TDDC77
Objektorienterad Programmering

Föreläsning 7

Sahand Sadjadee
IDA, Linköpings Universitet

Outline

- UML
- Arv (17.1, 17.2)
- Nyckelordet this(16.5)
- Komposition
- Enumerations

UML

UML

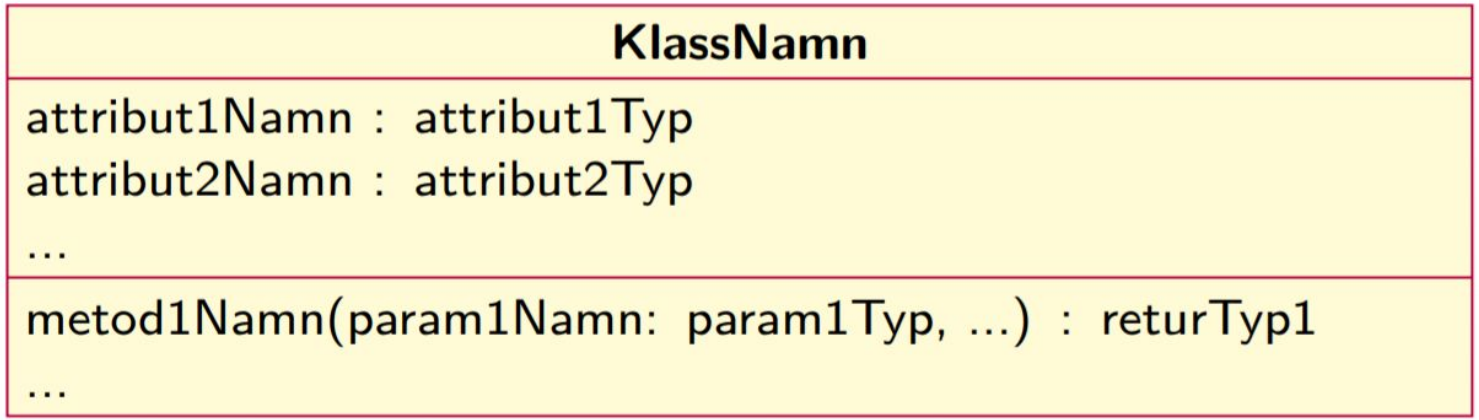
- UML = Unified Modeling Language
- En enorm standard för att modellera precis allt.
- En liten del kallas för klass-diagram och beskriver hur klasser hänger ihop med varandra.
- Vi kommer i kursen att använda en mycket begränsad delmängd av dessa klass-diagram kallad Lättviktsdiagram.

Lättviktsdiagram

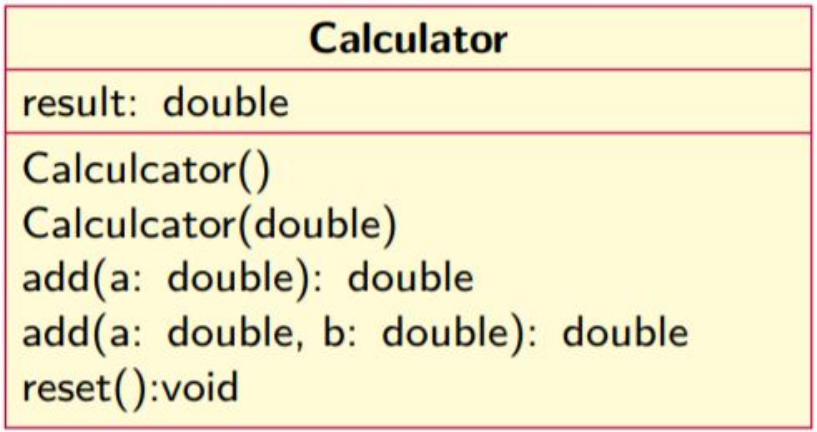
- Din handledare som ska rätta dina uppgifter behöver förstå din design och vill därför ha diagram som beskriver dina klasser och relationerna mellan dem.
- Ni behöver inte göra formellt korrekta UML-diagram.

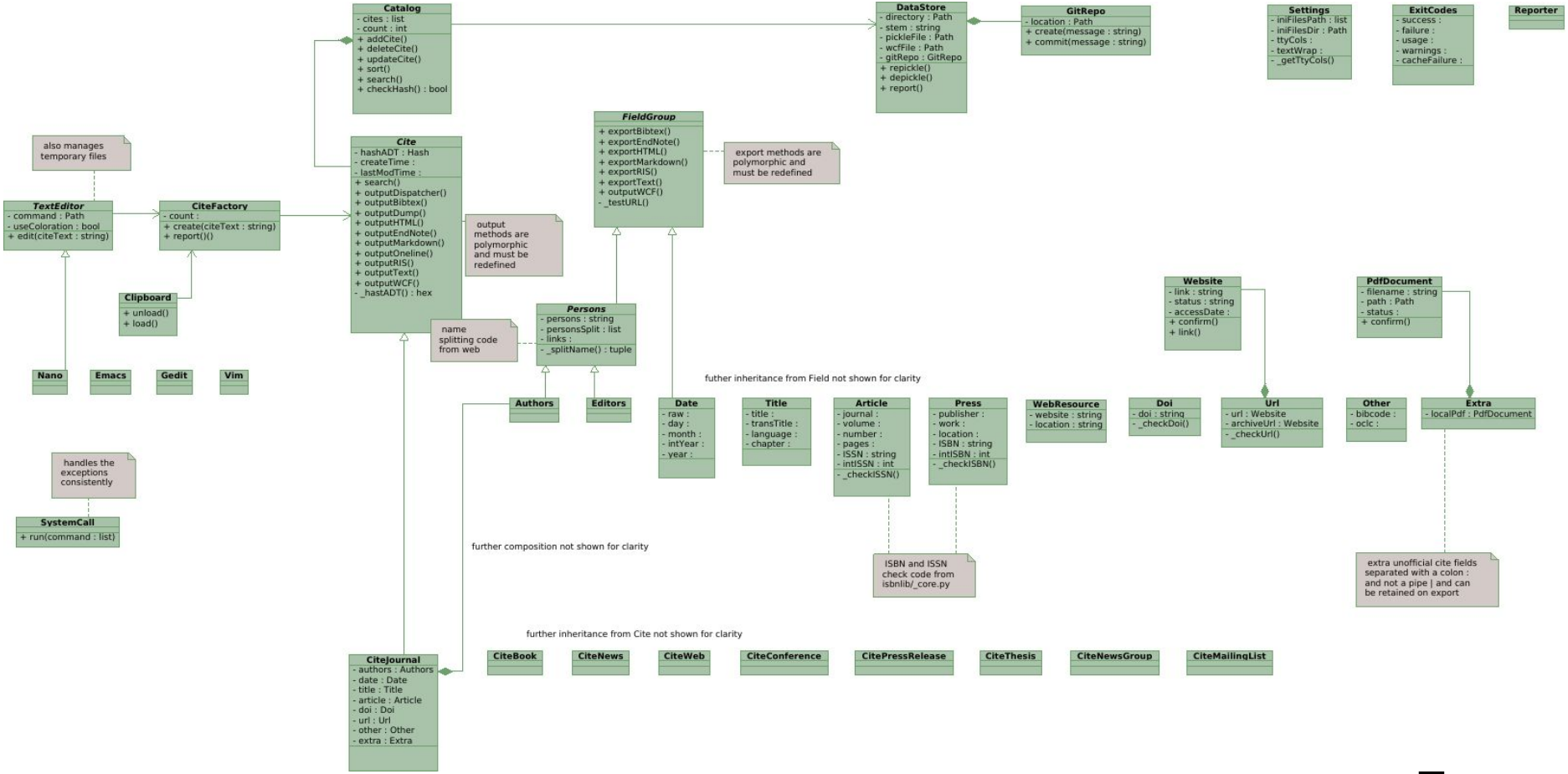
Klasser

- Klasser ritas som tredelade rutor .
- Översta rutan innehåller klassnamnet.
- Därefeter följer egenskaper (fält).
- Understa rutan innehåller beteenden (metoder).



Klassen Calculator





Ett verkligt exempel.

LINKÖPINGS UNIVERSITET

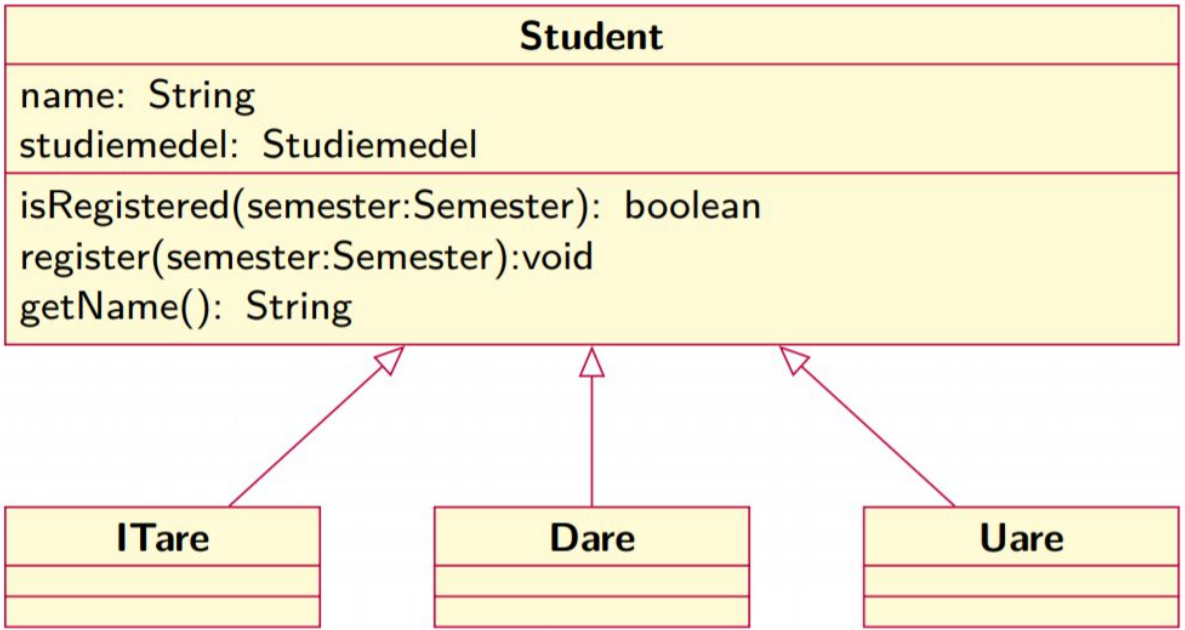
Arv(Inheritance)

Arv (inheritance)

- En klass, subklass, kan ärva egenskaper och beteenden från en annan klass, superklass.
- Ett sätt att dela på gemensam kod.
- En subklass utökar eller specialiserar sin superklass.
- En superklass kan ha flera direkt subklasser.
- En subklass kan bara ha en direkt superklass(single inheritance).
- En subklass `is-a` en superklass.
- Nyckelordet `extends` används för att ärva från en klass.

Arv (inheritance)

- Anta att klass B ärver från klass A. Detta illustreras med hjälp av en icke ifylld “pil” som ritas från B till A



Arv(Inheritance)

- Är en relation

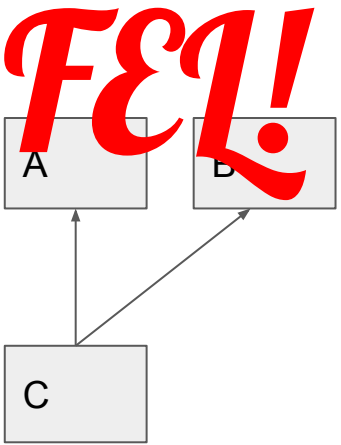
```
class Student{
    private int points;
    private Studiemedel studiemedel;
}

class ITare extends Student {
    private int pointsDatavetenskap;
}
```

*) Privata fält(variabler) i superklassen ärvs inte.

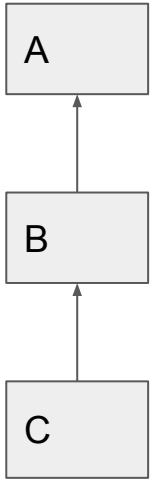
Arv

- I Java kan en klass bara ärva från **en** klass **direkt**.
- I Java kan en klass ärva från flera andra klasser **indirekt**.
- Arv är transitivit.

FEL!

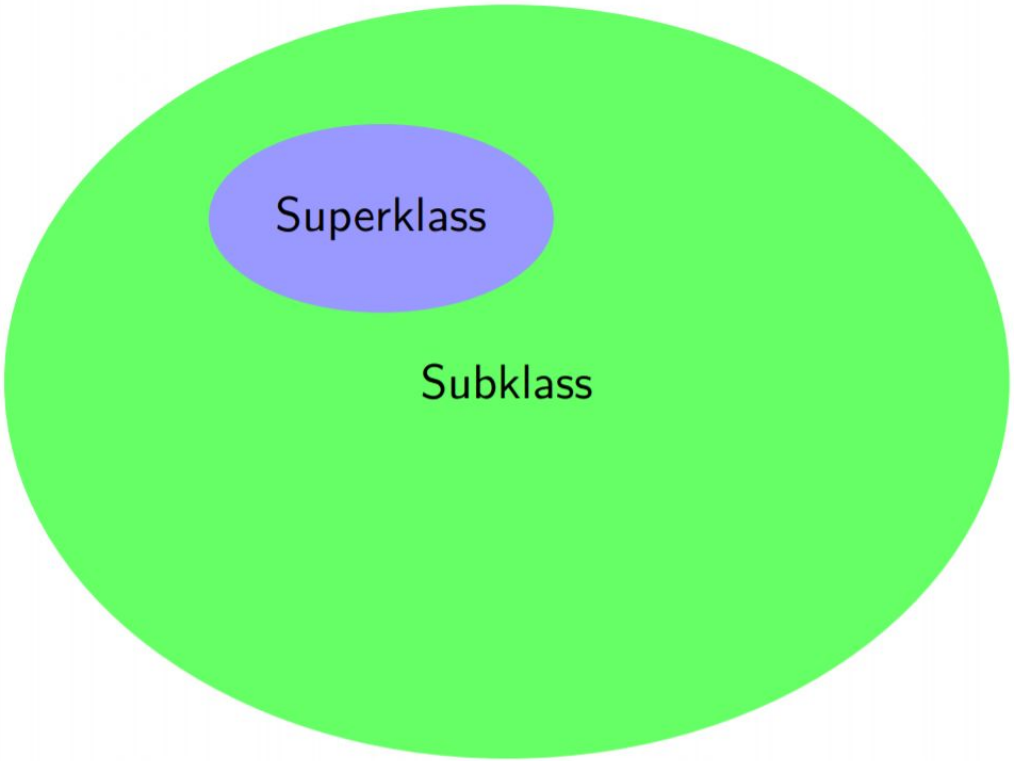


RÄTT!



Hierarki

Arv(Inheritance)



Protected

- Inte alla medlemmar ärvs av subklassen.
- `static` medlemmar ärvs inte.
- `private` medlemmar ärvs inte.
- `public` medlemmar ärvs hela tiden.
- `default` medlemmar ärvs om subklassen ligger i samma paket, mapp, som superklassen.
- **Den enda skillnaden mellan `protected` och `private` är att `protected` medlemmar ärvs hela tiden. Getters och Setters ska givetvis behövas för protected medlemmar.**

this

- `this` är ett nyckelord.
- `this` innehåller en referens till nuvarande objektet.

```
public class Employee {

    private int personalNumber;

    public Employee(int personalNumberValue) {
        this.personalNumber = personalNumberValue;
    }
}
```

this

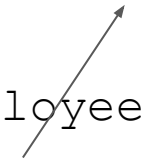
- Om `this` inte används då första metoden/variabeln som matchar namnet ska användas.

```
public class Employee {

    private int personalNumber;

    public Employee(int personalNumberValue) {
        personalNumber = personalNumberValue;
    }
}
```

this

- Ibland måste man använda this referensen för att lösa skuggnings problem.

```
public class Employee {

    private int personalNumber;

    public Employee(int personalNumber) {
        this.personalNumber = personalNumber;
    }
}
```

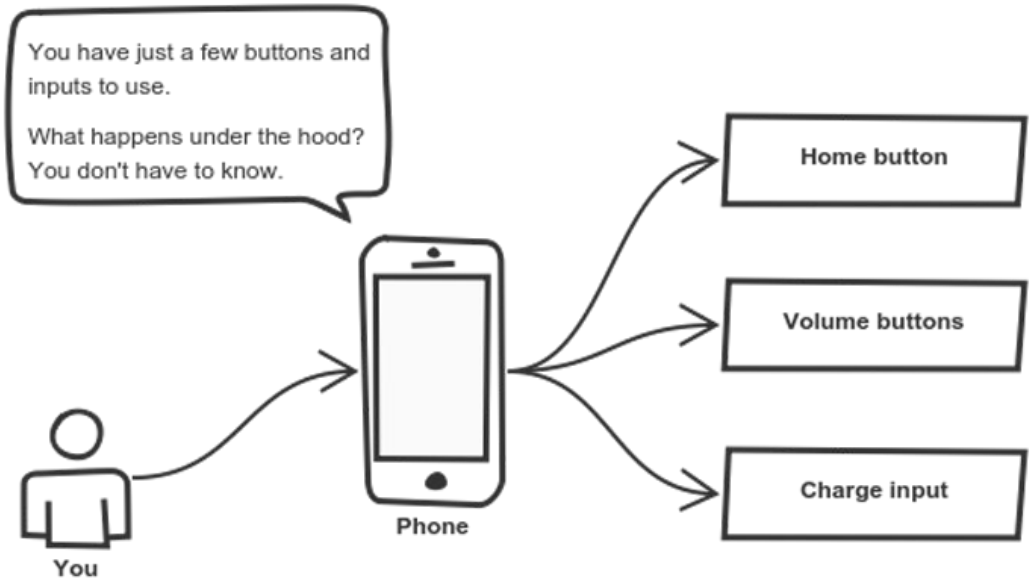


Komposition(Composition)

Komposition

- Ett objekt kan bestå av ett eller flera andra objekt.
- Till exempel, en telefon består av en skärm, ett antal knappar, sensorer och …
- Komposition kallas också för “`Has-a`” förhållande.

Komposition



MobilePhone utan komposition

```
class MobilePhone{
      private int volume;
      private int chargeLevel;
      public MobilePhone(int volume){
            this.volume = volume;
      }
      public void home(){
            showHomeScreen(); // a private method
      }
      Public void volumeUp(){
            volume++;
      }
      public void volumeDown(){
            volume--;
      }
      private void startCharging(){
            readElctricity(); // a private method
      }
}
```

MobilePhone med komposition

```
class MobilePhone{
	private ChargingUnit chargingUnit;
	private HomeScreenButton homeScreenButton;
	private VolumeUnit vUnit;
	private ProximitySensor proximitySensorFront;

	public MobilePhone(){
		homeScreenButton = new HomeScreenButton();
		vUnit = new VolumeUnit();
		proximitySensorFront = new ProximitySensor();
		chargingUnit = new ChargingUnit();
	}

	public void home(){
		homeScreenButton. showHomeScreen();;
	}
	public void volumeUp(){
		vUnit.up();

	}
	public void volumeDown(){
		vUnit.down();
	}
	private void startCharging(){
	      chargingUnit.readElctricity();
	}


}
```

```
class CharigngUnit{
	public void readElectricity(){
	}
}

Class HomeScreenButton{
	public void showHomeScreen(){
	}
}

Class VolumeUnit{
	public void up(){
	}
	public void down(){
	}

}

Class ProximitySensor{

}
```

LINKÖPINGS UNIVERSITET

Bra läsning!

https://www.baeldung.com/java-inheritance-composition

Enumerations

Enumerations

```
enum Seasons{
    winter ("vinter"), spring ("vår"), summer ("sommar"), autumn ("höst");

    private String name;

    Seasons(String name){
     this.name = name;
    }

    public String toString() {
     return name;
    }
}
```

```
import java.util.Scanner;

class SeasonsDriver {

    public static void main(String[] args) {

     Seasons[] values = {Seasons.winter, Seasons.spring, Seasons.summer, Seasons.autumn};

     System.out.println("Choose among: ");
     for(int i=0; i < values.length; i++)
         System.out.println(i + ". " + values[i]);

     Scanner scan = new Scanner(System.in);
     int i = scan.nextInt();
     System.out.println("You chose: " + values[i]);
    }
}
```

Enumerations(Uppräkningar)

- Enumerations kan inte ärva från andra enumerations eller klasser.
- Det är inte tillåtet att ärva från enumerations.
- Enumerarions kan implementera interfaces.

Thanks for listening!

LINKÖPINGS UNIVERSITET