# Föreläsning 12

- Metod med returvärde
- Konstruktor
- Klassvariabel, klassmetod och klasskonstant

JF: 5.4-5.5

# Metodanrop

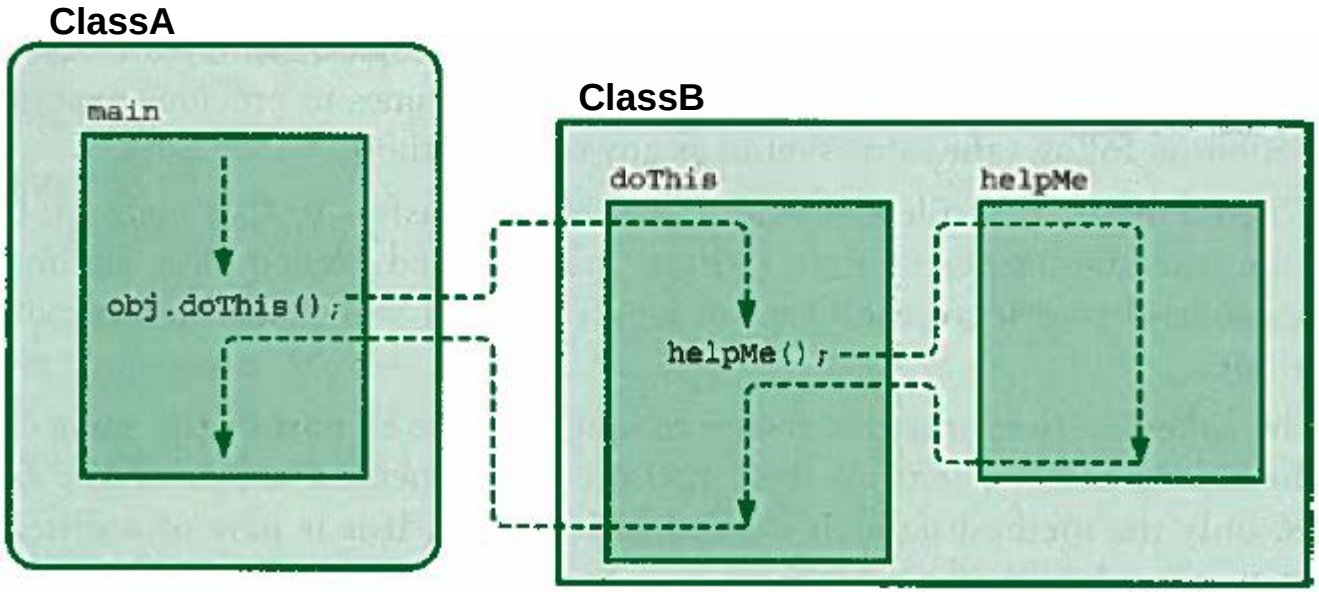



**FIGURE 5.8** The flow of control following method invocations

```
public class ClassA {
   public static void main(String[] args) {
      ClassB obj = new ClassB();
      obj.doThis();
      System.out.println("END");
   }
}
```

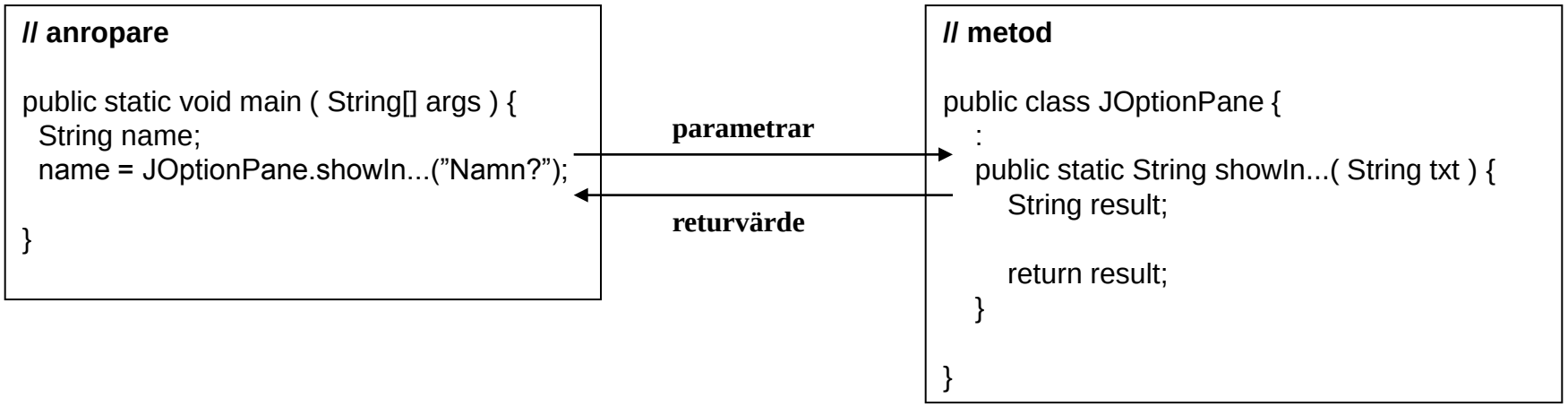
```
Hmm…, YES or NO?
YES
END
```

```
public class ClassB {
   public void doThis() {
      String answer;
      System.out.println("Hmm…, YES or NO?");
      answer = helpMe();  // = this.helpMe();
      System.out.println( answer );
   }

   private String helpMe() {
      return "YES";
   }
}
```

# Metod med returvärde

Med ***parametrar*** överför man ***data (värde)*** från anropare till metod.
Med ***returvärde*** överför man ***data (värde)*** från metod till anropare.




**name = JOptionPane.showInputDialog( ”Namn?” );**

***Vid anropet*** överförs en sträng med innehållet ***”Namn?”*** till ***parametern txt***.

När metoden exekverar raden

**return result;**

Överförs värdet i ***result*** till ***variabeln name***.

# Metod med returvärde

Klassen Commodity skulle kunna innehålla en metod som beräknar värdet av en vara på lagret, dvs antal · pris.

```
public class Commodity {
   private String name;
   private String category;
   private int quantity;
   private double price;
   :
   public  double value() {
      double total = this.quantity * this.price;
      return total;
   }
}
```

I metodens deklaration anger **double** att ett värde av typen double ska överföras från metod till anropare, att metoden svarar med en double.
I metodens kropp anger **return** det värde som ska överföras till anropande kod. I detta fall är det värdet i variabeln **total**.

```
Commodity com = new Commodity();
double totalValue;
:
totalValue = com.value();
```

Commodity.java

CommodityEx1.java

# Metod med returvärde

Klassen Commodity skulle kunna innehålla en metod som returnerar en sträng med information om varan.

```
public class Commodity {
   public String name;
   public String category;
   public int quantity;
   public double price;
   :
   public  String toString() {
      String res = ”Varunamn = ” + this.name + ”\nKategori = ” + this.category +
                   ”\nAntal = ” + this.quantity + ”\nPris = ” + this.price;
      return res;
   }
}
-----------------------------------------------------
Commodity com = new Commodity();
:
System.out.println( com.toString() );  // info om varan skrivs i Console-fönstret
```

CommodityEx2.java

# Klass med get-metoder

Ofta lägger man till get-metoder i en klass. En ***get-metod*** ska ***returnera värdet*** på en ***instansvariabel*** i klassen.

```
public class Commodity {
   private String name;
   private String category;
   private int quantity;
   private double price;

   : // set-metoder, value-metod och toString-metod här

   public String getName() {
      return this.name;
   }

   public String getCategory() {
      return this.category;
   }

   public int getQuantity() {
      return this.quantity;
   }

   public double getPrice() {
      return this.price;
   }
}
```

| Commodity |
| --- |
| - name : String<br>- category : String<br>- quantity : int<br>- price : double |
| + setName( String )<br>+ setCategory( String )<br>+ setQuantity( int )<br>+ setPrice( double )<br>+ getName() : String<br>+ getCategory() : String<br>+ getQuantity() : int<br>+ getPrice() : double<br>+ value() : double<br>+ toString() : String |

Som du ser deklareras get-metoderna så här:
**public variabeltyp getVariabelnamn() {…}**

CommodityEx3.java

# Konstruktor

En konstruktor exekveras ***alltid*** när ett objekt skapas och är till för att initiera objektet. Följande gäller för en konstruktor:

1. Konstruktorn har samma namn som klassen.
2. Konstruktorn fungerar som en metod
3. Konstruktorn deklareras utan returtyp. Returvärdet är alltid referens till objektet som skapas.
4. En konstruktor exekveras endast när objektet skapas.

```
public class Book {
   private String title;
   private String author;
   private String isbn;

   public Book( String inTitle, String inAuthor, String inIsbn ) {
      this.title = inTitle;
      this.author = inAuthor;
      this.isbn = inIsbn;
   }

   :
}
```

Book.java

BookEx.java

# Konstruktor, default-konstruktor

De klasser du hittills skrivit på kursen har inte haft någon konstruktor. När så är fallet lägger kompilatorn till en ***default-konstruktor*** vid kompileringen.

Klassen CD innehåller ingen konstruktor:

```
public class CD {
   private String title;
   private String artist;

   // här är metoderna setTitle, setArtist, getTitle, getArtist
}
```

Konstruktorn som läggs till vid kompileringen ser ut så här:

```
   public CD() {

   }
```

Som du ser har default-konstruktorn en tom parameterlista och tom kropp (inga instruktioner i kroppen).
Vid konstruktion av ett CD-objekt exekveras default-konstruktorn.

```
CD cd = new CD();
```

# Konstruktorer i klassen Commodity

Klassen Commodity har två konstruktorer:

```
public class Commodity {
   private String name;     // varans namn
   private String category; // varans avdelning
   private int quantity;    // antal av varan i butiken
   private double price;    // varans pris

   public Commodity() {
      this.namn = “”;
      this.kategori = “”;
   }

   public Commodity( String inName, String inCategory, int inQuantity, double inPrice) {
      this.name = inName;
      this.category = inCategory;
      this.quantity = inQuantity;
      this.price = inPrice;
   }

   // set- och get-metoder, value, toString
}
```

| Commodity |
| --- |
| + Commodity()<br>+ Commodity(String,String,int,double) |
| + setName( String )<br>+ setCategory( String )<br>: |

Commodity-objekt kan skapas på två sätt:

1. `Commodity com1 = new Commodity();`
2. `Commodity com2 = new Commodity(“Läkerol”,“Godis”,320,7.90 );`

| com1 : Commodity |
| --- |
| name = ”” |
| category = ”” |
| quantity = 0 |
| price = 0.0 |

| com2 : Commodity |
| --- |
| name = ”Läkerol” |
| category = ”Godis” |
| quantity = 320 |
| price = 7.90 |

CommodityEx4.java

# Klassvariabel

En **klassvariabel** deklareras direkt i klassen. Modifieraren ***static*** gör variabeln till en klassvariabel:

```
public class Account {
   public static double interestRate;
   :
}
```

Klassvariabeln är tillgänglig (synlig) i hela klassen. Den kan användas av alla metoder i klassen.

Klassvariabeln tillhör klassen. Alla objekt av typen Account delar på variabeln *interestRate*. Det innebär att alla konton har samma räntesats.

Klassvariabeln finns även om det inte skapats några objekt av klassen. Om klassvariabeln är ***public***-deklarerad så kommer kod i andra objekt åt variabeln genom att ange klassens namn + variabelns namn:

```
double rate = Account.interestRate;  // rate tilldelas värdet i interestRate
Account.interestRate = 0.032;  // interestRate tilldelas värdet 0.032
```

# Klassmetoder

En metod som deklareras med modifieraren ***static*** är en **klassmetod**.

```
public class Math {
   public static double random() {
      // instruktioner för att generera slumptal och returnera det
   }
```

Klassmetoden är allmänt tillgänglig om den är public-deklarerad. Med detta menas att den kan anropas från kod i andra klasser utan att det finns något objekt av klassen den tillhör. Anropet sker med hjälp av klassens namn, t.ex.

```
double rnd = Math.random();
```

Det som är typiskt för en klassmetod är att den är fristående från objekt av klassen den tillhör. Den får som regel all information som behövs via parametrar (kan dock erhålla information från t.ex. klassvariabler). T.ex. så behöver metoden ***Integer.parseInt*** en sträng som argument. Strängen anges vid anropet av metoden, t.ex.

```
int nbr = Integer.parseInt( "8376" );
```

En klassmetod kan inte använda instansvariabler eller instansmetoder i klassen. Instansvariabler och instansmetoder kräver ju att man skapat ett objekt av klassen.
I en klassmetod kan man använda klassvariabler och klassmetoder som deklareras i samma klass eller i en annan klass.

# Klasskonstanter

Det går bra att deklarera konstanter av de enkla variabeltyperna. En konstant är en variabel vars värde inte kan förändras. En konstant deklareras med modifieraren **final**. Konstanten måste ges ett värde innan den används.

```
public class ClassA {
   private final int  TUSEN = 1000;                    // Konstant i klassen
   public final static double PI = 3.141592653589793; // Klasskonstant
   :

   public void method() {
      final char CHR = 'U';                            // Konstant i metoden
      :
   }
}
```

Det är vanligt att konstanterna deklareras med ***static*** i java. De kallas då för **klasskonstanter**. I ovantående exempel kommer man åt konstanten PI med hjälp av klassens namn, t.ex.:

```
area = ClassA.PI * radius * radius;
```

Du kommer använda denna typ av konstanter bl.a. i samband med grafiskt användargränssnitt.