TDDC77 Objektorienterad Programmering

Föreläsning 4

Sahand Sadjadee
IDA, Linköpings Universitet

Outline

- Metoder (8.1, 8.2, 8.3, 8.5, 8.6, 8.7, 8.8, 8.9)
- Arrayer (11.1, 11.2, 11.3, 11.4, 11.5, 11.7)
- Flerdimensionella arrayer (12)
- Räckvidd och Livslängd

Arrayer

Vända om inlästa värdena

```
/* Reverse.java
 * Programmet demonstrera hur man kan läsa
 * in verden och skriva ut de i omvänd ordning
 */
import java.util.Scanner;
class Reverse{
    /* Be användaren om två tal och skriver ut de
     * i omvänd ordning
     */
    public static void main(String[] args){
     Scanner in = new Scanner (System.in);
     int x0, x1, x2, x3;

     System.out.println ("Mata in ett heltal: ");
     x0 = in.nextInt();
     System.out.println ("Mata in ett heltal: ");
     x1 = in.nextInt();
     System.out.println ("Mata in ett heltal: ");
     x2 = in.nextInt();
     System.out.println ("Mata in ett heltal: ");
     x3 = in.nextInt();
     System.out.print("Du matade in:");
     System.out.print(" "+ x3);
     System.out.print(" "+ x2);
     System.out.print(" "+ x1);
     System.out.print(" "+ x0);
     System.out.println();
    }
}
```

Vända om inlästa värdena

- Vi kan lätt göra samma sak för fyra variabler ...
- Hur gör vi för 400 inlästa värden ?
- Oftast behöver man flera likadana variabler
- Kallas “Array” på engleska, ibland “mängd” på svenska

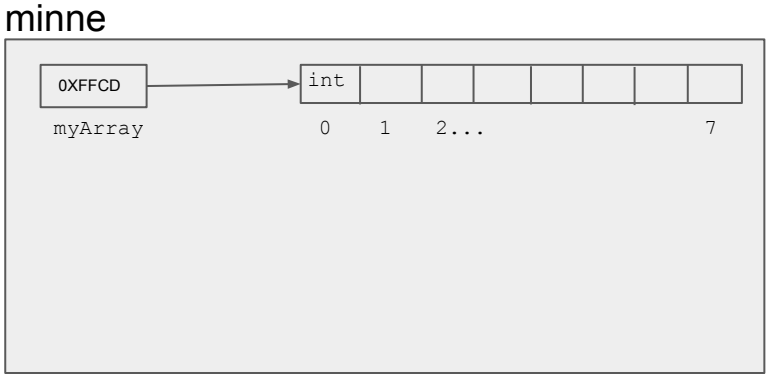
```
int[] numbers = new int[4];
// ... eller ...
int[] numbers;
numbers = new int[4];
// ... eller ...
int[] numbers = {3,5,8,13};

System.out.println("Första elementet är: " + numbers[0]);
System.out.println("Sista elementet är: " + numbers[numbers.Length - 1]);

float[] ratios = new float[12];
String[]  names = new String[MAX];
boolean[] flags = new boolean[numberOfFlags];
String[] names = {"TDDC77", "TDDD39", "TDDC75"};
```

Arrays

- En array består av ett antal variabler av samma typ.
- Arrayens längd kan inte ändras efter den har skapats.
- Varje variabel i array:en har ett nummer som kallas för **index**. Indexering börjar från 0.
- Alla variabler i arrayen initierar med värdet 0 eller motsvarande 0.
- Nyckelordet **`new`** används för att skapa array:er.
- När array:en är skapad, ersätts hela **`new uttrycket`** med referensen till skapade array:en i primärminnet.
- Referensen ska lagras i en variabel av typen datatype[] för att användas senare i koden.




```
int[] myArray = new int[8];
myArray[0] = 13;
System.out.println(myArray[0]);
```

Arrays - syntax

Att deklarera

```
DataType[] variableName;
```

Att skapa

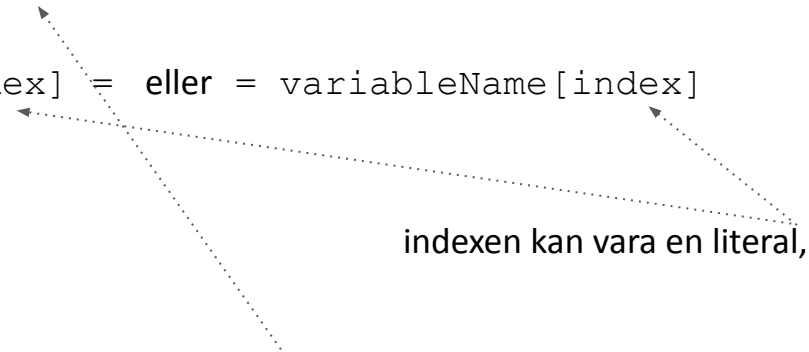
```
new DataType[length];
```

Att använda

```
variableName[index] =  eller = variableName[index]
```

Exempel:

```
int[] tddc77Grades;
tddc77grades = new int[37];
tddc77Grades[0] = 100;
```



indexen kan vara en literal, en variabel eller ett uttryck.

Längden kan vara en literal, en variabel eller ett uttryck.

Att vända om 400 inlästa värden

```
int nrOfInts = 400;
int[] värden = new int[nrOfInts];
int pos = 0;

while(pos <nrOfInts){
    System.out.println("Mata in ett heltal: ");
    värden[pos] = in.nextInt();
    pos = pos + 1;
}

pos = 399;

while(pos >= 0){
    System.out.println(värden[pos] + " ");
    pos = pos - 1;
}
```

Att traversera en array

- En for-slinga kan användas för att gå igenom, traverse, arrayen.
- “ to traverse” betyder att gå igenom variablerna i arrayen för att läsa eller uppdatera dem en i taget i ordning.
- Exempel:

```
int[] grades = new int[10];
for (int index = 0 ; index < grades.length ; ++index){
    grades[index] = input.nextInt();
    System.out.println (grades[index]);
}
```

Advanced for-loop

- En form av for-slinga som bara kan användas för att **läsa** variablerna i arrayen en i taget i ordning.
- Exempel:

```
int[] grades = new int[10];
for (int grade : grades){
    System.out.println (grade);
}
```

Att vända om 400 inlästa värden med en for-loop

```
int nrOfInts = 400;
int[] vaerden = new int[nrOfInts];


for(int pos=0; pos<nrOfInts; pos++){
    System.out.println("Mata in ett heltal: ");
    vaerden[pos] = in.nextInt();
}

for(int pos=nrOfInts-1; pos>=0; pos--){
    System.out.println(vaerden[pos] + " ");
}
```

Metoder

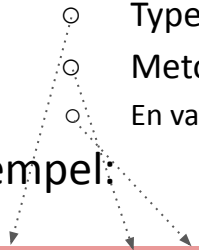
Metoder

- En del kod med eget namn.
- Kan anropas flera gånger från olika ställen i programmet.
- En metod kan returnera noll eller ett värde som resultat.
- En metod kan ta emot noll, ett eller flera värden.
- En metod består av en header och en kropp(body).
- kroppen skrivs mellan två måsvingar.
- exempel:

```
int add(int op1, int op2){ //header
    //body
    int result = op1 + op2;
    return result;
}
```

Metoder - header

- Headern består av:
  - Typen på data som returneras av metoden
  - Metodens namn
  - En variabellista, parameterlista, som används för att ta emot data. Variabellistan ligger mellan ett par parenteser.
- exempel:



```
int add(int op1, int op2){ //header
    //body
    int result = op1 + op2;
    return result;
}
```

Metoder - body

- kroppen består av:
  - Noll, en eller flera instruktioner som exekveras en i taget i ordning när metoden anropas.
  - En `return`-instruktion som returnerar ett värde och avslutar metoden. Om returtypen är `void` då ingen `return`-instruktion behövs.
  - Return kan returnera en literal, en variabel eller ett uttryck
- exempel:

```
int add(int op1, int op2){ //header
    //body
    int result = op1 + op2;
    return result;
}
```

Tillfälliga Regler

- Vi lägger till nyckelordet `static` till början av headern.
- Vi deklarerar/skapar bara våra metoder i klassen där `main` metoden finns.
- Exempel:

```
public class Main{
    public static void main(String[] args){
        add(3, 7);
    }
    static int add(int op1, int op2){
        int result = op1 + op2;
        return result;
    }
}
```

Metoder - anrop

- En metod kan anropas från en annan metod. Till exempel, main metoden.
- Ett anrop består av följande:
  - Metodens namn
  - En lista på värdena som ska skickas till metoden(arguments).
    - Argument skrivs mellan paranteser.
    - Varje värde i listan initierar motsvarande variabeln i parameterlistan.

```
public class Main{
     public static void main(String[] args){
          add(3, 7);
    }
     static int add(int op1, int op2){
            int result = op1 + op2;
             return result;
     }
}
```

Metoder - anrop

- Ett anrop är ett uttryck och ersätts av värdet som returneras av anropade metoden.

```
public class Main{
    public static void main(String[] args){
        int returnedValue = add(3, 7);
    }
    static int add(int op1, int op2){
           int result = op1 + op2;
           return result;
    }
}
```

Metoder - anrop

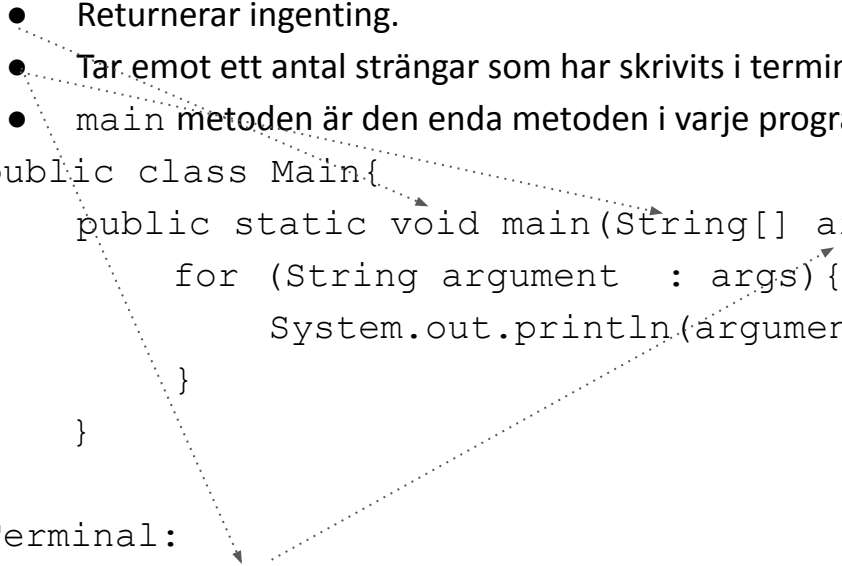
- När ett anrop sker, pausas exekveringen i anropande metoden tills anropade metoden returneras.

```
public class Main{
    public static void main(String[] args){
        int returnedValue = add(3, 7);
        System.out.println(returnedValue);
    }
    static int add(int op1, int op2){
           int result = op1 + op2;
           return result;
    }
}
```

`main` metoden

- Returnerar ingenting.
- Tar emot ett antal strängar som har skrivits i terminalen.
- `main` metoden är den enda metoden i varje program som anropas direkt av Java Virtuell Maskine.

```
public class Main{
    public static void main(String[] args){
        for (String argument  : args){
            System.out.println(argument);
        }
    }
}
Terminal:
java Main argument01 argument02 argument03
```



Metoder - arrayer

- En parameterlista kan också innehålla en variabel, eller flera, som pekar på en array.
- Ibland är det vettigare att använda en array istället för en lång parameterlista som består av flera variabler av samma typ.

```
public class Main{
      public static void main(String[] args){
            int[] ops = new int[2];
            ops[0] = 3;
            ops[1] = 7;
            int returnedValue = add(ops);
            System.out.println(returnedValue);
    }
      static int add(int[] ops){
             int result = ops[0] + ops[1];
               return result;
      }
}
```

```java
import java.util.Scanner;
public class NerBryten{

    // Få en int array som parameter och returnera summan av dess element
    static public long computeSum(int[] nums){
     long result = 0;
     for(int p=0; p<nums.length; p++)
         result += nums[p];
     return result;
    }

    //Få en int array som parameter och returnerar medelvärdet av dess element
    static public double computeAverage(int[] nums){
     return (double)computeSum(nums)/nums.length;
    }

    //läser in ett heltal som är större eller lik med parametern "min"
    static public int readIntLargerOrEqualTo(int minimal){
     Scanner scan = new Scanner(System.in);
     int result;
     do{
         System.out.print("Enter an int that is larger or equal to " + minimal + ": ");
         result=scan.nextInt();
     }while(result<minimal);
     return result;
    }

    //läser in antalet element och deras värden, skriver upp deras medelvärde
    static public void main(String[] args){
     int[] numbers = new int[readIntLargerOrEqualTo(1)];
     for(int k=0; k<numbers.length; k++)
         numbers[k] = readIntLargerOrEqualTo(0);
     System.out.println("The average value is: " + computeAverage(numbers));
    }
}
```

Räckvidd och Livslängd

Räckvidden(scope) för en variabel

- Räckvidden för en variabel är området i koden där variabeln kan användas.
- Det är inte möjligt att ha två variabler med samma namn i samma räckvidd.
- En variabel är inte användbar i kodrader som ligger före deklarationen.
- En metod har egen räckvidd.
    - En variabel som deklareras i en metod kan användas bara i den metoden.
    - En variabel som deklareras direkt i en metod, utanför block eller kontrollstrukturer, kallas för en lokal variabel.
    - En parameter i parameterlistan kan användas i hela metoden.
- Ett block har egen räckvidd
    - En variabel som har deklarerats i ett block kan bara användas i det blocket.
- Initieringsdelen i en for-slinga har egen räckvidd.
    - En sådan variabel kan användas i villkoret, uppdatering och for-slingans kropp.

livslängden(lifetime) för en variabel

En variabel avallokeras när exekveringen lämnar området där variabeln är användbar i.

Garbage Collection

Lokal-variabler lagras i Stack Memory.

Arrayer lagras i Heap Memory.

En array tas bort från primärminnet automatiskt om det finns ingen referens till den.

Skuggning

- En variabel kan skugga en anna variabel med samma namn.
- Det händer mest när en variabel i ett block har samma namn som en annan variabel utanför blocket(en lokal variabel).
- Exempel:

```
void add(int[] values){
        int sum = 0;
        for(int value : values){
            int sum;
            sum += value;
        }
        return sum;
}
```

Tack för att du lyssnade!

LINKÖPINGS UNIVERSITET