TDDC77 Objektorienterad Programmering

Föreläsning 5

Sahand Sadjadee
IDA, Linköpings Universitet

Outline

- Introduktion till Objektorienterad programmering
- Klasser, objekt och paket
- Instansiering
- ArrayList

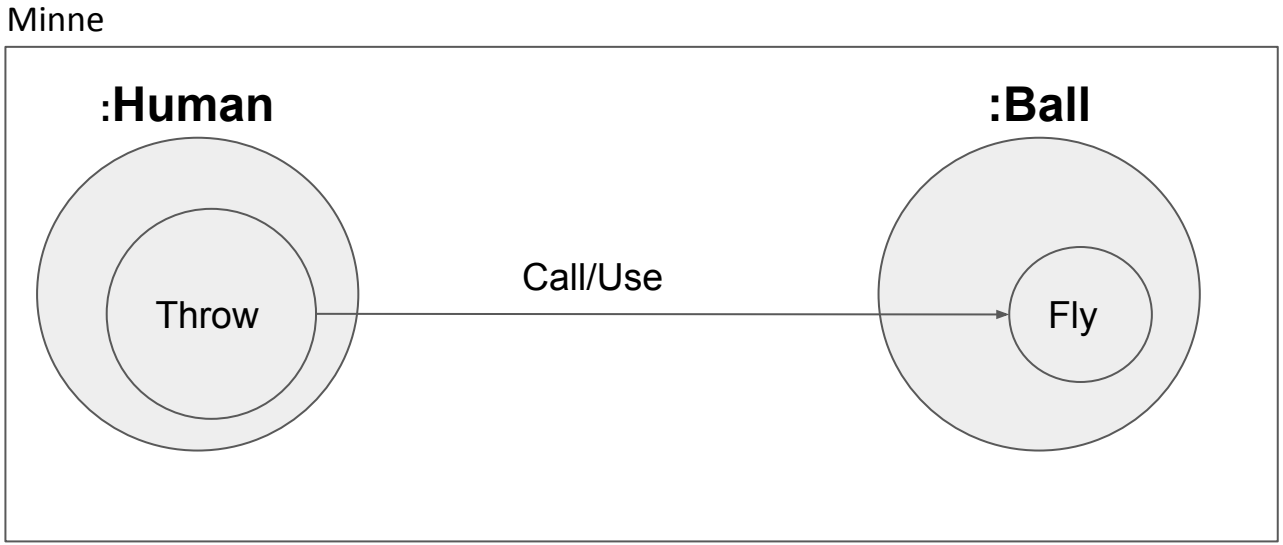
Introduktion till Objektorienterad Programmering

LINKÖPINGS UNIVERSITET

Objektorienterade program

- Objektorienterade program består av ett eller flera objekt.
- Varje objekt har egna beteenden och egenskaper.
- Objekts beteenden kan anropa/använda andra objekts beteenden. Till exempel en människa kan kasta en boll. För att kunna **kasta** en boll, ska bollen kunna **flyga**.
- Objekt ligger i datorns minne, Heap. Objekt skapas medan programmet körs(Run-time).

Objektorienterade program



Klasser

- En klass beskriver ett objekt liksom en blåkopia beskriver ett hus som inte har byggts än.
- En klass innehåller beskrivning för beteenden och egenskaper.
- Flera objekt kan skapas från en klass.
- Ett objektorienterat programmeringsspråk ska användas för att deklarera klasser. Till exempel Java och C#.

Pure objektorienterad programmering

Java kan användas för att implementera bara **objektorienterade program**.

En Java klass

- En Java klass innehåller **metoder(beteenden)** och/eller **fält/variabler(egenskaper)**.
- Varje klass har ett namn.
- En klass ligger i en fil med samma namn som klassen.
- Klasser deklareras genom att använda nyckelordet "`class`".
- Varje ord i klassnamnet börjar med en stor bokstav. Till exempel, `Human`, `HumanRace`, `AncientHuman`, ...

En Java klass

```
class Human{

    int throwBall(){


    }

    int height;

}
```

Klassens namn

Metodens namn

nyckelord

Variabelns namn

LINKÖPINGS UNIVERSITET

En metod

- En metod tar emot data(om behövs), gör beräkningar och returnera ett resultat(om behövs).
- Varje metod har ett namn.
- Varje metod har en signatur som följande:
  - `int walk(int meter) {`
  - `  // metodens kropp/body`
  - `  return 10;`
  - `}`

Typen på resultatet

Typen på emottagen data

- Signaturen är unik i klassen.
- Varje ord i metodnamnet börjar med en stor bokstav **förutom första ordet**. Till exempel, `walk`, `walkOnWater`, `walkSlowly`, ... Det är vanligt att metodnamnet börjar med ett verb.
- En metod innehåller en grupp av instruktioner. Varje instruktion/statement avslutas med en semikolon(;).

LINKÖPINGS UNIVERSITET

Ett enkelt program

- Körning avslutas när sista instruktionen i metoden `main` har anropats och är klar.
- Ett program kan bestå av flera klasser. Men men, det är bara en klass som innehåller metoden `main`.
- De klasserna som innehåller ingen main metod används direkt eller indirekt av klassen som innehåller `main` metoden.
- Obs! Instruktioner/statements som används för att implementera logiken KAN BARA LIGGA I METODER.

Egen identifierare

- Det finns konventioner.
  - Klassnamn: Alla ord i namnet börjar med stora bokstäver. Till exempel: `MyClass, AutoMobile, BusinessClass, NorthWest, Runnable, System`
  - Metodnamn:Alla ord i namnet börjar med stora bokstäver förutom första ordet. Till exempel: `walkOnWater, driveStraight, println, parseInt`
  - Variabelnamn: Alla ord i namnet börjar med stora bokstäver förutom första ordet. Till exempel: `totalNumber, favoriteColor, avergaeWeight`
- Alltid välj meningsfulla namn. `x` eller `y` är sällan bra namn.
- Ni kommer att snabbt glömma vad ni menade med `smUpTi`.
- För långa namn är också värdelösa.

Klasser, objekt och paket

Klasser

- Varje objekt som skapas måste tillhöra till en klass.
- Klassen definierar hur objekten fungerar, det vill säga deras beteende. Lite som en mall eller blåkopia för objekten.
- En klass deklareras i en separat fil.
- **Klassens namn blir objektens typ.**

```
/* Greeter.java
 * Enkel Klass som representera en person som
 * hälsa med ett specifikt meddelande
 */
class Greeter{
    private String message;

    public Greeter(String msg){
     message = msg;
    }

    private String speak(){
     return message + "!!!";
    }
}
```

LINKÖPINGS UNIVERSITET

Objekt

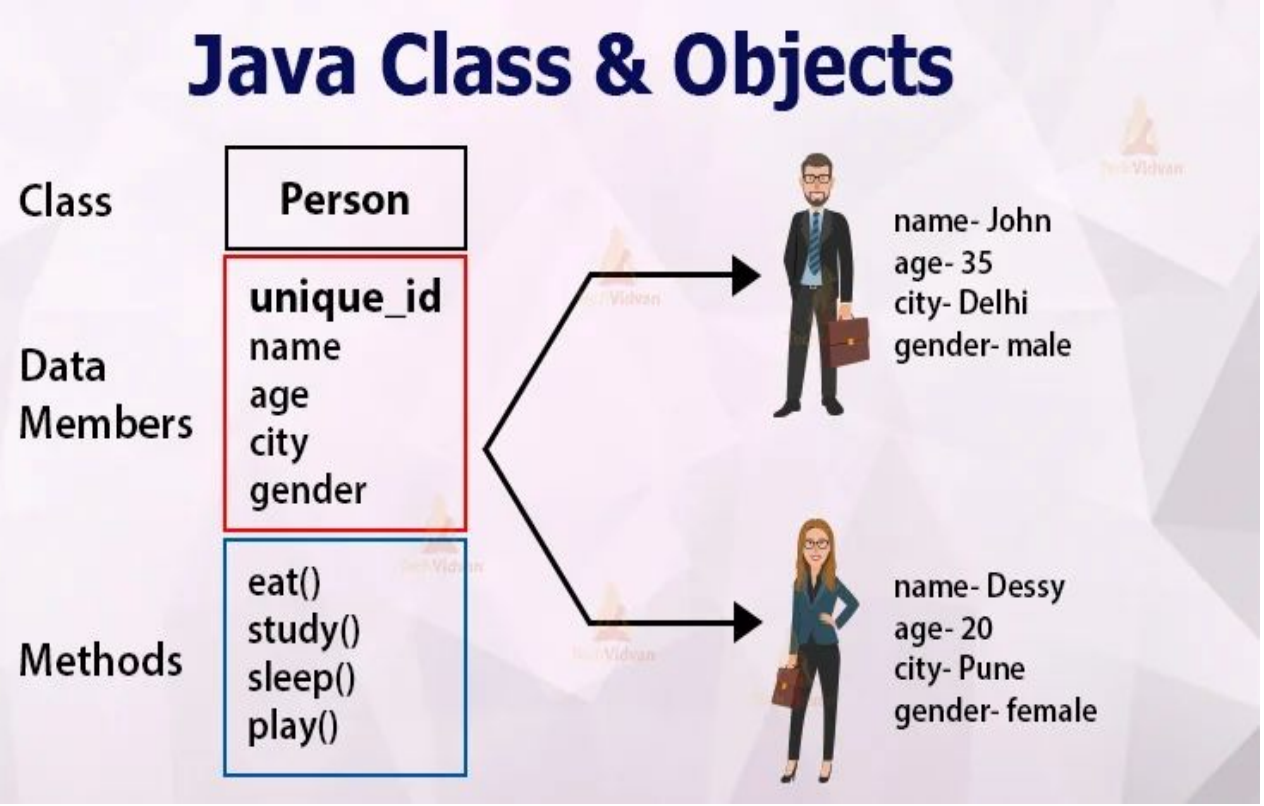
- Identitet : Typ
- Attribut: fält(variabler som deklareras direkt i klassen utanför metoderna)
- Beteenden: Metoder

Objekt

- Ett objekt kan även kallas för en instans.
- När man skapar ett objekt som tillhör till en viss klass så säger man att man instansierar klassen.
- Varje objekt tar sina egna “kopior” av variablerna som deklarerats i klassen. Dessa bestämmer objektets tillstånd.
- Ett objekt skapas med nyckelordet `new`.

```
/* Greeter.java
 * Enkel Klass som representera en person som
 * hälsa med ett specifikt meddelande
 */
class Greeter{
    private String message;

    public Greeter(String msg){
     message = msg;
    }

    private String speak(){
     return message + "!!!";
    }
}
```

Objekt


Java Class & Objects
Class
Person
Data Members
unique_id
name
age
city
gender
Methods
eat()
study()
sleep()
play()
name- John
age- 35
city- Delhi
gender- male
name- Dessy
age- 20
city- Pune
gender- female

Klasser VS Objekt

- Klasser skapas och sparas i filer.
- Objekt skapas i primärminnet.
- Klasser skapas av utvecklare under programmeringstiden.
- Objekt skapas av virtuell maskinen under körtiden.

Pac-man

Exampel

```
class Ghost {

    String name;
    int col=0,row=0;

    public Ghost(String name, int _col, int _row){
     this.name = name;
     col = _col;
     row = _row;
    }

    public void update(){
     col = (this.col==9? col-1: col+1);
     row = (row==9? row-1: row+1);
    }

    public String print(){
     return name + ": (" + row + ", " + col + ")";
    }
}
```

Paket(package)

- Ett paket är en mapp.
- Relaterade klasser ska ligga i samma mapp.
- Första raden i java filen ska innehålla en package-deklaration annars klassen tillhör till inget/default paket.
- Package deklaration består av först `package` nyckelordet följd med namnet på mappen där java-filen ligger.
- Namnet på mappen kan sig själv bestå av flera mappnamn som separeras av punkter.
- Paketnamn består bara av lilla bokstäver.

Paket

```
package se.liu.tddc77.lecture05;


public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub



    }

}
```

LINKÖPINGS UNIVERSITET

Instansiering

Konstruktorer

- En konstruktor är en speciell metod.
- En konstruktor saknar returtyp och heter samma som klassen.
- Om ingen konstruktor skrivs då skapas automatiskt en så kallad standardkonstruktor som inte tar argument.
- Om man skapar en egen konstruktor då skapas inte standardkonstruktorn automatiskt.
- Man kan ha godtyckligt många konstruktorer så länge de har olika parameterlistor.
- Detta gäller generellt för alla funktioner och kallas överlagring (tas upp senare).

Att instansiera en klass

- Man instansierar (skapar ett objekt av) en klass genom att anropa en konstruktor.
- Detta sker genom att använda nyckelordet `new` följt av klassnamnet och en parameterlista.
- Det kan finnas flera konstruktorer med olika parameterlistor.
- Ett separat objekt skapas när man instansierar en klass varje gång.

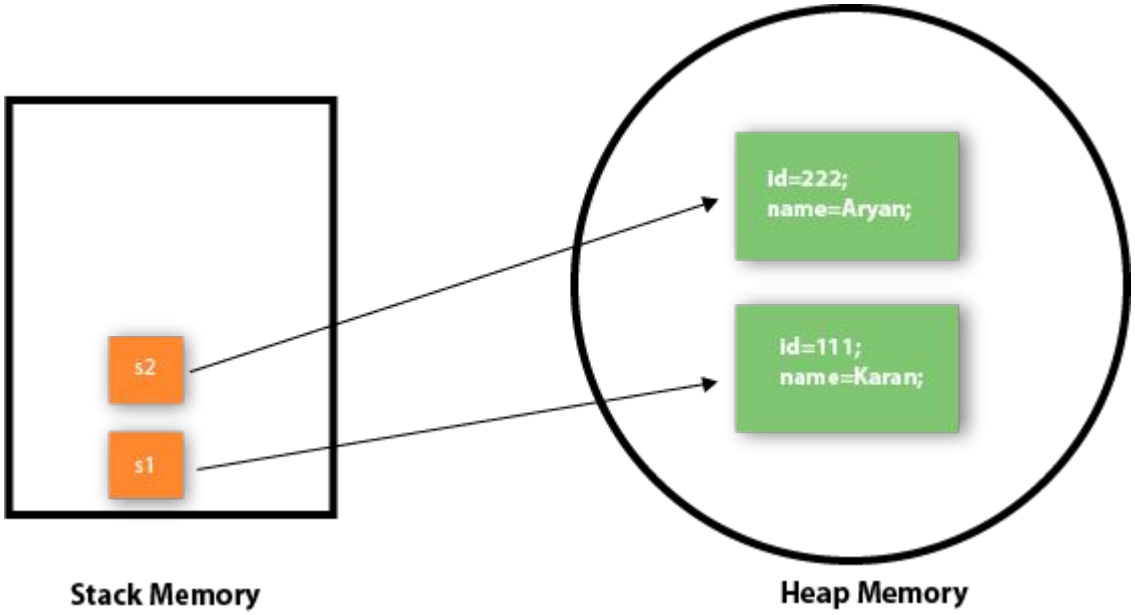
```
Greeter glad = new Greeter("hejhej");

Greeter compis = new Greeter("tjaba");
```

Referenser

- Klasser är referenstyper,det betyder att när man skapar ett objekt egentligen man får en referens tillbaka.
- Gör man sedan en tilldelning eller jämförelse så är det referensen som jämförs.

```
Greeter john = new Greeter("God morgon, jag heter John Smith!");
Greeter johnsmor = john;
Greeter johnsfar = johnsmor;
```

Exampel


s2
s1
Stack Memory
Id=222;
name=Aryan;
Id=111;
name=Karan;
Heap Memory

import

- Om man vill instansiera en klass från Java Standard Bibliotek då ska man **importera** den först.
- En import-instruktion kommer först i filen och utanför klassen.
- Man ska använda hela adressen till klassen.

```
package lec06;
import java.util.Scanner;

public class MainClass {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
    }
}
```

LINKÖPINGS UNIVERSITET

Exampel

```
import java.util.Scanner;

 class Game {
    static Scanner scan = new Scanner(System.in);
    static Ghost greenGhost = new Ghost("green", 1, 4);
    static Ghost blueGhost = new Ghost("blue ", 0, 3);

    public static void main(String[] args) {
        String command;
        do{
         //ask the ghosts to update there coordinates
         greenGhost.update();
         blueGhost.update();
         //print the new coordinates
         System.out.println(greenGhost.print());
         System.out.println(blueGhost.print());
         //check if the user wants to quit
         System.out.print("\nWrite \"c\" if you want to continue:
         command=scan.next();
     }while(command.equalsIgnoreCase("c"));
    }
}
```

LINKÖPINGS UNIVERSITET

`Static`

- Tidigare har vi skrivit `static` lite överallt.
- Från och med nu så kommer vi att använda oss mycket mindre av `static`.
- Vi ska fortfarande använda oss `static` när vi skriver main-metoden.

Två sorters fält(variabler)

- För att förstå vad nyckelordet `static` innebär behöver man veta att det i själva verket finns ytterligare ett sätt att indela variabler i två kategorier som anknyter till objekt och klasser.
- De två nya kategorierna kallas för **instans och klassvariabler**.

Instansvariabler

- En instansvariabel är individuell för varje instans av klassen.
- Det är denna sorts variabler ni i fortsättningen ska använda i nästan alla fall
- Deklareras utan nyckelordet `static`.

```
class Test1{
    int a = 0;

    void use(){
     System.out.println(a);
    }
}
```

LINKÖPINGS UNIVERSITET

Klassvariabler

- En klassvariabel är gemensam för alla instanser av en klass.
- Denna sorts variabler ska ni undvika mest.
- Deklareras med nyckelordet `static`

```
class Tests2{
    static int a = 0;

    void use(){
     System.out.println(Test2.a);
    }
}
```

LINKÖPINGS UNIVERSITET

Hur om metoder?

- Principen är samma för metoder.
- metoder deklarerade utan `static` hör samman med (och opererar på) enskilda instanser, anropas med `objekt.funktionsNamn()`
- metoder deklarerade med `static` hör samman med (och opererar på) klassen, anropas med `KlassNamn.funktionsNamn()`
- Detta innebär i förlängningen att metoder deklarerade med `static` inte kan komma åt variabler/metoder deklarerats utan `static` eftersom det inte är speciferat vilken instans instansvariabel som avses.

Punkt operatorn

- Punkt operator används för att komma åt klassmedlemmar eller instansmedlemmar
- KlassNamn.klassmedlem
  - Klassnamn.klassMetod()
  - Klassnamn.KlassVariabel
- objectReferens.instansmedlem
  - objektReferens.instansMetod
  - objektReferens.instansVariabel

Exempel?

- Kan du nämna några exempel på klassmetoder och instansmetoder du använt?
- Klassmetoder: Math.max(), Math.round(),
- Instansmetoder: input.nextInt();

ArrayList

ArrayList

- Längden på array kan inte justeras efter skapandet.
- ArrayList är en klass i standard bibliotek som implementerar en array.
- ArrayList kan användas när man inte vet längden på arrayen i förväg.

```
import java.util.ArrayList; // import the ArrayList class

ArrayList<String> cars = new ArrayList<String>(); // Create an ArrayList object
```

Tack för att du lyssnade!

LINKÖPINGS UNIVERSITET