# Repetition av viktiga begrepp inom objektorienterad programmering

*Tema:* Arv, parameteröverföring, statiska attribut och metoder.

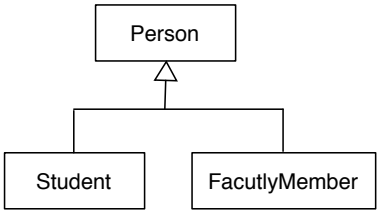
### Arv

U 1. Betrakta följande klasser:

```
public class Person {
    protected String name;

    public Person(String name) {
        this.name = name;
    }

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

    public String toString() {
        return name;
    }
}

public class Student extends Person {
    protected String program;
    protected int credits;

    public Student(String name, String program) {
        super(name);
        this.program = program;
        credits = 0;
    }

    public String toString() {
        return name + ",  " + program;
    }
    ...
}

public class FacultyMember extends Person {
    protected String department;

    public FacultyMember(String name, String department) {
        super(name);
        this.department = department;
    }

    public String toString() {
        return name + ", " + department;
    }
}
```

Antag att vi har gjort följande deklarationer:

```
Person p;
Student s;
FacultyMember f;
```




Figur 1: Superklassen Person med subklasserna Student och FacultyMember.

Avgör för var och en av följande satser om de är korrekta eller ej:

```
p = new Person("Lisa Svensson");
p = new Student("Lisa Svensson", "D");
s = new Person("Kalle Karlsson");
s = new Student("Kalle Karlsson","C");
s = new FacultyMember("Per Holm", "Computer Science");
p = s;
s = p;
f = s;
```

U 2. Antag att vi deklarerat en variabel `String info`. Ange vilket värde `info` får i var och en av de satser där den förekommer i en tilldelning i följande programrader:

```
Person p = new Person("Lisa Svensson");
info = p.toString();
p = new Student("Lisa Svensson","D");
info = p.toString();
Student s = new Student("Kalle Karlsson","C");
info = s.toString();
FacultyMember f = new FacultyMember("Per Holm", "Computer Science");
p = f;
info = p.toString();
```

U 3. Föregående uppgift handlade om polymorfism och metodanrop. Om vi deklarerar att en referensvariabel `ref` har en viss typ `C` enligt

```
C ref;
```

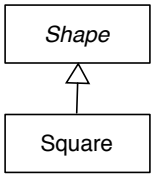
så får `ref` referera till objekt av klassen `C` eller till objekt av eventuella subklasser till `C`. Antag att det i både denna subklass och i klassen `C` finns en metod `p()`. Problemet som behandlades i föregående uppgift var att avgöra vilken av metoderna `p()` som exekveras vid ett anrop:

```
ref.p();
```

Formulera i egna ord hur detta avgörs.

U 4. Betrakta följande klasser:

```
public abstract class Shape {
   protected int x;
   protected int y;

   protected Shape(int x, int y) {
```




Figur 2: Den abstrakta klassen Shape och subklassen Square.

```
      this.x = x;
      this.y = y;
   }

   public void move(int dx, int dy) {
        x = x + dx;
        y = y + dy;
    }

    // övriga metoder
}

public class Square extends Shape {
   private int side;

   public Square(int x, int y, int side) {
      super(x, y);
      this.side = side;
   }

   public void draw() {
    // kod för att rita kvadraten
    }
}
```

Antag att vi har deklarerat två variabler `s` och `sq`:

```
Shape s;
Square sq;
```

Vilka av följande tilldelningssatser är korrekta?

```
sq = new Square(100, 100, 50);
sq = new Square(50);
s = new Shape(100, 100);
s = new Square(100, 100, 50);
```

U 5. Antag att vi lägger till ytterligare en subklass `Circle`. Då kan vi skriva och köra följande exempelprogram (som skapar några figurer och sedan flyttar dem):

```
public class Main {
    public static void main(String[] args) {
        Shape[] theShapes = new Shape[5];
        theShapes[0] = new Square(100, 300, 100);
        theShapes[1] = new Square(400, 200, 100);
        theShapes[2] = new Circle(400, 400, 50);
        theShapes[3] = new Square(450, 450, 50);
        theShapes[4] = new Circle(200, 200, 35);
        for (int i = 0; i < theShapes.length; i++) {
```

```
            theShapes[i].move(10, 10);
        }
    }
}
```

Om vi ändrar i programmet för att istället rita alla figurerna så fungerar det däremot inte:

```
        ...
        for (int i = 0; i < theShapes.length; i++) {
            theShapes[i].draw();
        ...
```

Förklara varför. Gör den ändring i klassen `Shape` som behövs för att det nya programmet ska fungera.

### Parameteröverföring

U 6. Givet en klass `C` med en metod `incr` enligt följande:

```
public class C {
    public void incr(int i) {
        i++;
    }
    ...
}
```

Vad skrivs ut när följande programrader exekveras?

```
int j = 2;
C c = new C();
c.incr(j);
System.out.println(j);
```

U 7. Antag att vi har en klass C med en metod `m` enligt följande:

```
public class C {
    public void m(Person p) {
        p = new Person("Kalle");
    }
}
```

Vad skrivs ut när följande programrader exekveras:

```
Person p = new Person("Lisa");
C c = new C();
c.m(p);
System.out.println(p);
```

U 8. Antag att metoden `m` i klassen `C` i stället har följande utformning:

```
public void m(Person p) {
    p.setName("Kalle");
}
```

Vad skrivs då ut om samma rader som i föregående uppgift exekveras?

U 9. Givet följande metod i någon klass `C`:

```
public static void changeElement(int[] a, int index, int newValue) {
    a[index] = newValue;
}
```

Vad skrivs ut när följande rader exekveras?

```
int[] a = {1, 2, 3, 4, 5};
C.changeElement(a, 3, 10);
System.out.println(a[3]);
```

U 10. Antag att följande (mer eller mindre vettiga) metoder finns i en klass `ArrayHandler`:

```
public static void r1(int[] a) {
   int[] temp = new int[a.length];
   for (int i = 0; i < a.length; i++) {
      temp[i] = a[a.length - 1 - i];
   }
   a = temp;
}

public static void r2(int[] a) {
   int[] temp = new int[a.length];
   for (int i = 0; i < a.length; i++) {
      temp[i] = a[a.length - 1 - i];
   }
   for (int i = 0; i < a.length; i++) {
      a[i] = temp[i];
   }

}
```

Vad har vektorerna `nbrs1` och `nbrs2` för värden efter det att följande rader exekverats?

```
int[] nbrs1 = {10, 20, 30, 40, 50};
int[] nbrs2 = {10, 20, 30, 40, 50};
ArrayHandler.r1(nbrs1);
ArrayHandler.r2(nbrs2);
```

### Statiska attribut och metoder

U 11. Antag att vi i klassen `Person` lägger till ett attribut och två metoder enligt följande:

```
protected static int seniorCitizenAge = 67;

public static void setSeniorAge(int i) {
    seniorCitizenAge = i;
}

public static int getSeniorAge() {
    return seniorCitizenAge;
}
```

Attributet representerar lagstadgad pensionsålder.

a) Motivera vad det innebär att attributet och metoderna deklarerats som `static`. Förklara också varför man valt att deklarera attributet `seniorCitizenAge` och de båda nya metoderna som `static`.

b) Vilka av metodanropen nedan är korrekta?

```
Person.setSeniorAge(65);
Person p = new Person(...);
p.setSeniorAge(65);
```

c) Vad skrivs ut när följande rader exekveras?

```
Person p = new Person(...);
Person q = new Person(...);
p.setSeniorAge(65);
System.out.println(q.getSeniorAge());
```