

# Lektionstillfälle 1

## OOP

Avancerad Javaprogrammering

Utbildare: Mikael Olsson

# En människa ser en bild:



skallig  
ansikte  
orange  
näsa  
En person  
bild  
blått  
Ser ut som tarmar  
hjärnlober  
kind  
genomskärningsbild  
mun  
hjärna  
huvud

# En dator ser en bild



```
10101000101010010100101010001010
01001001001010010101000100010100
10100100101001010010010100101001
01001010010101000010010100100010
10101010101010010010101010010101
01001010101010101001101010001010
10010100101010001010010010010010
10010101000100010100101001001010
01010010010100101001010010100101
01000010010100100010101010101010
10010010101010010101010010101010
10101001101010001010100101001010
10001010010010010010100101010001
00010100101001001010010100100101
00101001010010100101010000100101
00100010101010101010100100101010
10010101010010101010101010010101
00010111101110100101010100100101
```

# Objektorientering

- Information i en dator modelleras som den mänskliga hjärnan strukturerar sina sinnesintryck.

# Våra hjärnor gillar ”saker”

- Saker är objekt
- Beskrivs som klasser i kod
- Ofta Substantiv
- Exempel från vår hjärnbild:
  - Hjärna
  - Person
  - Ansikte
  - Öga
  - Bild

# Objekt har attribut

- T.ex "Ett öga har en färg"
- Att ett objekt har ett visst attribut beskrivs, i kod, genom att en klass har en instansvariabel
- Ibland används "egenskap" synonymt med attribut

```
class Öga{  
    String färg;  
    public Öga (String färg){  
        this.färg = färg;  
    }  
}
```

# Objekt har objekt

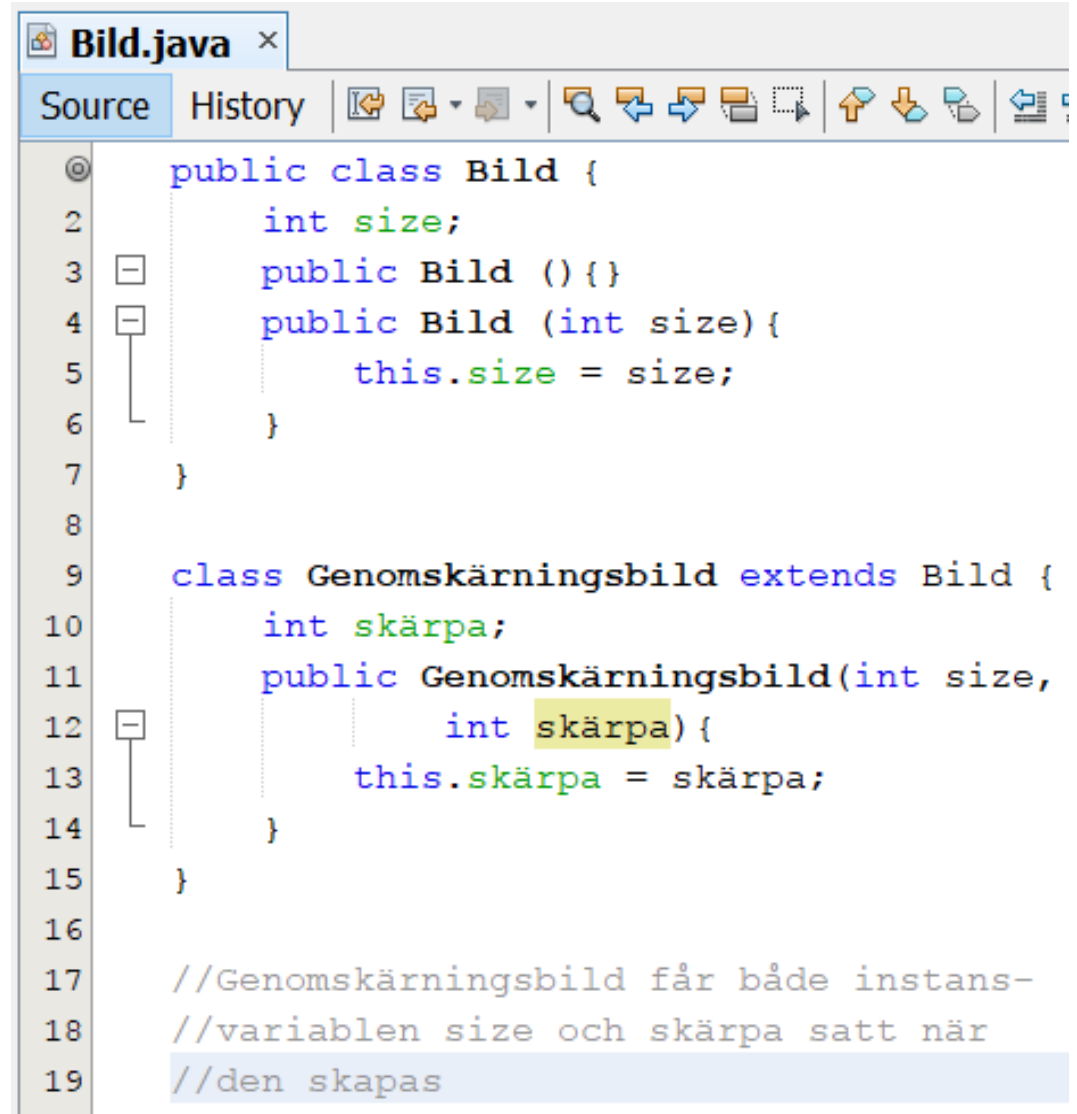
- Objekt kan **ha** andra objekt
  - Ex: Ett Huvud *har* två Ögon
  - "A har B" uttrycks genom att **A har instansvariabler av typ B**

```
class Huvud {  
    private Öga högerÖga;  
    private Öga vänsterÖga;  
    public Huvud(Öga h, Öga v) {  
        högerÖga = h;  
        vänsterÖga = v;  
    }  
}
```

```
class Öga{  
    String färg;  
    public Öga (String färg){  
        this.färg = färg;  
    }  
}
```

# Objekten ÄR ...

- Ett objekt kan **vara** ett annat objekt
  - Ex: En katt är ett däggdjur
  - Ex: Genomskärningsbild är också en Bild
  - Generalisering/specialisering
  - "A är B" uttrycks genom **A ärver B**
  - Arv betecknas med **extends**



```
Bild.java x
Source History
public class Bild {
    int size;
    public Bild (){}
    public Bild (int size){
        this.size = size;
    }
}

class Genomskärningsbild extends Bild {
    int skärpa;
    public Genomskärningsbild(int size,
        int skärpa){
        this.skärpa = skärpa;
    }
}

//Genomskärningsbild får både instans-
//variablen size och skärpa satt när
//den skapas
```



# Inkapsling

- Hemlig inre implementation
  - Privata instansvariabler – bara Bild själv har tillgång till size
- Känd yttre implementation
  - Publika metoder som andra objekt får prata med – setSize(int newSize)

```
public class Bild {  
  
    private int size;  
  
    public Bild () {}  
  
    public Bild (int size) {  
        this.size = size;  
    }  
  
    public void setSize(int newSize) {  
        size = newSize;  
    }  
}
```

# Separation of Concerns

- Ett objekt ska göra det som angår sig själv – inget annat!
- Fördelar
  - Mer lättförståelig kod
  - Lättare att återanvända kod

# Flödet i ett programmeringsprojekt

- Analys
  - Vad är det egentligen jag ska göra?
  - Vilka krav finns?
- Design
  - Vilka objekt behövs?
  - Hur ser objektens inbördes relationer ut?
  - Vika instansvariabler och metoder ska mina objekt ha?
  - Rita gärna klass- och flödesdiagram.
- Implementation
  - Programmera!
- Testa och verifiera att programmet funkar

# Övningsuppgift 1

- Konstruera en klass Person. En person ska ha ett namn, adress och ålder. Utforma en konstruktor och några lämpliga metoder för Person.
- Konstruera en klass Bilägare som ärver Person.
- Konstruera en klass Bil med registreringsnummer, modell och märke. En bil ska ha en Bilägare. Konstruera metoder som anropas när man köper eller säljer en bil (alltså byter bilägare).
- Lägg till kod som skapar upp några bilar och bilägare. Låt bilägarna köpa och sälja ett par bilar i programmet. Skriv, till sist, ut alla bilägarna samt vilka bilar de (eventuellt) äger.
- **OBS – lösningen ska följa Best Practices för inkapsling**

# Övningsuppgift 1, lösning Person och Bilägare

```
package övningsuppgift1;

public abstract class Person {

    private String namn;
    private String adress;
    private int ålder;

    Person(String namn, String adress, int ålder){
        this.namn = namn;
        this.adress = adress;
        this.ålder = ålder;
    }

    public String getName() {
        return namn;
    }

}
```

```
package övningsuppgift1;

public class Bilägare extends Person {

    Bilägare(String namn, String adress, int ålder) {
        super(namn, adress, ålder);
    }

}
```

# Övningsuppgift 1

## Klassen Bil

```
public class Bil {  
    private String registreringsnummer;  
    private String bilmärke;  
    private Bilägare ägare;  
  
    public Bil(String registreringsnummer, String bilmärke) {  
        this.registreringsnummer = registreringsnummer;  
        this.bilmärke = bilmärke;  
    }  
  
    public void ägsAv(Bilägare nyeÄgaren) {  
        ägare = nyeÄgaren;  
    }  
  
    public void såld() {  
        ägare = null;  
    }  
  
    public Bilägare getBilägare() {  
        return ägare;  
    }  
  
    public String getRegNummer() {  
        return registreringsnummer;  
    }  
  
    public String getSort() {  
        return bilmärke;  
    }  
}
```

```

public class Övningsuppgift1 {
    public void printBil(Bil bil){
        if (bil.getBilägare() == null){
            System.out.println("Bilen med regNummer" + bil.getRegNummer() + " har ingen ägare");
        }
        else {
            System.out.println("Bilen med regNummer " + bil.getRegNummer() + " är av typen "
                + bil.getSort() + " och ägs av " + bil.getBilägare().getName());
        }
    }
}

```

```

Övningsuppgift1 (){
    Bilägare Bosse = new Bilägare("Bosse", "Bilvägen 3", 65);
    Bilägare Lisa = new Bilägare("Lisa", "Laduvägen 8", 27);
    Bilägare Kim = new Bilägare("Kim", "Kalasvägen 6", 37);
    Bil rödaSaaben = new Bil("XYZ 123", "Saab");
    Bil vitaVolvon = new Bil("ERT 432", "Volvo");

    rödaSaaben.ägsAv(Bosse);
    vitaVolvon.ägsAv(Lisa);
    rödaSaaben.såld();
    rödaSaaben.ägsAv(Kim);

    printBil(rödaSaaben);
    printBil(vitaVolvon);
}

```

```

public static void main(String[] args) {
    Övningsuppgift1 övningsuppgift1 = new Övningsuppgift1();
}

```

Output:

Bilen med regNummer XYZ 123 är av typen Saab och ägs av Kim  
 Bilen med regNummer ERT 432 är av typen Volvo och ägs av Lisa

# Övningsuppgift 1, kommentar konstruktor

- Lägg märke till hur Bilägare kommer åt att sätta Fordons privata instansvariabler genom att anropa Fordons publika konstruktor mha `super (...)`.
- När en instans av en subklass skapas, skapas alltid automatiskt en instans av var och en av dess superklasser först, start med den översta
  - tänk ryska dockor, vi börjar med den innersta och jobbar oss utåt
- Ett explicit anrop till en superklass konstruktor via `super (...)` måste alltid stå på första raden i subklassens konstruktor
- Vid implicit anrop (när subklassen inte aktivt anropar superklassens konstruktor) måste en tom konstruktor finnas hos superklassen
  - `Person () {}`



# Sammanfattning

- Objektorientering är att modellera kod efter mänskligt tänkande
- Objekt kan oftast beskrivas med substantiv
- Ett objekts attribut uttrycks med instansvariabler
- "A är B" uttrycks genom **att A ärver B** (med "extends")
- "A har B" uttrycks genom **att A har instansvariabler av typ B**

# Framåtblick inför nästa lektion

- Arvshierarkier
- Abstrakta superklasser
- Interface
- Polymorfism
- Dynamisk bindning