Objektorienterad Programmering (TDDC77)

Föreläsning VI: eclipse, felsökning, felhantering

Ahmed Rezine

IDA, Linköpings Universitet

Hösttermin 2016

Outline

Felhantering

Eclipse

Felsökning

Command line argumenter

```
/* Cla.java
 * Programmet illustrerar hur man kan
 * passerar command line argument
 */
class Cla{
    //få ett kursnman som command line argument och returnerar om den ingår
    static public void main(String[] args){
     String[] it1Kurser = {"TDDC77", "TDDD39", "TDDC75"};

     if(args.length==1){
         boolean hittat= false;

         for(int pos=0; pos<it1Kurser.length && !hittat;pos++)
          System.out.println(it1Kurser[pos]);
          if(it1Kurser[pos].equals(args[0]))
              hittat = true;



         if(hittat){
          System.out.println(args[0] + " ingår i IT1");
         }
         else
          System.out.println(args[0] + " ingår inte i IT1");

     }
     else
         System.out.println("Anropa med: Cla kursnamn");
    }
}
```

Vad blir resultatet?

```
/* SkuggningLokalt.java
 * Programmet illustrerar hur lokala variabler
 * kan skugga klass variabler.
 */
class SkuggningLokalt{

    public static int a = 6;
    public static int b = 5;


    public static int addition (int c){
     int a = 2;
     return a + c;
    }

    public static void main(String[] args){
     a = addition(b);
     System.out.println(a);
    }
}
```

Nu då?

```
/* SkuggningParam.java
 * Programmet illustrerar hur parametrar
 * kan skugga klass variabler.
 */
class SkuggningParam{

    public static int a = 6;
    public static int b = 5;

    public static int addition (int a){
     return a + b;
    }

    public static void main(String[] args){
     a = addition(b);
     System.out.println(a);
    }

}
```

Outline

Felhantering

- Exception in thread “**main**” **java.lang.NumberFormatException**: 23.5
- En del metoder och operationer kan kasta ut fel
- I API:et står vilka fel som kastas för en viss metod
- Integer.parseInt(String) kastar ett NumberFormatException
- Alla fel kan fångas och hanteras med **try**/**catch**-konstruktionen

Läs in heltal

```
int number = 0;
print ("Mata in ett tal ");
number = Integer.parseInt(in.nextLine());
println("Du matade in talet: " + number);
```

- Fungerar för det mesta, men vad händer om man matar in “2.5” eller “e” ?

Läs in heltal med felhantering

```
boolean valueOK = true;
int number = 0;
do{
     valueOK = true;
     print ("Mata in ett tal ");
     Scanner in = new Scanner (System.in);
     String line = in.nextLine();
     try{
      number = Integer.parseInt(line);
     } catch (NumberFormatException e){
      System.out.println(line + " är inget tal");
      valueOK = false;
     }
} while (!value OK);
System.out.println("Du matade in talet: " + number);
```

Hantera felet

- java.lang.ArrayIndexOutOfBoundsException

```
int[] list = {45, 34, 67, 98};
while(true){
    print("Ange ett index: ");
    int index = Integer.parseInt(in.nextLine());
    print("Resultat: " + list[index]);
}
```

Scanner och Slingor: exempel

```
import java.util.Scanner;
import java.io.*;

public class HittaKonto{

    static public void main(String[] args) throws IOException{
     Scanner namnScan, filScan, radScan;
     String rad, namn;

     namnScan = new Scanner(System.in);
     System.out.print("Efternamn ? ");
     namn = namnScan.next();

     filScan = new Scanner(new File("it-konton-2014.txt"));

     while(filScan.hasNext()){
         rad = filScan.nextLine();
         radScan = new Scanner(rad);
         String radKod = radScan.next();
         while(radScan.hasNext()){
          String radNamn = radScan.next();
          if(radNamn.equalsIgnoreCase(namn))
              System.out.println("Matchar: " + rad);
         }
     }
    }
}
```

Hantera felet

- Kan man undvika att ett fel inträffar från första början så är det bättre än try/catch
- Vilket/vilka fel kan man undvika i arrayindexeringsexemplet?

Outline

Eclipse

- Eclipse är en Integrated Development Environement (IDE)
- Är helt gratis för alla operativsystem
- Finns i flera varianter för olika språk/tillämpningar
- Sköter organisationen av filer, editering av filer, kompilering, exekvering, felsökning mm

Hemma

- `http://eclipse.org`
- Klicka på “Download Eclipse”
- Välj “Eclipse IDE for Java Developers”

Starta

- Starta Eclipse (på universitetet med kommandot **eclipse&**)
- Eclipse vill att du ska välja workspace, välj en mapp som alltid är tillgänglig och som kan användas hela kursen. Här kommer alla dina projekt och filer att sparas
- Om du starta fe för första gången får du upp en välkomstskärm. Klicka på den böjda pilen till höger för att bli av med den
- Du bör nu ha Eclipse igång med sisådär fyra delfönster

Översikt

- Till vänster bör du se **Package Explorer**. Här ser du dina olika projekt, och dess filer (tom första gången)
- I mitten ligger **Kodfönstret**
- Till höger ligger **Outline** här syns en översikt över dina funktioner och globala variabler
- Längst ner finns en mängd tabbar, bland andra **Problems** för felmeddelanden om din kod och **Console** där in-/utmatning till/från programmet sker när du kör det

Skriv ut array

- Skriv en metod printArray(int[] list) som givet en array av heltal, list, skriver ut dess element separerade med kommatecken

Outline

Buggar överallt!!

- Java-kompilatorn borde kunna hitta syntaktiska fel, som har att göra med syntaxen: en saknad “;” eller “}”
- Java kompilatorn kan ibland hitta enkla semantiska fel: som oinitialiserade variabler, död kod
- De flesta logiska och semantiska fel, funktionella fel mm, som att använda fel variabel eller “||” istället för “&&”, de får ni leta efter själva !

Spår-utskrifter

- Man skriver helt enkelt ut väl valda värden på variabler under programkörning
- En av de vanligaste och enklaste sätten att felsöka ett program

Spår-utskrifter

```
static int tal1 = 6, tal2 = 5;

static int addition(int a, int b){
    System.out.println("addition: Startar funktionen");
    int c = a - b;
    return b;
}

public static void main(String[] args){
    int svar = addition(tal1, tal2);
    System.out.println("main: addition(" +
                       tal1 + ", " + tal2 + ") =" + svar);
}
```

```
> java FelaktigFunktion
addition: Startar funktionen
main: addition(6, 5) = 5
...
>
```

Debug-läget

- Ett speciellt Eclipse-läge som är anpassat för avlusning
- Arrangerar bland annat om underfönstren

Brytpunkter (eng. Breakpoints)

- Om man kör i debug-läge stannar programmet vid brytpunkter
- Man kan undersöka variablernas värden, och fortsätta "ett steg i taget"
- Mer invecklat än spårutskrifter, men också mer kraftfullt

Koda, undersök med spårutskrifter eller debug-läget

- Skriv en metod max(int[] s) som returnerar det högsta talet i arrayen s
- Skriv en metod som avgör om en array är sorterad: minst till störst
- Skriv en metod som avgör om en array är likadan som en annan