Laboration 13

Uppgift 13a

Klassen ***Media*** är given:

Media
–id : long –title : String
+getId() : long +setId(long) +getTitle() : String +setTitle(String)

```
package laboration13;

public class Media {
    private long id;
    private String title;

    public long getId() {
        return id;
    }

    public void setId( long id ) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
}
```

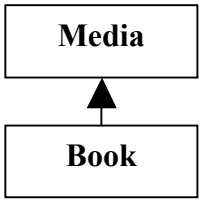
Testkör klassen *Media* med nedanstående instruktioner:

```
Media media = new Media();
media.setId( 837884976 );
media.setTitle( "En fantastisk värld" );
String res = media.getTitle() + "\n" +
             "toString-metoden: " + "\n" +
             media.toString();
javax.swing.JOptionPane.showMessageDialog( null, res );
```

Av testprogrammet och körresultatet kan du se att det finns en metod, ***toString()***, vilken anropas av Media-objektet. *toString*-metoden kan anropas med vilket objekt som helst och ger en utskrift på formen klassnamn@.... Hur detta går till återkommer vi till på nästa föreläsning, men det har med arv att göra.

Uppgift 13b

Klassen ***Book*** är given. Klassen ärver av klassen ***Media***:



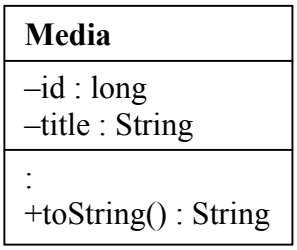

```
package laboration13;

public class Book extends Media {
}
```

Testkör klassen *Book* med samma instruktioner som i Uppgift 13a men ändra i första raden

```
Book media = new Book();
media.setId( 837884976 );
media.setTitle( "En fantastisk värld" );
String res = media.getTitle()+ "\n" +
             "toString-metoden: " + "\n" +
             media.toString();
javax.swing.JOptionPane.showMessageDialog( null, res );
```

Programmet ger ett liknande körresultat som det i Uppgift 13a. Den största skillnaden är att klassens namn har ändrats, från *laboration13.Media* till *laboration13.Book*.
Klassen Book har ärvt metoderna *setID*, *getID*, *setTitle* och *getTitle*. Metoderna anropas i testprogrammet.



Lägg till en ***toString***-metod i klassen *Media*:

```
public String toString() {
    String res = "Media: ID = " + this.id + ", Titel = " + this.title;
    return res;
}
```

Kör programmet i Uppgift 13a på nytt. Nu blir körresultatet som figuren till höger.

Som du ser används *toString*-metoden i klassen Media i det här fallet. Det finns ju en sådan i klassen.

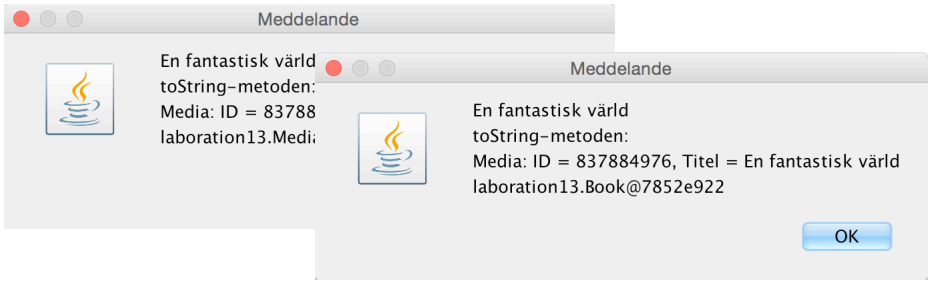
Kör programmet i Uppgift 13b (ovan). Körresultatet är samma som det för Uppgift 13a.
Även om objektet är av typen Book används *toString*-metoden i klassen Media. Klassen Book ärver ju klassen Media och därmed bl.a. metoderna *getTitle* och *toString*.

Ändra i ***toString***-metod i klassen *Media* till:

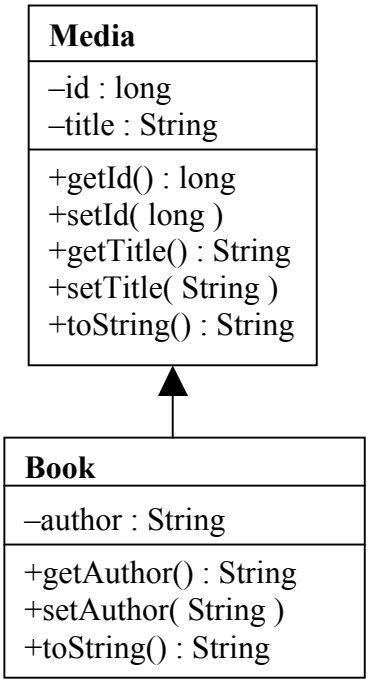
```
public String toString() {
    String res = "Media: ID = " + this.id + ", Titel = " + this.title + "\n" +
                 super.toString();
    return res;

}
```

Kör programmet i Uppgift 13a och Uppgift 13b på nytt. Körresultatet blir som figurerna till höger.



Uppgift 13c



Nu ska vi specialisera klassen ***Book*** så att den skiljer sig från ***Media***.

- Lägg till instansvariabeln *author* i klassen Book:
  `private String author;`
- Lägg till metoderna *getAuthor() : String* och *setAuthor(String)* i klassen Book.
- Lägg till metoden *toString() : String* i klassen Book. Metoden ska returnerar en sträng på formen: `"TITLE skriven av AUTHOR"`

Testkör klassen Book med nedanstående instruktioner:

```
Book media = new Book();
media.setId( 837884976 );
media.setTitle( "En fantastisk värld" );
media.setAuthor( "Alice Green" );
String res = media.getTitle() + " av " + media.getAuthor()
          + "\n" + "toString-metoden: " + media.toString();
javax.swing.JOptionPane.showMessageDialog( null, res );
```

I programmet används:

- Ärvda metoder från klassen Media (*setID*, *setTitle*, *getTitle*)
- Metoder implementerade i klassen Book (*setAuthor*, *getAuthor*, *toString*). Som du säkert märkte så användes "korrekt" version av *toString*-metoden, nämligen den i klassen Book. Om du avmarkerar metoden *toString* i klassen Book, kompilerar klassen Book och testkör ovanstående kod på nytt så ser du att *toString*-metoden i klassen Media används.

Uppgift 13d

När man ärver en klass så är klassen som regel färdigskriven och testad. Men av pedagogiska skäl ska du nu ändra i klassen ***Media*** för att se följderna av ändringen.

Lägg till en konstruktor i klassen Media:

```
public Media( long id, String title ) {
    // Lägg till kod
}
```

Kompilera klassen Media och testkör klassen med nedanstående kod:

```
Media media = new Media(837884976, "En fantastisk värld" );
String res = media.getTitle() + " är ett '" + media.getClass().getName() +
             "'-objekt" + "\n" + "toString-metoden: " + media.toString();
javax.swing.JOptionPane.showMessageDialog( null, res );
```

Körresultatet är samma som i Uppgift 1a. Enda skillnaden är ju att instansvariablerna får sina värde redan vid konstruktionen av Media-objektet.

Gå till klassen Book i Eclipse. Nu är det plötsligt en röd markering i kanten – när konstruktorn lades till i klassen Media så blev det ett fel i klassen Book. Vad beror detta på?

Innan du la till en konstruktor i klassen Media så fanns det ingen konstruktor i klassen. Det innebär att kompilatorn lade till en *default-konstruktor* i klassen:

```
public Media() {
}
```

På samma sätt lägger kompilatorn till en *default-konstruktor* i klassen Book vid konstruktion. Och kompilatorn placerar dessutom en instruktion i konstruktorn, ett anrop till en konstruktor i superklassen. En konstruktor i superklassen anropas ju alltid när ett objekt konstrueras.

```
public Book() {
    super(); // anrop av konstruktorn:  public Media()
}
```

Innan du la till en konstruktor i Media-klassen så fanns det en konstruktor utan parametrar i Media-klassen (default-konstruktorn). Det finns det inte längre och klassen Book är inte längre kompilerbar. För att åtgärda detta måste du lägga till en lämplig konstruktor i klassen Book, t.ex.:

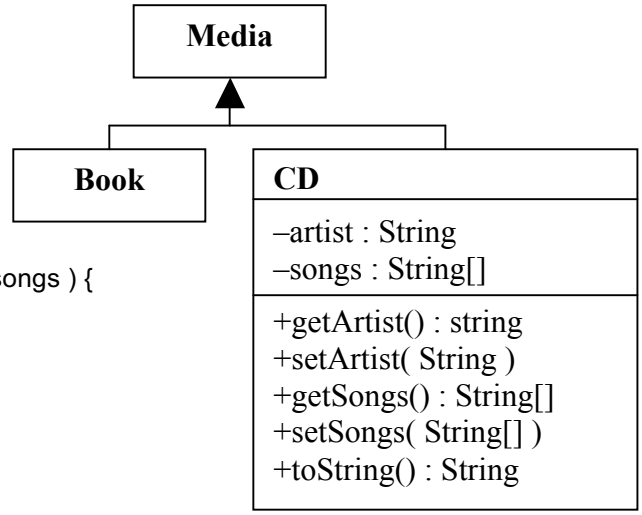
```
public Book( long id, String title, String author ) {
    super( id, title ); // Anrop av Media( long, String );
    this.author = author;
}
```

Lägg till ovanstående konstruktor i klassen Book, kompilera klassen Book och kör följande testprogram:

```
Book media = new Book( 837884976, "En fantastisk värld", "Alice Green" );
String res = media.getTitle() + " av " + media.getAuthor() + "\n" +
             "toString-metoden: " + media.toString();
javax.swing.JOptionPane.showMessageDialog( null, res );
```

Körresultatet ska vara samma som i uppgift 13c.

Uppgift 13e



Skriv klassen ***CD*** vilken ska ärva klassen ***Media***.

Klassen CD ska ha *två instansvariabler*:

- *artist* av typen String.
- *songs* av typen String[].

Klassen CD ska ha *en konstruktor*:

- ```
  public CD( long id, String title, String artist, String[] songs ) {
      super( id, title );
      this.artist = artist;
      this.songs = songs;
  }
  ```

Klassen ska ha följande *metoder*:

- *public void setArtist(String artist)*
- *public String getArtist()*
- *public void setSongs(String[] songs)*
- *public String[] getSongs()*
- *public String toString()* vilken ska skriva ut information om en CD på följande sätt:

  ```
  Artist: Ulrika Olsson
  Album: Sommar
  Melodier:
  1. a
  2. b
  3. c
  4. d
  5. e
  ```
  osv.

  det är lämpligt att börja med att bygga listan med melodier och därefter hela strängen som ska returneras, ungefär:

  ```
  String list = "", res;
  for-loop för samtliga melodier
     list += melodi + ny rad-tecken
  res = … + list;
  return res;
  ```

Testkör klassen CD med följande testprogram:

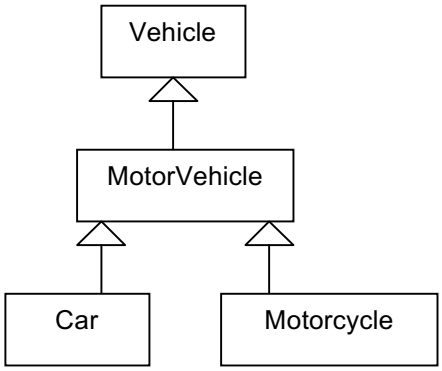
```
String[] mel = { "Mel 1", "Melodi 2", "Melodi 3", "Mel 4" };
CD cd = new CD( 39488852, "TITEL" ,"ARTIST", mel );
System.out.println( "----- Test av get-metoder -----" );
System.out.println( cd.getId() + ", " + cd.getArtist() + ", " +
                    cd.getTitle() );
System.out.println( "----- Test av toString -----" );
System.out.println( cd.toString() );
System.out.println( "----- Test av set-metoder -----" );
cd.setArtist( "Ulf Lundell" );
cd.setTitle( "Vargmåne" );
cd.setSongs( new String[]{ "M1", "M2", "M3", "M4", "M5", "M6" } );
System.out.println( cd.toString() );
```
```

Körresultat av testprogrammet på föregående sida:

```
----- Test av get-metoder -----
39488852, ARTIST, TITEL
----- Test av toString -----
Artist: ARTIST
Album: TITEL
Melodier:
1. Mel 1
2. Melodi 2
3. Melodi 3
4. Mel 4

----- Test av set-metoder -----
Artist: Ulf Lundell
Album: Vargmåne
Melodier:
1. M1
2. M2
3. M3
4. M4
5. M5
6. M6
```

Du ska jobba med en klasshierarki som ska se ut som figuren till höger:



Uppgift 13f

Skriv klassen **Vehicle** enligt beskrivningen nedan. Ägaren är av typen String.

Klassen Vehicle

instansvariabel:

- owner

metoder:

- konstruktorer
- setOwner
- getOwner
- toString

Skapa sedan en testklass med följande main-metod:

```
public static void main(String[] args){
    Vehicle v1= new Vehicle();
    Vehicle v2= new Vehicle("Lina Nilsson");
    System.out.println(v1.toString());
    System.out.println(v2.toString());
    v1.setOwner("Ola Torstensson");
    System.out.println(v1.getOwner());
}
```

Körning ska ge följande utskrift:

```
Ägare: Okänd ägare
Ägare: Lina Nilsson
Ola Torstensson
```

Uppgift 13g

Skriv klassen **MotorVehicle** enligt beskrivningen nedan. Hästkrafter är av typen int.

Klassen MotorVehicle:

är subklass till kassen Vehicle *(ärver från Vehicle)*

instansvariabler:

- hp (horsepower)

metoder:

- konstruktorer
- getHp
- setHp
- toString

toString-metoden ska anropa superklassens toString-metod.

Skapa en testklass med följande main-metod:

```
public static void main(String[] args){
    MotorVehicle m1= new MotorVehicle();
    MotorVehicle m2= new MotorVehicle("Lina Nilsson",100);
    System.out.println(m1.toString());
    System.out.println(m2.toString());
    m1.setOwner("Ola Torstensson");
    m1.setHp(200);
    System.out.println(m1.toString());
}
```

Körning ska ge följande utskrift:

```
Ägare: Okänd ägare Motorns hästkrafter: 0
Ägare: Lina Nilsson Motorns hästkrafter: 100
Ägare: Ola Torstensson Motorns hästkrafter: 200
```

Uppgift 13h

Skriv klasserna **Car** och **Motorcycle** enligt nedanstående beskrivningar:

Klassen Car

är subklass till klassen MotorVehicle *(ärver från MotorVehicle)*

instansvariabler:

- regNbr (registration number)

metoder:

- konstruktorer
- getRegNbr
- setRegNbr
- toString

Klassen Motorcycle

är subklass till klassen MotorVehicle *(ärver från MotorVehicle)*

instansvariabler:

- regNbr (registration number)

metoder:

- konstruktorer
- getRegNbr
- setRegNbr
- toString

Gör en testklass som har följande main-metod:

```
public static void main(String[] args) {
    Car car1 = new Car("Doris Bengtsson",80,"FGT 450");
    Motorcycle mc = new Motorcycle("Klas Bengtsson",70,"KKI 333");
    Car car2 = new Car();
    car2.setOwner("Fredrik Hansson");
    car2.setHp(100);
    car2.setRegNbr("HHH 778");
    System.out.println(car1);
    System.out.println(mc1);
    System.out.println(car2);
}
```

Körning ska ge följande utskrift:

```
Ägare: Doris Bengtsson Motorns hästkrafter: 80 Registeringsnummer: FGT 450
Ägare: Klas Bengtsson Motorns hästkrafter: 70 Registeringsnummer: KKI 333
Ägare: Fredrik Hansson Motorns hästkrafter: 100 Registeringsnummer: HHH 778
```

Uppgift 13i

Skapa ny klass **Bicycle** (cykel) med en ägare och ett antal växlar. Vad är det lämpligt att Bicycle ärver ifrån? Gör lämpliga metoder åt den nya klassen.

Testa denna klass genom att göra en testklass med en main-metod.

Extrauppgifter

Uppgift 13j

Lägg till metoden

```
public int compareTo(MotorVehicle m)
```

i klassen MotorVehicle.

Anropet

```
b1.compareTo(b2)
```

ska returnera 1 om b1 har en motor med fler hästkrafter än b2. -1 om b2 har en motor med fler hästkrafter än b1 och 0 om deras motorer har lika många hästkrafter.

Använd denna metod för att skapa en metod i en testklass som utifrån en lista av MotorVehicle och ett intervall returnerar en ny lista med de MotorVehicle som har motorer med antalet hästkrafter inom intervallet. Metodhuvudet ska se ut så här:

```
public MotorVehicle[] getInterval(MotorVehicle[] list,int min,int max)
```

Om det inte finns några MotoVehicle-objekt inom intervallet så ska en lista med längden 0 returneras.

Testa din metod.

Förslag till lösningar

Uppgift 13c

```
public class Book extends Media {
    private String author;

    public String getAuthor() {
        return this.author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String toString() {
           return getTitle() + " skriven av " + this.author;
    }
}
```

Uppgift 13d

```
public class Media {
    private long id;
    private String title;

    public Media( long id, String title ) {
        this.id = id;
        this.title = title;
    }
    :
}

public class Book extends Media {
    private String author;

    public Book( long id, String title, String author ) {
        super( id, title ); // Anrop av Media( long, String );
        this.author = author;
    }
    :
}
```

Uppgift 13e

```
public class CD extends Media {
    private String artist;
    private String[] songs;

    public CD(long id, String title, String artist, String[] songs) {
        super(id, title);
        this.artist = artist;
        this.songs = songs;
    }

    public String getArtist() {
        return this.artist;
    }

    public void setArtist(String artist) {
        this.artist = artist;
    }

    public String[] getSongs() {
        return this.songs;
    }

    public void setSongs(String[] songs) {
        this.songs = songs;
    }

    public String toString() {
        String list = "", res;
        for(int i=0; i<songs.length; i++) {
            list += (i+1) + ". " + songs[i] + "\n";
        }
        res = "Artist: " + this.artist + "\nAlbum: " + getTitle() +
              "\nMelodier:\n" + list;
        return res;
    }
}
```

Uppgift 13f – 13j

```
public class Vehicle {
    private String owner;

    public Vehicle() {
        this("Okänd ägare");
    }

    public Vehicle(String owner) {
        this.owner = owner;
    }

    public void setOwner(String owner) {
        this.owner = owner;
    }

    public String getOwner() {
        return owner;
    }

    public String toString() {
        return "Ägare: " + owner;
    }
}

-----------------------------------------------------
public class MotorVehicle extends Vehicle {
    private int hp;

    public MotorVehicle() {}

    public MotorVehicle(String owner, int hp) {
        super(owner);
        this.hp = hp;
    }

    public void setHp(int hp) {
        this.hp = hp;
    }

    public int getHp() {
        return hp;
    }

    public String toString() {
        return super.toString() + " Motorns hästkrafter: " + hp;
    }

    // Uppgift 13j
    public int compareTo(MotorVehicle m) {
        if(hp<m.getHp())
            return -1;
        else if(hp>m.getHp())
            return 1;
        else
            return 0;
    }

    // Uppgift 13j
    public static MotorVehicle[] getInterval(MotorVehicle[] list, int min,
int max) {
        int count = 0;
```

```
        MotorVehicle[] res;
        for(int i=0; i<list.length; i++)
            if( (list[i].getHp()>=min) && (list[i].getHp()<=max) )
                count++;
        res = new MotorVehicle[count];  // fungerar även då antal==0
        for(int i=list.length-1; i>=0; i--)
            if( (list[i].getHp()>=min) && (list[i].getHp()<=max) ) {
                count--;
                res[count] = list[i];
            }
        return res;
    }

    // Test av Uppgift 13j
    public static void main(String[] args){
        MotorVehicle[] fordon = {new MotorVehicle("A",100),
                new MotorVehicle("B",150),new MotorVehicle("C",80),
                new MotorVehicle("D",100),new MotorVehicle("E",110)};
        MotorVehicle[] resultat = getInterval(fordon, 95, 110);
        for(int i=0; i<resultat.length; i++)
            System.out.println(resultat[i]);
    }
}

-------------------------------------------------------
public class Car extends MotorVehicle {
    private String regNbr;

    public Car() {
        regNbr="";
    }

    public Car(String owner, int p, String regNbr) {
        super(owner, p);
        this.regNbr = regNbr;
    }

    public void setRegNbr(String regNbr) {
        this.regNbr = regNbr;
    }

    public String getRegNbr() {
        return regNbr;
    }

    public String toString() {
        return super.toString() + " Registreringsnummer: " + regNbr;
    }
}

-------------------------------------------------------
public class Motorcycle extends MotorVehicle {
    private String regNbr;

    public Motorcycle() {
        regNbr="";
    }

    public Motorcycle(String owner, int hp, String regNbr) {
        super(owner, hp);
        this.regNbr = regNbr;
    }
```

```
    public void setRegNbr(String regNbr) {
        this.regNbr = regNbr;
    }

    public String getRegNbr() {
        return regNbr;
    }

    public String toString() {
        return super.toString() + " Registreringsnummer: " + regNbr;
    }
}

-------------------------------------------------------
public class Bicycle extends Vehicle {
    private int gears=1;

    public Bicycle() {}

    public Bicycle(String owner, int gears) {
        super(owner);
        this.gears = gears;
    }

    public void setGears(int gears) {
        this.gears = gears;
    }

    public int getGears() {
        return gears;
    }

    public String toString() {
        return super.toString() + " Antal växlar: " + gears;
    }

    public static void main(String[] args){
        Bicycle c1= new Bicycle();
        Bicycle c2= new Bicycle("Sven Jakobi",21);
        System.out.println(c1);
        System.out.println(c2);
        c1.setOwner("Inga Bok");
        c1.setGears(3);
        System.out.println(c1.getOwner() + " " + c1.getGears() + "
växlar");
    }
}
```