Föreläsning 11

- Klass
- Synlighet
- Modell med instansvariabler och metoder
- Klassdiagram
- Metod med parameter

Java Foundations: 5.1-5.3

Modell

En modell är en förenklad representation av verkligheten. Endast de egenskaper som är viktiga ingår i modellen.

En **person** kan beskrivas med varierande egenskaper, dvs. egenskapernas vikt beror på sammanhanget, t.ex.:

- Skattemyndigheten finner bl.a. följande egenskaper viktiga:
  Personnummer, Taxerad inkomst, Skatt
- Arbetsgivaren finner bl.a. följande egenskaper viktiga:
  Kompetens, Lönekrav, Lojalitet
- Adressboken finner bl.a. följande egenskaper viktiga:
  Namn, Adress, Telefonnummer

På samma sätt beskrivs en **bil** på skilda sätt, t.ex.:

- Bilhandlaren är intresserad av modell och pris
- P-vakten är intresserad av modell och registreringsnummer
- Navigationssystemet är intresserat av bilens geografiska läge

Klass

I datorprogram utgörs en modell av ett ***objekt***.

I Java bygger man objekt med hjälp av en ***klass***. Klassen är en beskrivning av objektet, dvs. vilka attribut som ska finnas i objektet och vilka beteenden objektet ska ha.

Ett program består normalt sett av många objekt som på olika sätt påverkar varandra.
Och en del av objekten som används kan vara av samma typ. Om många personer behandlas i ett program så kommer programmet att använda många objekt av typen Person.

Klassens struktur

En klass i java kan bl.a. innehålla:

- **Instansvariabel** Variabel som gäller i hela klassen (kallas även *attribut*)
- **Konstruktor** Metod som exekveras när objektet skapas
- **Instansmetod** Kallas kortare för ***metod***.
- **Inre klass** En klass som deklareras i en annan klass

```
public class ClassEx {
   // Instansvariabler (attribut)
   private int length;
   private double average;

   // Konstruktor
   public ClassEx( int length) {
      this.length = length;
   }

   // Metod (instansmetod)
   public int getLength() {
      return this.length;
   }

   // Inre klass
   private class InnerClass {
      // instansvariabler / konstruktorer / metoder
   }
}
```

En ***instansvariabel*** ska som regel vara ***private***. Det innebär att det inte går att komma åt variabeln från kod utanför klassen.
I praktiken innebär det att kod utanför klassen inte kan avläsa eller ändra variabelns värde.

En ***metod*** är oftast ***public***. Det innebär att det går bra att anropa metoden från t.ex. andra objekt.

Synlighet

I java deklarerar man synlighet för bl.a. instansvariabler och metoder. Alternativen är:

Deklaration	**Åtkomst av instansvariabel / metod**
public	Från all kod
private	Från kod inom klassen
protected	Från kod inom klassen, kod i samma paket och kod i subklass
(ej angiven)	Från kod i samma paket

```
public class ClassEx {
   // Instansvariabler (attribut)
   private int length;
   private double average;



   // Metod (instansmetod)
   public int getLength() {
      return this.length;
   }
   :
}
```

En ***instansvariabel*** ska som regel deklareras som ***private***. Det innebär att det inte går att komma åt variabeln från kod utanför klassen. I praktiken innebär det att kod utanför klassen inte kan avläsa eller ändra variabelns värde.

En ***metod*** deklareras oftast som ***public***. Det innebär att det går bra att anropa metoden från kod utanför klassen.

Kommunicera med objekt

De metoder i objektet som är public-deklarerade kan man interagera med. Metoderna utgör "interfacet" eller "gränssnittet" vid kommunikation med objektet.

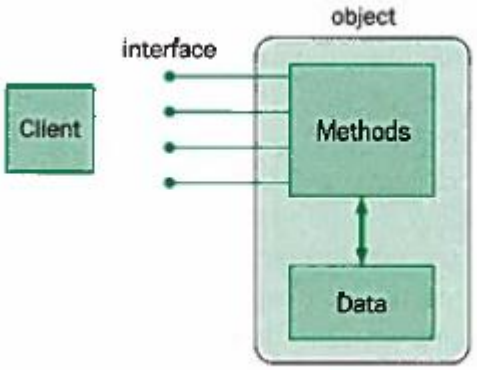


FIGURE 5.6 A client interacting with another object

Ett *väl inkapslat* objekt. All kommunikation sker via metoder.

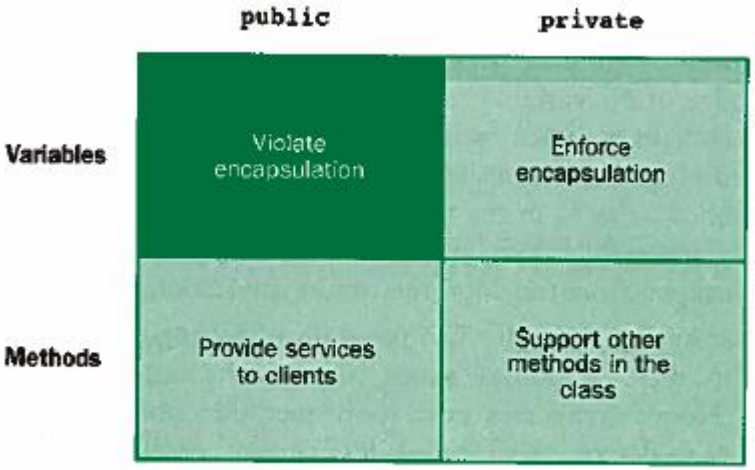


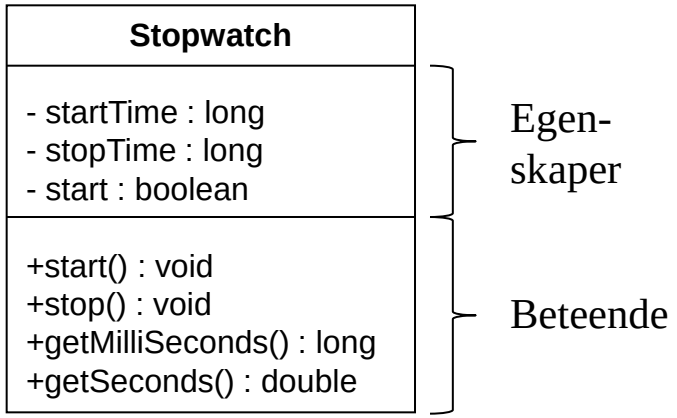
FIGURE 5.7 The effects of public and private visibility

Stopwatch – Modell av ett stoppur

Implementering

```
public class Stopwatch {
   private long startTime;
   private long stopTime;
   private boolean start = false;

   public void start() {
      this.startTime =System.currentTimeMillis();
      this.stopTime = this.startTid;
      start = true;
   }

   public void stop() {
      this.stopTime = System.currentTimeMillis();
   }

   public long getMilliSeconds() {
      if(start==true)
         return (this.stopTime-this.startTime);
      else
         return 0;
   }

   public double getSeconds() {
      return getMilliSeconds()/1000.0;
   }
}
```

Klassdiagram



Klassdiagrammet beskriver vilka *egenskaper* och *metoder* objekt av klassen Stopwatch har.
Av klassdiagrammet framgår också synligheten.

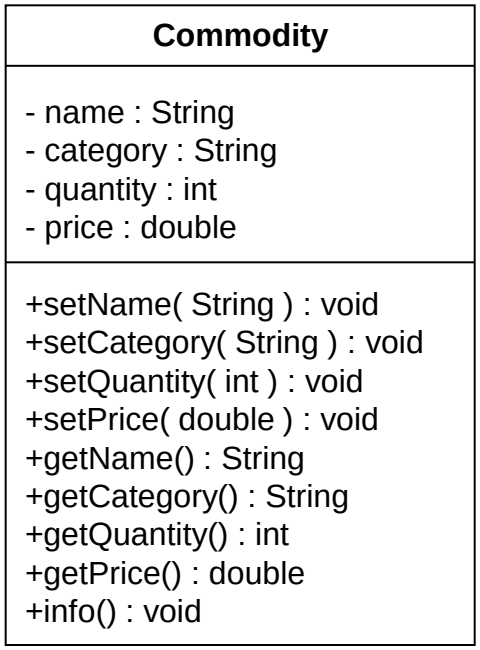
Stopwatch.java

Commodity – Modell av en vara i en affär

Implementering

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

   public void setName(String name) {
      this.name = name;
   }

   public void setCategory(String category) {
      this.category = category;
   }

   public void setQuantity(int quantity) {
      this.quantity = quantity;
   }

   public void setPrice(double price) {
      this.price = price;
   }

   // fler metoder här, se Commodity.java

   public void info() {
      System.out.println(this.name + ", " + this.category + "\n" +
                         "Antal i lager: " + this.quantity + "\n" +
                         "Pris: " + this.price + " kr");
   }
}
```

Klassdiagram



Commodity.java

Commodity – instansvariabel

I en butik med varor är följande egenskaper för en vara viktiga:

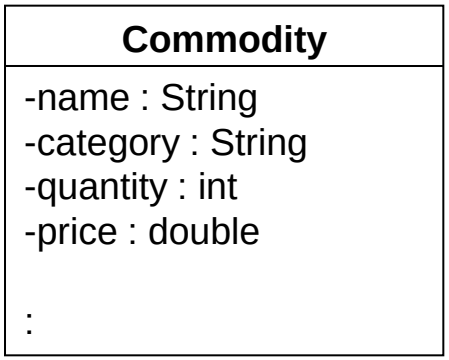
- namn t.ex. ”Mors köttbullar”
- kategori t.ex. ”Kött”
- antal t.ex. 400
- pris t.ex. 16.95

Om man ska lagra information om en vara i ett program bör *namn* och *kategori* vara av typen *String*, *antal* av typen *int* och *pris* av typen *double*.

I klassen Commodity är varje viktig egenskap en ***instansvariabel***.

I programmering kan man beskriva en vara med:

ett ***klassdiagram*** (UML)



en ***klass*** (java)

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

    :
}
```

Commodity - instansvariabel

I ett javaprogram kan man skapa ett objekt, en instans, av typen Commodity:

```
Commodity com1 = new Commodity();
```

Nu skapas utrymme i datorns internminne som rymmer 4 variabler: två referenser till String, en int och en double.

Variablerna ”nollställs” när objektet skapas.

com1 : Commodity	
name	null
category	null
quantity	0
price	0.0

I ett program kan man skapa flera objekt, instanser, av samma typ.
Varje objekt får sitt eget minnesutrymme:

```
Commodity com1 = new Commodity();
Commodity com2 = new Commodity();
```

com1 : Commodity	
name	null
category	null
quantity	0
price	0.0

com2 : Commodity	
name	null
category	null
quantity	0
price	0.0

Commodity - metod

Man använder ett Commodity-objekt i ett program med hjälp av de metoder som finns i klassen Commodity. Metodernas namn ska beskriver deras funktion.

Set-metod anropas för att ändra värdet i en instansvariabel. I klassen Commodity finns följande set-metoder: *setName*, *setCategory*, *setQuantity*, *setPrice*

Get-metod anropas för att få reda på värdet i en instansvariabel. I klassen Commodity finns följande get-metoder: *getName*, *getCategory*, *getQuantity*, *getPrice*

Metoden ***info*** skriver ut ett meddelande om ett Commodity-objekt i Console-fönstret. Meddelande speglar innehållet i instansvariablerna.

```
Pepparkakor, Bageri
Antal i lager: 0
Pris: 22.5 kr
```

Commodity
- name : String - category : String - quantity : int - price : double
+setName(String) : void +setCategory(String) : void +setQuantity(int) : void +setPrice(double) : void +getName() : String +getCategory() : String +getQuantity() : int +getPrice() : double +info() : void

com : Commodity	
name	”Pepparkakor”
category	”Bageri”
quantity	0
price	22.50

Metod

```
public double interest ( double sum, double interestRate ) {
    double calculatedInterest;
    calculatedInterest = sum * interestRate;
    return calculatedInterest;
}
```

// Deklaration

// Kropp

Metodens deklaration innehåller:

- **Modifierare - public** (gäller även variabler)
  - **public** - metoden tillgänglig från andra klasser
  - **private** - metoden endast tillgänglig inom klassen
  - **protected** - metoden tillgänglig inom klassen, subklasser och samma paket
  - **(ej angiven)** - metoden tillgänglig i samma paket
- **Returvärde - double**
  void innebär inget returvärde
  Tänkbara returtyper: enkla variabler och referensvariabler.
- **Namn - interest**
  Ska alltid börja med liten bokstav.
- **Parameterlista - (double sum, double interestRate)**
  Om en metod saknar parametrar så är parameterlistan tom: ().

Metodens kropp innehåller koden som ska exekveras då metoden anropas

```
{
   double calculatedInterest;
   calculatedInterest = sum * interestRate;
   return calculatedInterest;
}
```

Metod med parameterlista

En metod med parameterlista har en lista med en eller flera parametrar i en lista efter metodnamnet. I metoden *add* är parameterlistan **int nbr1, int nbr2**

```
public class Parameter {
   public void add(int nbr1, int nbr2) {
      int sum = nbr1 + nbr2;
      System.out.println( nbr1 + ” + ” + nbr2 + ”=” + sum );
   }
}
```

Metodens beteende beror på de argument användaren ger vid anropet till metoden.

```
Parameter param = new Parameter();
int a = 23, b = 40;
param.add(10, 20);
param.add(15, 7);
param.add(a, b);
```

```
10+20=30
15+7=22
23+40=63
```

Först förs värdet i argumentet över till parametern (kopieras)

```
param.add(10, 20);   =>   nbr1=10, nbr2=20          // tilldelning vid anropet
```

Sedan exekveras instruktionerna i metoden

```
int sum = 10 + 20;
System.out.println( 10 + ”+” + 20 + ”=” + 30 );
```

Parameter.java	ParameterEx.java

Variabler i metod

I en metod kan du använda följande variabler:

- ***Lokalt deklarerade variabler***. Det är variabler som deklareras inuti metodkroppen. Variabeln kan användas efter deklarationen. Variabeln måste dock tilldelas ett värde innan den används i ett uttryck.

```
public void showInfo() {
    double calculatedInterest; // calculatedInterest är en lokal variabel i metoden showInfo.
    :
}
```

- ***Parametrar i parameterlistan***. Parametrarna används som vanliga variabler inuti metoden. En parameter har alltid ett värde och kan användas direkt i metoden.

```
public void setQuantity( int inQuantity) {
   this.quantity = inQuantity; // inQuantity används som en variabel i metoden setQuantity.
}
```

- ***Instansvariabler***. Instansvariabler kan användas i metoden. En instansvariabel kommer man åt genom att skriva **this.variabelnamn** eller enbart **variabelnamn.**

```
public void setQuantity( int inQuantity) {
    this.quantity = inQuantity; // instansvariabeln quantity går bra att använda i metoden setQuantity.
}
```

Lokala variablers synlighet

En lokal variabel är endast användbar (synlig) i det **block** i vilket den är deklarerad. Och endast efter att den deklarerats. Ett block startar med { och slutar med }.

Om du deklarerar variabeln i början av metoden så kan den användas i resten av metoden (**result** nedan). Hela metoden-kroppen är ett block.

Om du deklarerar en variabel i ett inre block kommer den endast vara användbar i det inre blocket (**sum** nedan).

```
public void method( int number ) {
   double result;
   // div kod
   if(number>0) {  // block inuti block
      int sum = 0;
      // div kod
   }
   // div kod
}
```

Här kan variabeln **sum** användas

Här kan variabeln **result** användas

set - metod

Man använder *set-metod* för att ändra värdet i en instansvariabel.

```
Commodity com = new Commodity();
// ”Pepparkakor” förs över till intansvariabeln name
com.setName(”Pepparkakor”);
```

com : Commodity	
name	”Pepparkakor”
category	null
quantity	0
price	0

```
// 22.50 förs över till intansvariabeln price
com.setPrice(22.50);
```

com : Commodity	
name	”Pepparkakor”
category	null
quantity	0
price	22.50

```
// ”Bageri” förs över till intansvariabeln category
com.setCategory(”Bageri”);
```

com : Commodity	
name	”Pepparkakor”
category	”Bageri”
quantity	0
price	22.50

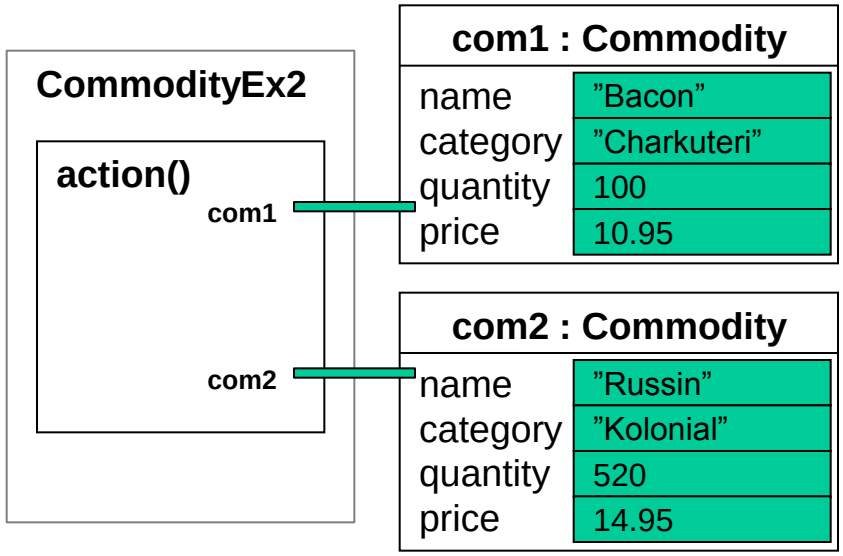
```
// Utskrift om varan i Console-fönstret
com.info();
```

```
Pepparkakor, Bageri
Antal i lager: 0
Pris: 22.5 kr
```

CommodityEx1.java

Flera Commodity-objekt i samma program

```
package f13;
/**
 * Programmet visar att det går utmärkt att använda flera objekt av samma
 * typ (här Commodity-objekt) i ett program. Varje objekt har sitt eget minnesutrymme.
 * @author Rolf
 */
public class CommodityEx2 {
   public void action() {
      Commodity com1 = new Commodity ();
      Commodity com2 = new Commodity ();

      com1.setName("Bacon”);
      com1.setCategory("Charkuteri”);
      com1.setQuantity(100);
      com1.setPrice(10.95);

      com2.setName("Russin”);
      com2.setCategory("Kolonial”);
      com2.setQuantity(520);
      com2.setPrice(14.95);

      com1.info();
      com2.info();
   }

   public static void main(String[] args) {
      CommodityEx2 prog = new CommodityEx2();
      prog.action();
   }
}
```



com1 och **com2** utgör länkar till Commodity-objekten i minnet

CommodityEx2.java

Modell av en bok

En bok kan beskrivas med följande egenskaper:

- titel t.ex. ”Java Direkt”
- författare t.ex. ”Jan Skansholm”
- isbn t.ex. ”9144038437”

Book
- title : String - author : String - isbn : String
+init(String,String,String) +info();

```
public class Book {
   privateString title;
   privateString author;
   privateString isbn;

   public void init( String title, String author, String isbn ) {
      this.title = title;
      this.author = author;
      this.isbn = isbn;
   }

   public void info() {
      System.out.println(”Title: ” + this.title + ”\nAuthor: ” + this.author + ”\nISBN: ” + this.isbn);
   }
}
```

I ett program används Book-objekt enligt samma regler som Vara-objekt:

```
Book book = new Book(); // Ett Book-objekt skapas. Variabeln book tilldelas referensen.
book.init(”Djup”, ”Henning Mankell”, ”9173430897”);
book.info();
```

BookEx.java

Book.java

Modell av reskostnader

Kostnaderna för en bilsemester kan beskrivas med följande egenskaper:

- bensin
- logi
- mat
- övrigt

TravelCosts
- petrol : double - lodging : double - food : double - otherCosts : double
+buypetrol(double) +buyLodging(double) +buyFood(double) +otherCosts(double) +travelInfo()

```
public class TravelCosts{
   private double petrol;
   private double lodging;
   private double food;
   private double otherCosts;

   public void buyPetrol(double cost) {
      this.petrol = this.petrol + cost;
   }

   public void buyLodging(double cost) {
      this.lodging = this.lodging + cost;
   }

   // fler metoder

   public void travelInfo() {
      double total = this.petrol+ this.lodging + this.food + this.otherCosts;
      String message = "KOSTNADER PÅ RESA\n" + …;
      JOptionPane.showMessageDialog(null,message);
   }
}
```

TravelCosts.java

TravelEx.java