Föreläsning 1

- Allmän information
- Klassens struktur
- Variabler
- Metoder
- Konstruktorer

Praktisk info

Lärare: Edward Blurock(Kursansvarig, kursmaterial)
Email: edward.blurock@mah.se

Lärare: Phillip Söderwall (Inlämningsuppgifter, forum)
Meddelande via It's learning

Kursadministratör: Janet Stridh
Telefon: 040 – 6657314
Email: janet.stridh@mah.se

Kurssida: https://mah.itslearning.com/elogin/

Om kursen

Skriftlig tentamen:

Preliminärt i maj och augusti. De som önskar tentera på annan ort finner info om detta på kurssidan (Kursplanering).

De som tenterar här på MAH ska anmäla sig till tentamen via webben. Jag påminner via anslag på It's learning.

Kursmaterial:

Publiceras ett per vecka. Består ofta av av föreläsningsbilder, kommentarmaterial, exempelfiler och laboration.
Även kommande veckas material kommer i möjligaste mån att vara publicerat. Detta för att göra er studiesituation något mer flexibel.

Inlämningsuppgifter:

Under kursen ska ni lösa tre programmeringsuppgifter vilka lämnas in via kurssidan.

Projekt:

Veckorna 14 – 19 ska en större programmeringsuppgift lösas. Detta kan ske enskilt eller tillsammans med kursare.

Klassens struktur

En klass i java kan bl.a. innehålla:

Variabler	de kallas för ***instansvariabler*** eller ***attribut***
Konstruktorer	
Metoder	de kallas för ***instansmetoder*** eller kortare ***metoder***
Klasser	de kallas för ***inre klasser***

```
public class Class {
   // Instansvariabler (attribut)
   private int length;
   public double average;

   // Konstruktor
   public Class( int length) {
      this.length = length;
   }

   // Metod
   public int getLength() {
      return this.length;
   }

   // Inre klass
   private class InnerClass {
      // instansvariabler / konstruktorer / metoder
   }
}
```

Instansvariabel / Attribut

En klass kan innehålla variabler som beskriver viktiga egenskaper i klassen. Dessa variabler kallas för **instansvariabler** eller **attribut**.

En klass som ska beskriva en adress kan t.ex. innehålla attribut för gatuadress, postnummer och postadress.

```
public class Address {
    private String street;
    private int postalCode;
    private String town;
    :
}
```

En instansvariabel deklareras normalt som **private**. Detta ger störst kontroll över hur variabeln används. Endast kod i metoder som tillhör klassen kan använda instansvariabeln.

Det går även bra att deklarera en variabel som **public**. Då kan även kod som inte tillhör klassen avläsa och ändra variabelns värde. Detta är som regel inte bra och ska **unvikas**.

En tredje variant för synlighet är **protected**. protected kommer att behandlas i föreläsningen om arv.

En klass kan innehålla metoder

I klassen kan man skriva metoder som använder en eller flera av instansvariablerna.

```
public class Address {
   private String street;
   private int postalCode;
   private String town;

   public String toString() {
      return street + ”\n” + postalCode + ”\n” + town;
   }
}
```

När metoden **toString** anropas så används objektets instansvariabler, dvs vid anropet

addr.toString() används instansvariablerna i det objekt som *addr* refererar till

Address.java

AddressTest.java

set- och get-metoder

Med hjälp av set- och get-metoder får andra klasser tillgång till attributen. Set-metoder skriver man till de attribut vars värde ska kunna ändras och get-metoder för de attribut vars värde ska kunna avläsas.

```
public class Address {
    private String street;
    private int postalCode;
    private String town;

    // Ändrar värdet i attributet street
    public void setStreet(String street) {
        this.street = street;
    }

    // Returnerar värdet i attributet street
    public String getStreet() {
        return street;
    }

    // Ändrar värdet i attributet postalCode
    public void setPostalCode(int postalCode) {
        this.postalCode = postalCode;
    }

    :
}
```

Referensvariabel

```
Address school= new Address();
school.setStreet(”Östgatan 11”);
```

school är en speciell typ av variabel, nämligen en **referensvariabel**. En referensvariabel refererar till ett objekt. När variabeln deklareras anger man typen av objekt som variabeln ska kunna referera till. I ovanstående kod kan variabeln *school* referera till objekt av typen *Address*.

När variabeln tilldelats ett objekt (fått ett objekt att referera till) kan man använda variabeln för att använda det som är public-deklarerat i objektets typ (typ = klass).

Metoden *setStreet* är deklarerad som **public** och kan därmed anropas. Anropet sker med punkt-operatorn:

```
school.setStreet(”Östgatan 11”); // referensvariabel.metodnamn
```

Instansvariablerna är private-deklarerade och kan därmed inte nås via referensvariaqbeln skola.

~~`skola.street = ”Östgatan 11”;`~~ `// GÅR EJ, gatuadress är private`

Deklaration av variabler

Variabler kan deklareras på olika sätt.

- En l**okal variabel** deklareras i en metod.
- En **instansvariabel** deklareras i en klass (ej i metod).
- En **klassvariabel** deklareras i en klass (ej i en metod). Modifieraren **static** gör variabeln till en klassvariabel.

```
public class Ex {
   public static int max = 100;          // max - klassvariabel
   private int size = 0;                 // size - instansvariabel
   public int nbr = 1000;                // nbr - instansvariabel

   public void method1(int inc) {
      int newSize = size + inc;          // newSize - lokal variabel
      if(newSize > max) {
         int owerflow = newSize - max;   // overflow - lokal variabel
         :
      }
      :
   }

   private void method2() {
      String str;                        // str - lokal variabel
      :
   }
}
```

Variabler och synlighet

De olika typerna av variabler har olika synlighet. Med detta menas att de kan användas på olika sätt.

- En **lokal variabel** är synlig i blocket ({…}) där den deklareras. Dock endast efter deklarationen. När blocket exekverats färdig så tas variabeln bort ur minnet (stacken).
- En **instansvariabel** är synlig i samtliga metoder i klassen (och i deklarationer av instansvariabler som deklareras längre ner i klassen). Om variabeln är public så är den synlig även från andra klasser (via en variabel som refererar till ett objekt av klassen).
- En **klassvariabel** är synlig i samtliga metoder i klassen. Den kan även användas i deklaration av klassvariabler och instansvariabler som deklareras längre ner i klassen. Klassvariabeln är allmänt synlig om den deklareras som public. Med detta menas att kod placerad i andra klasser kan använda variabeln utan något objektet av klassen.
  Klassvariabeln tillhör klassen. Variabeln finns även om det inte skapats några objekt av klassen.

Variabler och synlighet

Färgmarkeringen visar var variabeln är synlig.

```
public class Ex {
   public static int max = 100;                    // max - klassvariabel
   private int size = 0;                           // size - instansvariabel
   public int nbr= 1000;                           // nbr - instansvariabel

   public void method1(int inc) {
      int newSize = size + inc;                    // nyttAntal - lokal variabel
      if(newSize > max) {
         int overflow = newSize – max;             // overflow - lokal variabel
         :
      }
      :
   }

   private void method2() {
      String str;                                  // str - lokal variabel
      :
   }
}

public class Ex2 {
   public void method3() {
      Ex.max = 25;                                 // OK
      Ex  ref = new Ex();                          // ref – lokal variabel
      ref.nbr = 2500;                              // OK
      :
   }
}
```

Klassmetoder

En metod som deklareras med modifieraren **static** är en **klassmetod**. En sådan metod tillhör klassen. Klassmetoden är allmänt tillgänglig om den är public-deklarerad. Med detta menas att den anropas från kod i andra klasser utan att det finns något objekt av klassen den tillhör.

Några exempel på klassmetoder är:
Math.random()
Math.cos(…)
Integer.parseInt(…)

Det som är typiskt för en klassmetod är att den är fristående från klassen den tillhör. Den får som regel all information som behövs via parametrar (kan dock erhålla information från t.ex. klassvariabler). T.ex. så behöver metoden Math.cos en vinkel för att returnera ett värde. Vinkeln anges vid anropet av metoden, t.ex. Math.cos(4).

En klassmetod kan inte använda instansvariabler eller instansmetoder i klassen. Instansvariabler och instansmetoder kräver ja att man skapat ett objekt av klassen.
Däremot går det naturligtvis bra att använda klassvariabler och klassmetoder som deklareras i samma klass eller i annan klass.

Klasskonstanter

Det går bra att deklarera konstanter av de enkla variabeltyperna. En konstant är en variabel vars värde inte kan förändras. En konstant deklareras med modifieraren **final**. Konstanten måste ges ett värde innan den används.

```
public class AClass {
   private final int CAPACITY = 1000;
   public final static double PI = 3.141592653589793;
   :

   public void enMetod() {
      final char CHR = 'U';
      :
   }
}
```

Det är vanligt att konstanterna deklareras med static i java. De kallas då för **klasskonstanter**. I ovantående exempel kommer man åt konstanten PI med hjälp av klassens namn, t.ex.:

```
area = AClass.PI * radius * radius;
```

Du kommer använda denna typ av konstanter bl.a. i samband med grafiskt användargränssnitt.

Vad är objektorienterad programmering?

- Organisera programmen som vi organisera verkligheten.
- Världen består av ting (objekt) som meddelar sig till varandra med hjälp av metoder.
- En klass är en beskrivning av en typ av objekt.

Exempel

Jag, Kristina, är en människa.

Jag är ett objekt som klassificeras som människa.

Jag har alla **egenskaper** som beskriver en människa, t.ex. blå ögonfärg. (**instansvariabler, attribut**)

Jag **kan** röra mitt lillfinger (**metod**)

Johan (ett annat objekt) kan säga åt mig att röra mitt lillfinger (anropa en metod som jag har).

En klass av objekt beskrivs genom dess egenskaper och vad de kan.

Varför objektorienterad programmering?

- **Inkapsling av kod**, en del metoder och sätt att spara information syns inte från utanför klasser. Man ska kunna ändra i en klass utan att andra klasser ska påverkas. Gör stora system stabilare. Endast gränssnittet viktigt för andra klasser.
- **Återanvändning av kod**, genom arv och att klasser används in nya program.
- **Att organisera koden** på ett sätt som påminner om verkligheten ger programmeraren möjlighet att greppa över större system utan att behöva förstå varje detalj.

Instans av objekt

När en instans av ett objekt skapas med **new** så sker följande:

- instansvariablerna nollställs.
  Numeriska variabler får värdet 0.
  char får värdet 0 vilket skrivs ut som ett blanktecken.
  boolean får värdet false.
  Referensvariabler får värdet null.
- Instansvariablerna initieras.
- En konstruktor exekveras. **Konstruktorn** ger användaren möjlighet att bestämma startvärdena I objektet som skapas, dvs. att initiera objektet. Konstruktorn fungerar som en metod men den anropas endast då objektet skapas.

Konstruktorer

- Konstruktorn har samma namn som klassen och ingen typ.

```
public class Address {
   private String street;
   private int postalCode;
   private String town;

   public Address(String street, int postalCode, String town) {
      this.street = street;
      this. postalCode = postalCode;
      this.town = town;
   }
   :
}
```

- Om man inte definierar någon konstruktor så skapas en konstruktor, utan parametrar, av systemet

```
public Address() {}
```

- Om man definierat flera konstruktorer används den som har matchande parameterlista.

- En konstruktor kan anropa en annan konstruktor i klassen. Detta ska på första raden i konstruktorn.

```
public Address() {
   this(”Algatan 3”,28634, ”Tornerup”);
   // har kan vara mer kod som exekveras när den anropade konstruktorn exekverat färdigt
}
```

varvid konstruktorn

```
public Address(String street, int postalCode, String town) {…}
```

anropas.

Konstanta objekt

Ett konstant objekt:

- Har lämpliga konstruktorer för att initiera instansvariabler
- Har inga metoder som ändrar värdet på instansvariablerna.

Klassen **ConstantPoint** får sin position vid konstruktion och sedan kan inte positionen ändras. Metoden innehåller endast metoder för att avläsa x- respektive y-koordinaten.

```
public class ConstantPoint {
   private int x;
   private int y;

   public ConstantPoint (int x, int y) {
      this.x = x;
      this.y = y;
   }

   public int getX() {
      return x;
   }

   public int getY() {
      return y;
   }
}
```

Objekt av typen **String** är konstanta objekt i java. Klassen String innehåller inga metoder för att ändra på strängens innehåll.

När man t.ex. använder +-operatorn

```
String str = ”Elin”;
str = str + ” är ett namn”;
```

så skapas ett nytt String-objekt av String-objekten **”Elin”** och **” är ett namn”**.