

## Laboration 8

Avsikten med laborationen är att ska skriva klasser som innehåller instansvariabler, instansmetoder och konstruktorer. Speciellt tränas att använda attribut som är referenser till andra objekt.

Skapa paketet **laboration8** innan du fortsätter med laborationen.

### Uppgift 8a

Klassen **PhoneNumber** kan lagra ett antal telefonnummer (se klassdiagrammet nedan).

Klassen innehåller tre instansvariabler:

- *home* (String) – telefonnummer hem
- *work* (String) – telefonnummer arbetet
- *mobile* (String) – mobilnummer

Skriv klassen **PhoneNumber** i paketet *laboration8*.

Metoden **toString** ska returnera en sträng på formen:

```
Home:    040-123456  
Work:    040-333333  
Mobile:  0705223344
```

Om du exekverar nedanstående kod ska du få körresultatet nedanför koden.

```
PhoneNumber phone = new PhoneNumber( "0413-11111", "040-123456",  
"0735191919" );  
System.out.println( phone.getHome() + ", " + phone.getWork() + ", " +  
phone.getMobile() );  
System.out.println( "-----" );  
System.out.println( phone.toString() );  
System.out.println( "-----" );  
phone.setHome("040-222222");  
phone.setWork("");  
phone.setMobile("0707657654");  
System.out.println( phone.toString() );
```

#### Körresultat

```
0413-11111, 040-123456, 0735191919  
-----  
Home:    0413-11111  
Work:    040-123456  
Mobile:  0735191919  
-----  
Home:    040-222222  
Work:  
Mobile:  0707657654
```

#### PhoneNumber

- home : String  
- work : String  
- mobile : String

+PhoneNumber( String,  
String, String )

+ setHome( String )  
+ setWork( String )  
+ setMobile( String )  
+ getHome() : String  
+ getWork() : String  
+ getMobile() : String  
+ toString() : String

## Uppgift 8b

Skriv klassen **Contact** vilken ska representera en person i en telefonbok. Klassen innehåller två attribut, namnet på personen och ett *PhoneNumber*-objekt med personens olika telefonnummer.

- **Konstruktorn** ska ta emot fyra argument – namn, hemnummer, arbetsnummer och mobilnummer. Samtliga ska vara av typen String.
- **toString**-metoden ska returnera ett String-objekt på formen:  
Contact: Eva Bok  
Home: 0413-11111  
Work: 040-123456  
Mobile: 0735191919

Om du exekverar nedanstående kod ska du få körresultatet nedanför koden.

```
Contact contact = new Contact("Eva Bok","040-123456","040-333333","0705223344");
System.out.println( "Namn: " + contact.getName() + "\n" +
                    "Telefonnummer: " + contact.getHome() + ", " +
                    contact.getWork() + ", " + contact.getMobile() );
System.out.println("-----");
System.out.println( contact.toString() );
```

### Körresultat

```
Namn: Eva Bok
Telefonnummer: 040-123456, 040-333333, 0705223344
-----
Contact: Eva Bok
Home: 040-123456
Work: 040-333333
Mobile: 0705223344
```

### Contact

```
- name : String
- phone : PhoneNumber

+ Contact( String, String,
String, String )
+ getName() : String
+ getHome() : String
+ getMobile() : String
+ getWork() : String
+ toString() : String
```

## Uppgift 8c

Lägg till metoderna **setName**, **setHome**, **setWork** och **setMobile** i klassen **Contact**. Testa deras funktion genom att lägga till följande instruktioner efter koden i *Uppgift 8b*.

```
System.out.println("-----");
contact.setHome("040-999999");
contact.setWork("0415-22222");
contact.setMobile("0776192837");
System.out.println( contact.toString() );
```

### Körresultat

```
Namn: Eva Bok
Telefonnummer: 040-123456, 040-333333, 0705223344
-----
Contact: Eva Bok
Home: 040-123456
Work: 040-333333
Mobile: 0705223344
-----
Contact: Eva Bok
Home: 040-999999
Work: 0415-22222
Mobile: 0776192837
```

### Contact

```
- name : String
- phone : PhoneNumber

:
+ setName( String )
+ setHome( String )
+ setWork( String )
+ setMobile( String )
+ toString() : String
```

## Uppgift 8d

Klasserna **Building** och **Address** är givna (se klassdiagram till höger). Klassen **Building** representerar en byggnad och innehåller instansvariablerna *floors* (våningar) och *area* (bostadsyta). Klassen **Address** representerar en adress och innehåller instansvariablerna *street* (gata), *postalCode* (postadress) och *town* (ort).

Placera klasserna i paketet *laboration8* och testkör main-metoden i vardera klassen.

Skriv klassen **RealEstate** (fastighet, se klassdiagram till höger) vilken innehåller attribut för

- fastighetens beteckning (name),
- byggnaden (building)
- fastighetens adress (address)

Följande gäller för klassen:

- Klassen ska innehålla en **konstruktor** som tar argument för samtliga attribut (instansvariabler).  
**public RealEstate( String, Building, Address )**
- Klassen ska innehålla **get**-metoder till attributen.
- Klassen ska innehålla metoden **toString** vilken ska returnera ett String-objekt på formen:  
Fastighet: Bryggeriet 6  
Byggnad: 3 våningar, yta 343 kvm  
Adress: Algatan 4, 26733 Malmö  
*toString*-metoderna i klasserna *Building* och *Address* ska du använda i din lösning (testa dem – de returnerar lämpliga String-objekt).

Din lösning ska kunna köras med programmet:

```
Building building = new Building(5, 1350);
Address address = new Address("Nygatan 4", 12345,
"Ystad");
RealEstate re = new RealEstate("Kvarnen 4", building, address);
System.out.println(re.getName());
System.out.println(re.getBuilding().toString()); // toString i Building-
klassen
System.out.println(re.getAddress().toString()); // toString i Address-
klassen
System.out.println("-----");
System.out.println(re.toString());
```

### Körresultat

```
Kvarnen 4
Byggnad: 5 våningar, yta 1350 kvm
Adress: Nygatan 4, 12345 Ystad
-----
Fastighet: Kvarnen 4
Byggnad: 5 våningar, yta 1350 kvm
Adress: Nygatan 4, 12345 Ystad
```

Building
- floors : int - area : int
+ Building( int, int )  + getFloors() : int + getArea() : int + toString() : String

Address
- street : String - postalCode : int - town : String
+ Address( String, int, String ) + getStreet() : String + getPostalCode() : int + getTown() : String + toString() : String

RealEstate
- name : String - building : Building - address : Address
+ RealEstate( String, Building, Address ) + getName() : String + getBuilding() : Building + getAddress() : Address + toString() : String

## Uppgift 8e

Lägg till metoderna *setName*, *setBuilding* och *setAddress* i klassen *RealEstate*. Testa deras funktion genom att lägga till följande instruktioner efter koden i *Uppgift 8d*.

```
System.out.println("-----");
re.setName("Skorpionen 17");
re.setBuilding( new Building(6, 1470) );
re.setAddress( new Address("Nygatan 4", 55555, "Ystad") );
System.out.println(re.toString());
```

### Körresultat

```
Kvarnen 4
Byggnad: 5 våningar, yta 1350 kvm
Adress: Nygatan 4, 12345 Ystad
-----
Fastighet: Kvarnen 4
Byggnad: 5 våningar, yta 1350 kvm
Adress: Nygatan 4, 12345 Ystad
-----
Fastighet: Skorpionen 17
Byggnad: 6 våningar, yta 1470 kvm
Adress: Nygatan 4, 55555 Ystad
```

### RealEstate

```
- name : String
- building : Building
- address : Address

:
+ setName( String )
+ setBuilding( Building )
+ getAddress( Address )
+ toString() : String
```

## Uppgift 8f

Gör ett **klassdiagram** som beskriver klassen **Circle** nedan. Klassen **Point** är given och beskrivs av klassdiagrammet till höger.

```
public class Circle {
    private double radius;
    private Point position;

    public Circle( double inRadius, Point inPosition ) {
        this.radius = inRadius;
        this.position = inPosition;
    }

    public void setPosition( Point inPosition ) {
        this.position = inPosition;
    }

    public double getRadius() {
        return this.radius;
    }

    public int getX() {
        return this.position.getX();
    }

    public int getY() {
        return this.position.getY();
    }

    public Point getPosition() {
        return this.position;
    }

    public String toString() {
        return "CIRCLE, radius = " + this.radius + ", position = " +
        position.toString();
    }
}
```

### Point

```
- x : int
- y : int

+Point( int, int )
+ setX( int )
+ setY( int )
+ getX() : int
+ getY() : int
+ toString() : String
```

## Uppgift 8g

Placera klasserna **Circle** och **Point** i paketet *laboration8*. Kör nedanstående kod och studera körresultatet. Hur kan en ändring av innehållet i *p2* påverka cirkelns position?

```
Point p1, p2;  
p1 = new Point( 10, 12 );  
Circle c1 = new Circle( 3.5, p1 );  
System.out.println( "p1 = " + p1.toString() );  
System.out.println( "c1 = " + c1.toString() );  
System.out.println("-----");  
p2 = c1.getPosition();  
p2.setX( 333 ); // x-värdet i p2 ändras till 333  
System.out.println( "p1 = " + p1.toString() );  
System.out.println( "c1 = " + c1.toString() );  
System.out.println( "p2 = " + p1.toString() );
```

### Körresultat

```
p1 = (10,12)  
c1 = CIRCLE, radius = 3.5, position = (10,12)  
-----  
p1 = (333,12)  
c1 = CIRCLE, radius = 3.5, position = (333,12)  
p2 = (333,12)
```

Hur kan en ändring av x-värdet i *p2* påverka

- värdet på x i Point-objektet *p1*?
- cirkelns position?

## Uppgift 8h

Lägg till konstruktorn

```
public Point( Point point )
```

i klassen **Point**. Konstruktorn ska initiera instansvariablerna *x* resp *y* med motsvarande värde i argumentet *point*. Det nya Point-objektet ska alltså ha samma tillstånd som argumentet *point*.

Om du exekverar koden:

```
Point p1 = new Point( 10, 12 );  
Point p2 = new Point( p1 );  
System.out.println( p1.toString() );  
System.out.println( p2.toString() );
```

ska du få körresultatet till höger.

```
(10,12)  
(10,12)
```

## Uppgift 8i

Lägg till metoden

```
public Point copy()
```

i klassen **Point**. Metoden ska skapa ett Point som har samma tillstånd (värde i instansvariablerna) som aktuellt Point-objekt.

Om du exekverar koden:

```
Point p1 = new Point( 11, 13 );  
Point p2 = p1.copy(); // Det nya Point-objektet ska ha tillståndet x=11 och y=13 (samma som p1)  
System.out.println( p1.toString() );  
System.out.println( p2.toString() );
```

ska du få körresultatet till höger.

```
(11,13)  
(11,13)
```

### Uppgift 8j

Lägg till metoden

**public void setPoint( Point point )**

i klassen **Point**. Metoden ska tilldela instansvariablerna *x* resp *y* med motsvarande värde i argumentet *point*.

Om du exekverar koden:

```
Point p1 = new Point( 10, 12 );  
Point p2 = new Point( 20, 15 );  
p1.setPoint( p2 );  
System.out.println( p1.toString() );  
System.out.println( p2.toString() );
```

ska du få körresultatet till höger.

```
(20,15)  
(20,15)
```

### Uppgift 8k

Ändra i metoden

**public Point getPosition()**

i klassen **Circle** så att metoden returnerar en *kopia* av instansvariabeln *position*. Använd *copy*-metoden i klassen **Point**.

Om du exekverar koden i uppgift 8g så ska körresultatet bli:

#### Körresultat

```
p1 = (10,12)  
c1 = CIRCLE, radius = 3.5, position = (10,12)  
-----  
p1 = (10,12)  
c1 = CIRCLE, radius = 3.5, position = (10,12)  
p2 = (10,12)
```

Ändringen av *x*-värdet i *p2* påverkar inte *x*-värdet i *p1* eller cirkelns position längre. Hur kommer det sig?

## Uppgift 8l

Om du kör nedanstående kod så ändras Circle-objektets (*c1*:s) position när x-värdet i *p1* förändras. Detta beror ju på att *c1* använder samma Point-objekt som *p1* refererar till.

```
Point p1;  
p1 = new Point( 10, 12 );  
Circle c1 = new Circle( 3.5, p1 );  
System.out.println( "p1 = " + p1.toString() );  
System.out.println( "c1 = " + c1.toString() );  
System.out.println("-----");  
p1.setX( 444 );  
System.out.println( "p1 = " + p1.toString() );  
System.out.println( "c1 = " + c1.toString() );
```

### Körresultat

```
p1 = (10,12)  
c1 = CIRCLE, radius = 3.5, position = (10,12)  
-----  
p1 = (444,12)  
c1 = CIRCLE, radius = 3.5, position = (444,12)
```

För att åtgärda detta ska du se till att Circle-objektet alltid har ett eget Point-objekt, dvs. ett Point-objekt som det inte finns några externa referenser till.

Gör följande:

- Ändra konstruktorn *Circle( double, Point )* så att en kopia av Point-argumentet lagras i instansvariabeln *position*. Kopieringen finns det inga referenser till utom den i instansvariabeln *position*. En kopia får du genom att skapa ett nytt Point-objekt med vettiga argument / genom att anropa *copy*-metoden.
- Ändra metoden *setPosition( Point )* så att Point-objektet som *position* refererar till får samma x- resp y-värde som Point-argumentet.

Efter ovanstående ändringar ger ovanstående kod följande körresultat.

### Körresultat

```
p1 = (10,12)  
c1 = CIRCLE, radius = 3.5, position = (10,12)  
-----  
p1 = (444,12)  
c1 = CIRCLE, radius = 3.5, position = (10,12)
```

## Uppgift 8m

Uppgift 8m är fortsättning på uppgift 7o från föregående laboration. Så innan du löser uppgifterna 8m och 8n måste du lösa uppgifterna 7m, 7n och 7o.

Börja med att kopiera klassen *Picture* från paketet *laboration7* till paketet *laboration8*.

Din uppgift är att komplettera klassen *Picture* med fyra metoder:

**public boolean containsX(int minX, int maxX)**

Ska returnera *true* om bildens *x*-koordinat är i intervallet *minX-maxX*. Annars ska metoden returnera *false*.

**public boolean containsY(int minY, int maxY)**

Ska returnera *true* om bildens *y*-koordinat är i intervallet *minY-maxY*. Annars ska metoden returnera *false*.

**public void changeXDirection()**

Ska ändra *dx* till värdet av  $-dx$  (byta tecken på *dx*).

**public void changeYDirection()**

Ska ändra *dy* till värdet av  $-dy$  (byta tecken på *dy*).

Om du exekverar **Exercise8m**, när du kompletterat metoderna, ska bilderna studsas runt i fönstret.

## Uppgift 8n

Din uppgift är att skriva klassen **BouncingPicture**.

Klassen beskrivs av klassdiagrammet och nedanstående beskrivning.

**Instansvariabler:**

*window* : Instans av *Paintwindow* i vilken bilden ska rita sig.

*picture*: *Picture*-instans

*minX, maxX*: *Picture*-instansens *x*-värde ska hålla sig någorlunda inom intervallet *minX-maxX*.

*minY, maxY*: *Picture*-instansens *y*-värde ska hålla sig någorlunda inom intervallet *minY-maxY*.

**Instansmetoder:**

**setDx(int dx)**

anropar *setDx* i *Picture*-instansen

**setDy(int dy)**

anropar *setDy* i *Picture*-instansen

**showPicture()**

anropar *showImage* i *PaintWindow*-instansen

**move()**

ser till att ny position beräknas i *Picture*-instansen, att *Picture*-instansen visas i ny position och att *dx* respektive *dy* ändras i *Picture*-instansen om.

Om du exekverar **Exercise8n**, när du implementerat metoderna, ska bilderna studsas runt i fönstret (en av gubbarna i en del av fönstret).

BouncingPicture
- window : PaintWindow - picture : Picture - minx : int - maxX : int - minY : int - maxY : int
+ BouncingPicture( Paintwindow, Picture, int, int , int ,int )
+ setDx( int ) + setDy( int ) + addPicture() + move()



## Lösningsförslag

### Uppgift 8a

```
public class PhoneNumber {
    private String home;
    private String work;
    private String mobile;

    public PhoneNumber(String home, String work, String mobile) {
        super();
        this.home = home;
        this.work = work;
        this.mobile = mobile;
    }

    public void setHome(String home) {
        this.home = home;
    }

    public void setWork(String work) {
        this.work = work;
    }

    public void setMobile(String mobile) {
        this.mobile = mobile;
    }

    public String getHome() {
        return home;
    }

    public String getWork() {
        return work;
    }

    public String getMobile() {
        return mobile;
    }

    public String toString() {
        return "Home:   " + this.home + "\n" +
               "Work:    " + this.work + "\n" +
               "Mobile: " + this.mobile;
    }
}
```

### Uppgift 8b

```
public class Contact {
    private String name;
    private PhoneNumber phone;

    public Contact(String name, String home, String work, String mobile) {
        this.name = name;
        this.phone = new PhoneNumber(home, work, mobile);
    }

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

    public String getHome() {
        return this.phone.getHome();
    }
}
```

```
    }

    public String getWork() {
        return this.phone.getWork();
    }

    public String getMobile() {
        return this.phone.getMobile();
    }

    public String toString() {
        return "Contact: " + name + "\n" + this.phone.toString();
    }
}
```

### Uppgift 8c

```
public class Contact {
    :
    public void setName(String name) {
        this.name = name;
    }

    public void setHome(String home) {
        this.phone.setHome(home);
    }

    public void setWork(String work) {
        this.phone.setWork(work);
    }

    public void setMobile(String mobile) {
        this.phone.setMobile(mobile);
    }
    :
}
```

### Uppgift 8d

```
public class RealEstate {
    private String name;
    private Building building;
    private Address address;

    public RealEstate(String name, Building building, Address address) {
        this.name = name;
        this.building = building;
        this.address = address;
    }

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

    public Building getBuilding() {
        return this.building;
    }

    public Address getAddress() {
        return this.address;
    }

    public String toString() {
        return "Fastighet: " + this.name + "\n" +
```

```
        this.building.toString() + "\n" +  
        this.address.toString();  
    }  
}
```

### Uppgift 8e

```
public class RealEstate {  
    :  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public void setBuilding(Building building) {  
        this.building = building;  
    }  
  
    public void setAddress(Address address) {  
        this.address = address;  
    }  
    :  
}
```

### Uppgift 8f

Circle
<ul style="list-style-type: none"><li>– radius : double</li><li>– position : Point</li></ul>
<ul style="list-style-type: none"><li>+ Circle( double, Point )</li><li>+ setPosition( Point )</li><li>+ getRadius() : double</li><li>+ getX() : int</li><li>+ getY() : int</li><li>+ getPosition() : Point</li><li>+ toString() : String</li></ul>

---

### Uppgift 8g

När Circle-objektet skapas används variabeln *p1* som argument. Referensen i *p1* kopieras till parametern *inPosition*. Med instruktionen:

```
this.position = inPosition;
```

kopieras sedan samma referens till instansvariabeln *position*. Därför refererar *p1* och instansvariabeln *position* till samma Point-objekt.

Metoden *getPosition* returnerar referensen som lagras i *position*. Denna referens tilldelas nu variabeln *p2*. Nu refererar *p1*, *p2* och instansvariabeln *position* till samma Point-objekt.

### Uppgift 8h + i + j

```
public class Point {  
    :  
    // h  
    public Point( Point point ) {  
        this.x = point.getX();  
        this.y = point.getY();  
    }  
  
    // i
```

```
    public Point copy() {
        Point point = new Point( this.x, this.y);
        return point;
    }

    // j
    public void setPoint( Point point ) {
        this.x = point.getX();
        this.y = point.getY();
    }
    :
}
```

### Uppgift 8k + l

```
public class Circle {
    private double radius;
    private Point position;

    // l
    public Circle( double inRadius, Point inPosition ) {
        this.radius = inRadius;
        this.position = new Point(inPosition.getX(),inPosition.getY());
    }

    // k
    public Point getPosition() {
        return this.position.copy();
    }

    // l
    public void setPosition( Point inPosition ) {
        this.position.setX( inPosition.getX() );
        this.position.setY( inPosition.getY() );
    }
    :
}
```

### Uppgift 8m

```
package laboration8;
import javax.swing.ImageIcon;

public class Picture {
    :

    public boolean containsX(int minX, int maxX) {
        if(x>=minX && x<=maxX) {
            return true;
        } else {
            return false;
        }
    }

    public boolean containsY(int minY, int maxY) {
        return (y>=minY && y<=maxY);
    }

    public void changeXDirection() {
        dx = -dx;
    }
}
```

```
        public void changeYDirection() {  
            dy = -dy;  
        }  
    }  
}
```

## Uppgift 8n

```
package laboration8;  
import laboration6.PaintWindow;  
  
public class BouncingPicture {  
    private PaintWindow window;  
    private Picture picture;  
    private int minX;  
    private int maxX;  
    private int minY;  
    private int maxY;  
  
    public BouncingPicture(PaintWindow window, Picture picture, int minX, int maxX,  
int minY, int maxY) {  
        this.window = window;  
        this.picture = picture;  
        this.minX = minX;  
        this.maxX = maxX-picture.getIcon().getIconWidth();  
        this.minY = minY;  
        this.maxY = maxY-picture.getIcon().getIconHeight();  
    }  
  
    // Delegate methods  
    public void setDx(int dx) {  
        picture.setDx(dx);  
    }  
  
    public void setDy(int dy) {  
        picture.setDy(dy);  
    }  
    // Delegate methods  
  
    public void showPicture() {  
        window.showImage(picture.getIcon(), picture.getX(), picture.getY());  
    }  
  
    public void move() {  
        picture.move();  
        showPicture();  
        if(!picture.containsX(minX, maxX)) {  
            picture.changeXDirection();  
        }  
        if(!picture.containsY(minY, maxY)) {  
            picture.changeYDirection();  
        }  
    }  
}
```