DA102A—F3: Klasser

Malmö universitet

Institutionen för datavetenskap och medieteknik

2025–??–??

Innehåll

- Objekt
  - Objekt sen tidigare
  - Vad är ett objekt?
  - Varför använda objekt?
- Skapa klasser
  - Skapa klasser
  - Konstruktor
  - Attribut
  - Operationer
  - Åtkomstidentifierare
- Sammanfattning

Objekt

Objekt sen tidigare
Vad är ett objekt?
Varför använda objekt?

Objekt sen tidigare

Vi har redan använt objekt tidigare, exempelvis `Scanner`

Andra objekt vi använt är `System.out` och eventuellt `Math`

I Java kan man lite förenklat säga att allt du kan sätta en punkt efter är ett objekt.

Vad är ett objekt?

Ett objekt är en *instans* av en klass.

Vad är en klass?

En klass är en mall för att skapa objekt.

Ett exempel

Tänk dig en cykel. Begreppet cykel är en klass.

En specifik cykel är ett objekt. Till exempel min cykel.

Ett exempel

Tänk dig en cykel. Begreppet cykel är en klass.

En specifik cykel är ett objekt. Till exempel min cykel.

Alla cyklar har vissa egenskaper, exempelvis färg och antal växlar. Dessa kallas *attribut*

Alla cyklar har också saker de kan göra, exempelvis rulla och bromsa. Dessa kallas *operationer*

Varför använda objekt?

Objekt hjälper oss att strukturera och organisera vår kod.

Objekt kan användas för att modellera verkliga saker och koncept.

Varför använda objekt?

Exempel

Tänk att vi ska skapa ett program för att hantera cyklar.

Vi hade då haft två alternativ:

1. Skapa variabler och funktioner för varje cykel
2. Skapa en klass `Cykel` och sedan skapa objekt av den klassen.

Skapa klasser

Skapa klasser
Konstruktor
Attribut
Operationer
Åtkomstidentifierare

Skapa klasser

I Java behöver varje klass ligga i sin egen fil.

Varje fil måste heta samma sak som klassen.

(Se tillbaka till tidigare genomgång.)

Skapa klasser

```java
public class Bike{
    private String colour; // Attribut
    private int gears; // Attribut
    public Bike(String colour, int gears){ // Konstruktor
        this.colour = colour;
        this.gears = gears;
    }
    public void roll(){ // Operation/metod
        System.out.println("Cykeln rullar");
    }
    public void brake(){ // Operation/metod
        System.out.println("Cykeln bromsar.");
    }
}
```

Skapa objekt

```
1 Bike myBike = new Bike("svart", 3); // Notera Bike i början
2 Bike yourBike = new Bike("röd", 7); // Notera new
3
4 myBike.roll();
5 yourBike.brake();
```

I koden ovan *instansierar* vi två objekt av klassen `Bike`

Sen använder vi objekten genom att anropa deras operationer.

Konstruktor

```
1 public class Bike{
2     public Bike(String colour, int gears){
3         // Code goes here
4     }
5 }
```

En konstruktor är en metod som anropas när ett objekt skapas.

Konstruktorn har samma namn som klassen och saknar returtyp.

Konstruktor

```
1 public Bike(String colour, int gears){ // Vår konstruktor
2     this.colour = colour;
3     this.gears = gears;
4 }
```

I konstuktorn brukar vi sätta värden på objektets attribut.

Nyckelordet `this` refererar till det aktuella objektet.

Om vi inte använder `this` skulle vi referera till parametrarna. Ha för vana att alltid använda `this` när du refererar till attribut.

Konstruktor

Flera konstruktorer

```
1  public class Bike{
2      public Bike(){
3          this.colour = "svart";
4          this.gears = 1;
5      }
6      public Bike(String colour, int gears){
7          this.colour = colour;
8          this.gears = gears;
9      }
10 }
```

Precis som metoder kan vi ha flera konstruktorer med olika parametrar.

Attribut

En variabel som är deklarerad i en klass kallas för ett attribut.

Attribut brukar deklareras överst i klassen.

```
1 public class Bike{
2     private String colour; // Attribut
3     private int gears; // Attribut
4     public Bike(String colour, int gears){
5         // Code goes here
6     }
7 }
```

Attribut

Åtkomstmodifierare

Kommandot `private` och `public` kallas åtkomstidentifierare.

Attribut bör nästan alltid vara `private`.

Ett `private` attribut kan endast nås inifrån klassen.

Operationer

En metod som är deklarerad i en klass kallas för en operation.

```
1 public class Bike{
2     public void roll(){ // Operation/metod
3         System.out.println("Cykeln rullar");
4     }
5     public void brake(){ // Operation/metod
6         System.out.println("Cykeln bromsar.");
7     }
8 }
```

Operationer

```
1 Bike myBike = new Bike("svart", 3);
2 myBike.roll();
```

Precis som att vi skriver `input.nextInt()` för att anropa metoden `nextInt` skriver vi `myBike.roll()` för att anropa metoden `roll`.

Operationer

I övrigt fungerar operationer precis som vanliga metoder.

Operationer

Static

Vi har tidigare använt statiska metoder. Exempelvis `Math.sqrt()`

Jämför vi dem med vanliga metoder så är skillnaden att en statisk metod inte behöver komma från ett initialiserat objekt.

När vi använder `Math.sqrt()` så är det inte kopplat till något objekt.

När vi har använt `input.nextInt()` så är det kopplat till objektet `input` av klassen `Scanner`.

Åtkomstidentifierare

public och private

Ett attribut eller en operation med `public` åtkomstidentifierare kan nås från andra klasser.

Ett attribut eller en operation med `private` åtkomstidentifierare kan endast nås inom den egna klassen.

Åtkomstidentifierare

Exempel

```
1  public class Bike{
2      public String colour; // Publikt attribut
3      private inte gears; // Privat attribut
4
5      public Bike(String colour, int gears){
6          this.colour = colour;
7          this.gears = gears;
8      }
9  }
```

Åtkomstidentifierare

Exempel

```
1 // Detta är en annan klass
2 Bike myBike = new Bike("svart", 3);
3 System.out.println(myBike.colour); // Fungerar
4 System.out.println(myBike.gears); // Fungerar inte
```

Åtkomstidentifierare

Get och Set

En vanlig standard är att göra attribut `private` och sedan skapa `public` `get`- och `set`-metoder för att läsa och ändra värdet på attributen.

```
1 public String getColour(){
2     return this.colour;
3 }
4 public void setColour(String colour){
5     this.colour = colour;
6 }
```

Sammanfattning

Sammanfattning

Klasser är mallar för att skapa *objekt*.

Man kan skapa flera objekt av samma klass.

Objekt har *attribut* (variabler) och *operationer* (metoder).

Attribut och operationer kan vara `public` eller `private`.

Konstruktorn är en särskild metod som anropas när ett objekt skapas.

`this` refererar till det aktuella objektet.

Sammanfattning

Objekt hjälper oss att strukturera och organisera vår kod.

Objekt kan användas för att modellera verkliga saker och koncept.

Vi kommer att kolla på hur vi kan använda flera klasser i samma program senare.

Det mesta som hjälper oss att läsa och strukturera vår kod är bra. Vi lägger mer tid på att titta på vår kod än att faktiskt skriva den.

Rekommenderad läsning

Dietel & Dietel, Kapitel 7, 298–338

Tove Janson, *Det osynliga barnet*