Arv og subklasser - Del II

IN1010 - Uke 4

Tobias Paulsen

Vår 2021

Oblig 1

- Mye bra
- Exceptions, best å spesifisere hvilket unntak, og lurt å håndtere så tidlig som mulig. Prøv å unngå `throws Exception`, det gjør feilsøking vanskelig
- Prøv å spre funksjonalitet utover, gir mer lettleselig, oversiktlig og fleksibel kode

Arv

- Oblig 2 er nå ute, og handler om arv og subklasser. Dere skal utvikle et system over 3 innleveringer
- Spesialisering og generalisering
- Må være en grunn til at man oppretter en subklasse
- Polymorfi

Nøkkelord - Repetisjon

```
class Hund extends Husdyr{...}
```

- extends: for å lage en subklasse

```
protected int alder;
```

- protected: alle av klassens subklasser kan se egenskapen

```
abstract class Dyr {...}
```

- abstract: hvis en klasse er abstrakt kan man ikke opprette en instans av denne klassen, men subklassene arver egenskaper

```
public abstract void spis();
```

- i en abstrakt klasse så kan man også ha abstrakte metoder som man ikke trenger å implementere, subklassene må da implementere metoden

```
h instanceof Hund
```

- instanceof brukes for å sjekke om et objekt er en instanse av en klasse

Super

- **Superklassen** er den klassen som en **subklasse** arver fra `super(farge);`
- I konstruktøren til en subklasse brukes super for å sende verdier opp til superklassen
- Et kall på super må legges først i konstruktøren
- Hvis man ikke kaller på super så legger java inn et tomt kall på super når programmet kompileres: `super();`
- Hvis en klasse ikke har konstruktør så legger også java inn et tomt kall på super.
- Man må derfor ha med super hvis konstruktøren i superklassen tar inn noen verdier

Polymorfi

- Spesialisering
- Ulike subklasser kan ha den samme metode-signaturen, men forskjellige implementasjon
- Ofte ønsker man å omskrive en metode fra superklasssen i subklassen. For å være sikker på at dette skjer så bruker man: `@Override`
- For å hindre omdefinering kan man bruke: `final`
- Hvis man ønsker å bruke superklassen sin implementasjon kan man skrive: `super.metodenavn();`

Fra forrige time(+ @Override):

```
  // I Rektangel.java:
@Override
public double areal() {
    return lengde * bredde;
}
// I Sirkel.java:
@Override
public double areal() {
    return 3.14 * radius * radius;
}
```

Object

- Object er alle klassers superklasse
- To av metodene som arves derfra er: `toString()` og `equals()`
- `toString()` returnerer en String og kalles automatisk om man printer et objekt
- `equals()` returnerer en boolean og spesifiseres hva som skal sammenlignes

```
// For å sjekke to objekter av String er like:
String a = new String("Hei");
String b = new String("Hei");
System.out.println(a.equals(b));  // true
System.out.println(a == b);       // false
```

Overrloading

- Metoder kan ha samme navn men forskjellig *signatur*

```
public void settDimensjoner(int hoyde) {...}
public void settDimensjoner(int hoyde, int bredde) {...}
```

- Gjør at en klasse også kan ha flere konstruktører

```
// Oblig 1:
public Dataklynge(int noderPerRack) {...}

public Dataklynge(String filnavn) {...}
```