi in programski jeziki (4. letnik)

Izjeme – razlaga B

(neprečiščeno besedilo)

bostjan.vouk@tsc.si

Sistem izjem v Javi

- Pregled
- Ozadje
  - Napake
  - Izjeme
- Podpora v Javi
  - Predstavitev izjem
  - Kreiranje & obravnavanje izjem
  - Načrtovanje & uporaba izjem

bostjan.vouk@tsc.si

Programske napake

- Sintaksne napake
  - Napake pri pisanju kode (slovnica, tipi)
  - Prepoznavanje med prevajanjem
- Napake v času izvajanja
  - Nelegalne operacije, ne dajo se izvršiti
  - Odkrivanje med izvajanjem programa
  - Obravnavane so kot izjeme v Javi
- Logične napake
  - Operacije vodijo v nepravilno stanje programa
  - Preverjanje in generiranje izjem
  - Lahko in tudi ne vodijo v napake v času izvajanja
  - Odkrivamo z razhroščevanjem

bostjan.vouk@tsc.si

Izjeme

- Redek dogodek izven običajnega obnašanja programa
- Primeri
  - Deljenje z nič
  - Dostop izven meja polja
  - Prekoračitev spomina
  - Napačen format vhoda
  - Napaka pri pisanju v datoteko
  - Manjkajoča vhodna datoteka

Obravnavanje izjem

- Izvajane akcij za izjeme
- Primeri
  - Izhod iz programa (abort)
  - Ignoriranje izjeme
  - Obravnavaj izjemo in nadaljuj
    - Izpiši napako
    - Branje novih podatkov
    - Ponovno poskusi

Obravnavanje izjeme – Problem

- Izjeme ni mogoče obravnavati lokalno
  - Ni zadosti podatkov za obravnavanje izjeme lokalno
  - Obravnava izjeme potrebuje več podatkov
- Obravnavanje izjeme na višjih nivojih
  - Metoda, ki je sprožila klic
  - Odločitev na nivoju aplikacije
  - Primeri
    - Nepravilen format podatkov → zahtevaj ponoven vnos
    - Datoteke ni mogoče odpreti → zahtevaj novo ime datoteke
    - Ni prostora na disku → vprašaj po brisanju datotek

Obravnava izjem – Prekinitev

- Pristop
  - Izhod iz programa s sporočilom o napaki / kodi napake
- Primer
  - **if (error) {**
    System.err.println("Napaka!"); // message
    System.exit(1); // error code
    }
- Problem
  - Drastična rešitev
  - Dogodek obravnava uporabnik s ponovnim klicem programa
  - Program ne zna obravnavati nekaterih izjem

bostjan.vouk@tsc.si

Obravnava izjem – Koda napake

- Pristop
  - Program vrne vrednost → koda napake
- Primer
  - **A( ) { if (error) return (-1); }**
  - **B( ) { if ((retval = A( )) == -1) return (-1); }**
- Problemi
  - Kličoča funkcija mora preveriti in procesirati napako
    - Lahko preprosto pozabi obravnavati napako
    - Lahko vrne kodo napake navzgor k klicatelju
  - Potreben je dogovor o pomenu kode napake
  - Koda za obravnavanje napake pomešana z normalno kodo

bostjan.vouk@tsc.si

Obravnavanje izjem – Vrzi izjemo

- Pristop
  - Vrzi izjemo
- Primer
- A( ) {
  if (error) throw new ExceptionType();
  }
  B( ) {
  try {
  A( );
  }
  catch (ExceptionType e) { ...action... }
  }

Izjema se vrača k kličoči funkciji dokler ne najde primeren catch blok

bostjan.vouk@tsc.si

Obravnavanje izjeme – Vrzi izjemo

- Prednosti
  - Prevajalnik zagotavlja, da se izjema ulovi
  - Ni potrebe po prepuščanju izjeme kličoči funkciji
    - Vrnitev se izvrši avtomatično
  - Hierarhija razredov definira pomen izjeme
    - Ni potrebe po ločenih definicijah kode za napake
    - Koda za obravnavanje izjem je ločena

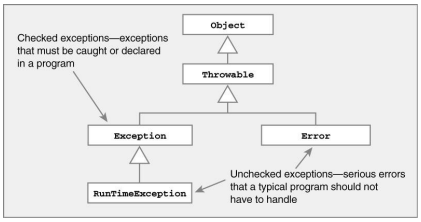
Predstavitev izjem

- Izjeme so predstavljene kot
  - Objekti, ki izvirajo iz razreda Throwable
- Koda
- **public class Throwable( ) extends Object {**

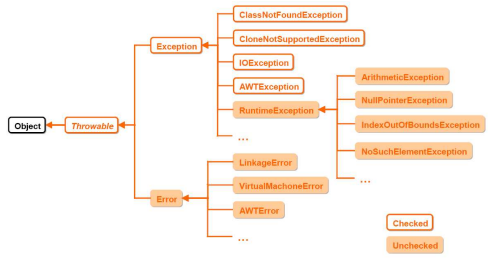
```
Throwable( )                        // No error message
Throwable( String mesg )            // Error message
String getMessage()                 // Return error mesg
void printStackTrace( ) { … }       // Record methods
…                                   // called & location
}
```


Predstavitev izjem

- Hierarhija razredov Exception
  - Dva tipa izjem → preverjene & nepreverjene



Hierarhija razredov Exception



Izjeme – Primeri

- FileNotFoundException (java.io)
  - Zahteva po odprtju datoteke ne uspe
- IllegalArgumentException (java.lang)
  - Metoda je podala nelegalen / neprimeren argument
- IOException (java.io)
  - Generična I/O napaka
- NullPointerException (java.lang)
  - Namera dostopa do objekta null
- UnsupportedOperationException (java.lang)
  - Objekt ne vsebuje zahtevane operacije

Nepreverjene izjeme

- Class Error & RunTimeException
- Resne napake, ki jih program ne zna obravnavati
- Običajno nakažejo logične napake
- Primeri
  - **NullPointerException, IndexOutOfBoundsException**
- Lovljenje nepreverjene izjeme je opcijsko
- Obravnavana v JVM, če je ulovljena

Preverjene izjeme

- Class Exception (razen RunTimeException)
- Napake, ki bi jih tipično morali ujeti programi
- Metode lahko ujamejo napake
- Primeri
  - IOException, ClassNotFoundException
- Zahteva prevajalnika "ujemi ali deklariraj"
  - Ujemi in obravnavaj izjemo v metodi,

ALI

  - deklariraj, da metoda lahko vrže izjemo
  - s tem silimo kličočo funkcijo, da deklarira izjemo
  - Primer
    - void A( ) throws ExceptionType { … }

bostjan.vouk@tsc.si

Generiranje & obravnavanje izjem

- Java stavki
  - **try**
  - **throw**
  - **catch**
  - **finally**
- Recept za uporabo izjem
  - Objami kodo, ki generira izjemo z blokom **try**
  - Uporabi **throw za generiranje izjem**
  - Uporabi **catch za rokovalnik izjem**
  - Uporabi **finally za akcije po izjemi**

bostjan.vouk@tsc.si

Sintaksa

```
try {                          // try block encloses throws
throw new eType1();            // throw jumps to catch
}
catch (eType1 e) {             // catch block 1
    ...action...               // run if type match
}
catch (eType2 e) {             // catch block 2
    ...action...               // run if type match
}
finally {                      // final block
    ...action...               // always executes
}
```

bostjan.vouk@tsc.si

Stavek try

- Formira blok **try**
- Objame stavke, ki lahko vržejo izjemo
- Področje bloka **try je dinamično**
- Vsebuje kodo, ki se izvaja znotraj metod klicanih v bloku **try** (in njihovimi nasledniki)

bostjan.vouk@tsc.si

Stavek try

- Primer

```
try {          // try block encloses all exceptions in A & B
   A( );       // exceptions may be caught internally in A & B
   B( );       // or propagated back to caller's try block
}
void A( ) throws Exception {        // declares exception
   B( );
}
void B( ) throws Exception {        // declares exception
   throw new Exception( );          // propagate to caller
}
```

bostjan.vouk@tsc.si

Stavek throw

- Pove, da se je zgodila izjema
- Običajno definira eno izjemo
  - Objekt je primerek razreda Exception
- Ko je vržena izjema
  - Kontrola se nahaja v bloku **try**
  - Izvajanje se nadaljuje na najbližjem rokovalniku izjem za blokom **try**
  - Preverijo se pogoji za rokovalnike
  - Prvi rokovalnik, ki uspe s pogojem se izvede
  - Izvede se koda bloka **finally**

bostjan.vouk@tsc.si

Stavek catch

- Nahaja se za **try blokom**
- Definira kodo, ki se izvaja za izjemo
  - Koda v bloku **catch → rokovalnik izjem**
- Blok **catch na začetku vsebuje pogoj za** izvajanje rokovalnika
- Lahko uporabimo več blokov **catch za en sam try blok**
  - za procesiranje izjem različnih tipov
  - bloki **catch se procesirajo po vrsti**
  - nadrazred lahko prevzame blok **catch za** podrazrede, če se blok **catch v nadrazredu pojavi** najprej

bostjan.vouk@tsc.si

Stavek catch

- Primer
- class eType1 extends Exception { … }

```
try {
  … throw new eType1( ) …
}
catch (Exception e) {          // Catch block 1
  ...action...                 // matches all exceptions
}
catch (eType1 e) {             // Catch block 2
  ...action...                 // matches eType1
}                              // subsumed by block 1
                               // will never be executed
```

bostjan.vouk@tsc.si

Konstrukt try catch

- ***Primer:***
-
```
public class TryCatch{
  public static void main(String[] args){
    try{
      String niz = args[0];
      System.out.println("Program se je izvedel brez izjem.");
    }
    catch(IndexOutOfBoundsException i){
      System.out.println("Pozabil si podati niz.");
    }
  }
}
```

- ***Izpis:***
-
```
> java TryCatch
Pozabili ste podati niz.
>
```

-
```
> java TryCatch niz
Program se je izvedel brez izjem.
>
```

bostjan.vouk@tsc.si

Stavek catch

- Zna ponovno vreči izjemo
  - Izjema se preusmeri h kličoči funkciji
- Primer
- **catch (ExceptionType e) {**

  … **// local action for exception**

  throw e; **// rethrow exception**

  } **// propagate exception to caller**

Več zaporednih blokov catch

- Če želimo prestreči več izjem hkrati lahko to storimo s pomočjo zaporednih blokov ***catch***.
- V veliko primerih se lahko zgodi, da bi radi nekatere operacije naredili ne glede na to, ali je prišlo do izjeme ali ne. Blok, s katerim to dosežemo, se imenuje ***finally***. Koda v tem bloku se bo izvršila neodvisno od proženja izjeme.

- Struktura:

```
try{
  A;
}
catch(ImeIzjeme1 i){
  B1;
}
catch(ImeIzjeme2 i){
  B2;
}
...
catch(ImeIzjemeN i){
  BN;
}
finally{
  C;
}
```


Primer:

```
import java.io.*;
public class VecIzjem{
  public static void main(String[] args) throws Exception{
    //poskušamo prebrati dano datoteko
    try{
      String dat = args[0];
      BufferedReader datoteka = new BufferedReader(new FileReader(dat));
      while(datoteka.ready()){
        System.out.println(datoteka.readLine());
      }
      datoteka.close();
    }
    //dva zaporedna catch stavka za dve različni izjemi
    catch (ArrayIndexOutOfBoundsException e){
      //izjema se sproži, če ne podamo argumenta
      System.out.println("Podaj ustrezni argument!");
    }
    catch (FileNotFoundException e) {
      //izjema se sproži, kadar podana datoteka ne obstaja
      System.out.println("Datoteka ne obstaja.");
    }
    finally{
      System.out.println("Blok se izvrsi vedno.");
    }
  }
}
```


- ***Izpis:***
- > java VecIzjem neobstojecaDat.txt
  Datoteka ne obstaja.
  Blok se izvrsi vedno.
  >
- > java VecIzjem
  Podaj ustrezni argument!
  Blok se izvrsi vedno.
  >
- > java VecIzjem datoteka.txt
  prva vrstica datoteke
  druga vrstica datoteke
  tretja vrstica datoteke
  Blok se izvrsi vedno.
  >

Stavek finally

- Sledi stavkom try in vsem blokom catch
- Formira zadnji blok
- Koda za “čiščenje” (cleanup)
  - Izvedejo jo vsi rokovalniki izjem
  - Poskuša restavrirati konsistentno stanje programa
- Vedno se izvede
  - Ne glede na to kateri blok catch se izvede
  - Tudi če se catch blok ne izvede
  - Izvede se preden gre kontrola kličoči funkciji
    - Če se izjema ne ulovi lokalno

Primer

- Kaj izpiše spodnji program?
- public static void main(String[] args) {

```
System.out.println("A");
try {
  if (args.length != 1) return;
  System.out.println("B");
} finally {
  System.out.println("C");
  }
  System.out.println("D");
}
```

- *V primeru, da podamo en argument*
  - A
  - B
  - C
  - D
- *V primeru brez(ali več) argumentov*
  - A
  - C

Načrtovanje in uporaba izjem

- Uporabi izjeme samo za redke dogodke
  - Ne za običajne primere → lovljenje konca zanke
  - Več procesiranja pri delu z izjemami
- Postavi stavke, ki skupaj naredijo postopek v en **try/catch blok**
  - **try/catch se med sabo dopolnjujeta**
- Uporabi obstoječe Java izjeme, če je mogoče

Povzetek

- Jezikovni gradniki
  - **try**
- Formira block **try**
- Objame vse stavke, ki lahko vržejo izjeme
  - **throw**
- Vrže izjemo
  - **catch**
- Ulovi izjemo določenega tipa
- Koda v bloku catch → rokovalnik izjeme
  - **finally**
    - Formira blok **finally**
    - Vedno se izvede, sledi blokoma **try & catch**

bostjan.vouk@tsc.si

Načrtovanje in uporaba izjem

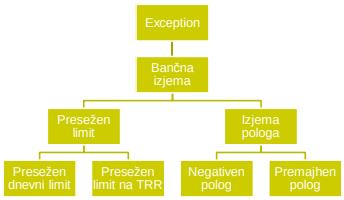
- Izognimo se enostavnemu lovljenju in ignoriranju izjem.
  - Slab stil, običajno se lovljenje izjem izplača
- Primer

```
try {
throw new ExceptionType1( );
throw new ExceptionType2( );
throw new ExceptionType3( );
}
catch (Exception e) {          // catches all exceptions
      …                        // ignores exception & returns
}
```

bostjan.vouk@tsc.si

Uporabniške izjeme

- definira programer
- zahtevajo dodatno delo
- poenostavijo programiranje



bostjan.vouk@tsc.si

Uporabniške izjeme

- Primer:
- public void Polog(float vsota) throws PremajhenPolog, NegativenPolog {
  if (vsota<0)
  throw new NegativenPolog(vsota);
  else if ((vsota<1000)&& !(vsota<0))
  throw new PremajhenPolog(vsota,String.valueOf(vsota));
  else
  stanje+=vsota;
  }

bostjan.vouk@tsc.si