Repetisjon: Arv, polymorfi og interface

IN1010 vår 2023

Mentimeter

Dagens plan

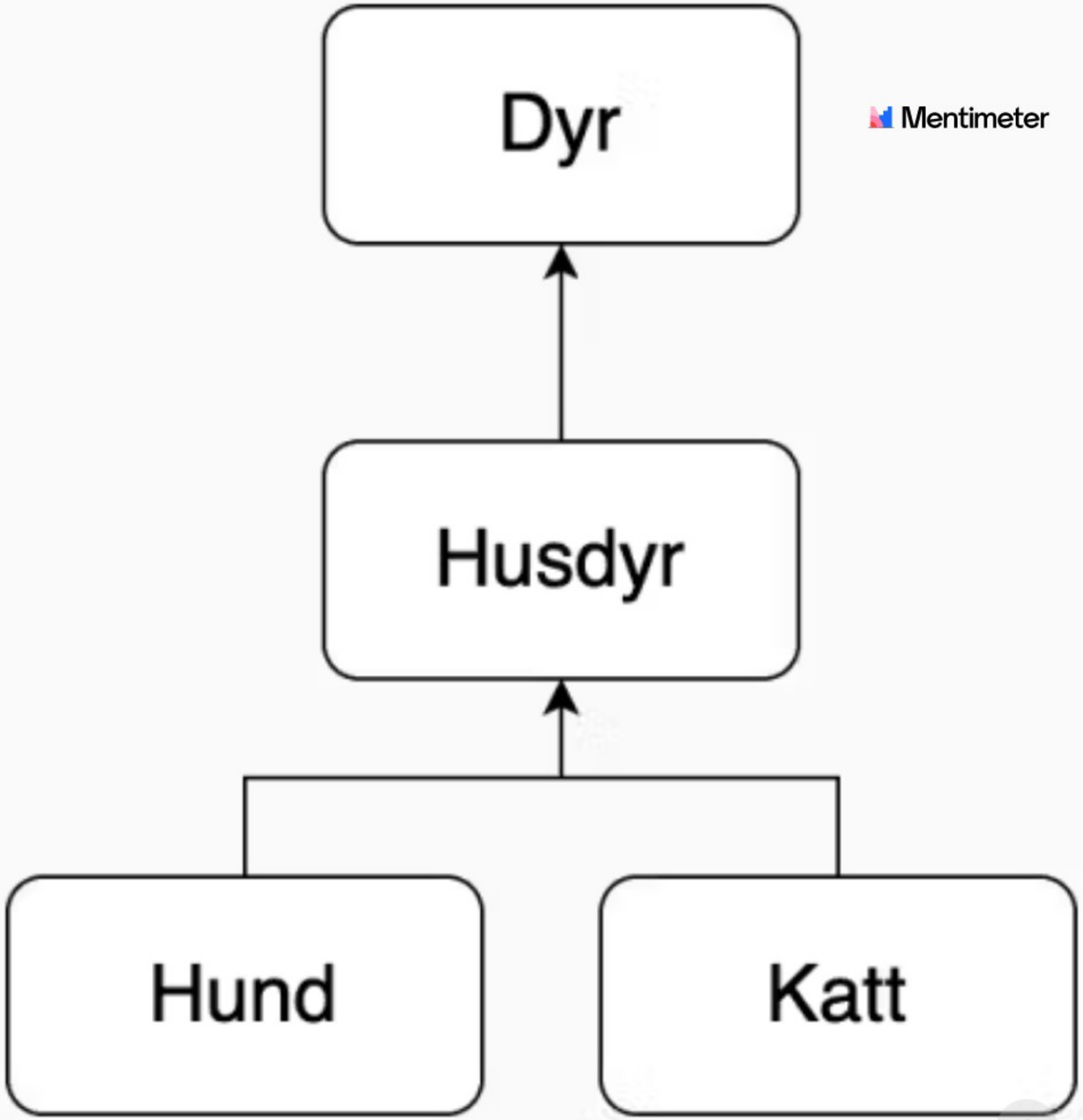
- → Klassisk gjennomgang
- → Menti quiz
- → Gjennomgang av en oppgave

Arv

- Klasser i objektorientert programmering representerer noe som deler et sett med egenskaper
- En subklasse har egenskapene til en klasse + noen nye mer sesialiserte egenskaper
- Subklassen arver egenskapene til superklassen
- Hvorfor subklasser?
  - Ønsker å modellere virkeligheten
  - Gir struktur til systemet vi lager
  - Gjenbruk

Klassehierarki

- Kan ha flere nivåer med arv
- Alle hunder er både dyr og husdyr
- Alle katter er både dyr og husdyr
  - Ingen hunder er katter
  - Ingen katter er hunder



```
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
- alle subklassene må da implementere metoden

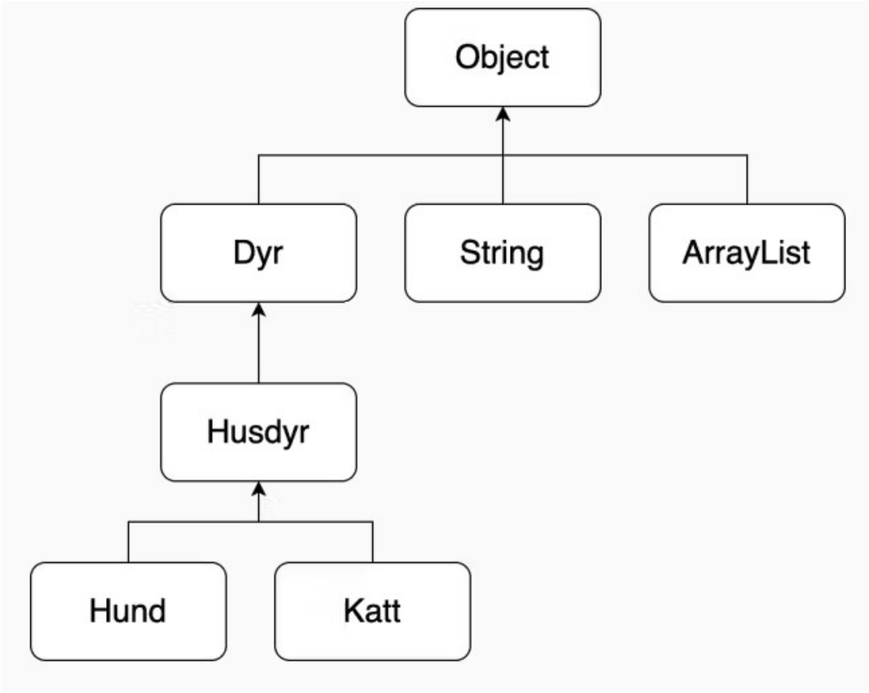
Nøkkelord

Super i konstruktør

- super(variabel); i konstruktøren til en subklasse brukes super for å sende verdier opp til superklassens konstruktør
- 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 (Ikke redefiner superklassens variabler i subklassens konstruktør)

Klassen Object

- Alle klasser arver fra klassen Object (automatisk)
- Metoder som clone(), equals() og toString()
  - Derfor kan man kalle på toString() før den er implementert



Referanser

- Variabel av type A kan referere til objekt av B, men ikke omvendt
- Kan derfor skrive:
  - A var1 = new A();
  - A var2 = new B();
  - B var4 = (B) var2;
- men ikke:
  - B var5 = new A();
  - B var6 = (B) new A();

```
class A {}

class B extends A {}
```

Polymorfi

- Spesialisering i subklasser
- I stedet for å bruke instanceof
- Hvis en metode ikke er final kan metoden overskrives (@Override)
- Ulike subklasser kan da ha den samme metode-signaturen, men forskjellige implementasjon
- Hvis man ønsker å bruke superklassen sin implementasjon kan man skrive:
  - super.metodenavn();

```
  // I klassen Hund
@Override
public void lagLyd() {
    System.out.println("Voff");
}
// I klassen Katt
@Override
public void lagLyd() {
    System.out.println("Mjau");
}
```

Interface

- Interface sikrer at en klasse har implementert noen metoder
- Kun abstrakte metoder, ligner på en abstrakt klasse uten implementasjoner og variabler (Kan ha konstanter)
- I java kan en klasse kun arve fra én annen klasse, men fra flere interfaces
- En klasse deler muligens også egenskaper med andre klasser som ikke har samme superklasse, kan da bruke interface
- Eksempler: List, Comparable, Iterable

Interface

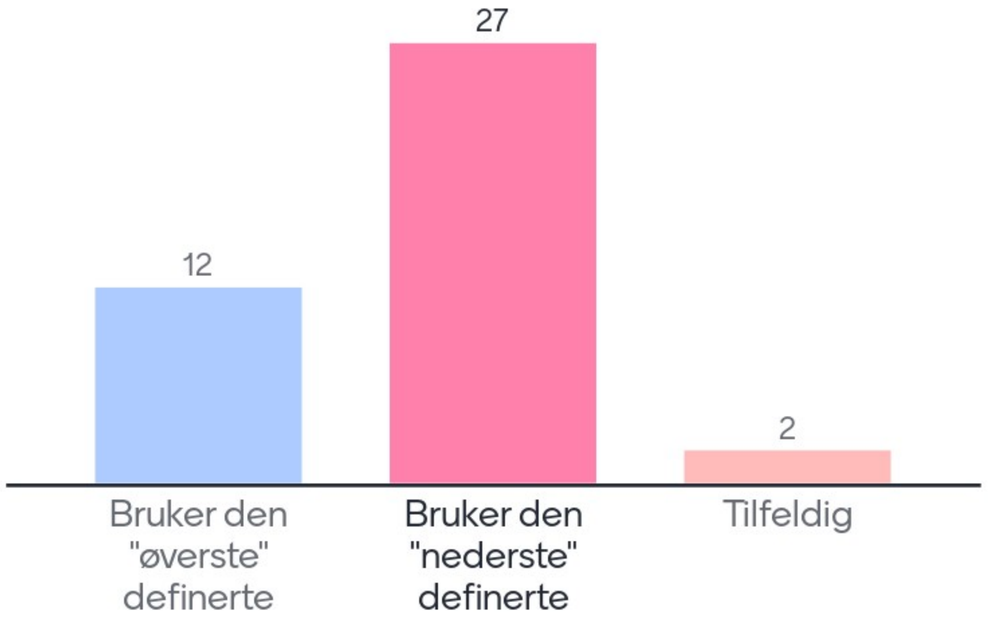
- Kan bruke interface som typen til en peker
- Example var1 = new B();
- Har da kun tilgang til de metodene deklarert i interfacet

```
interface Example {
    public void method1();
}

class A {}
```

```
class B extends A implements Example {

    @Override
    public void method1() {
        System.out.println("Ex.");
    }
}
```

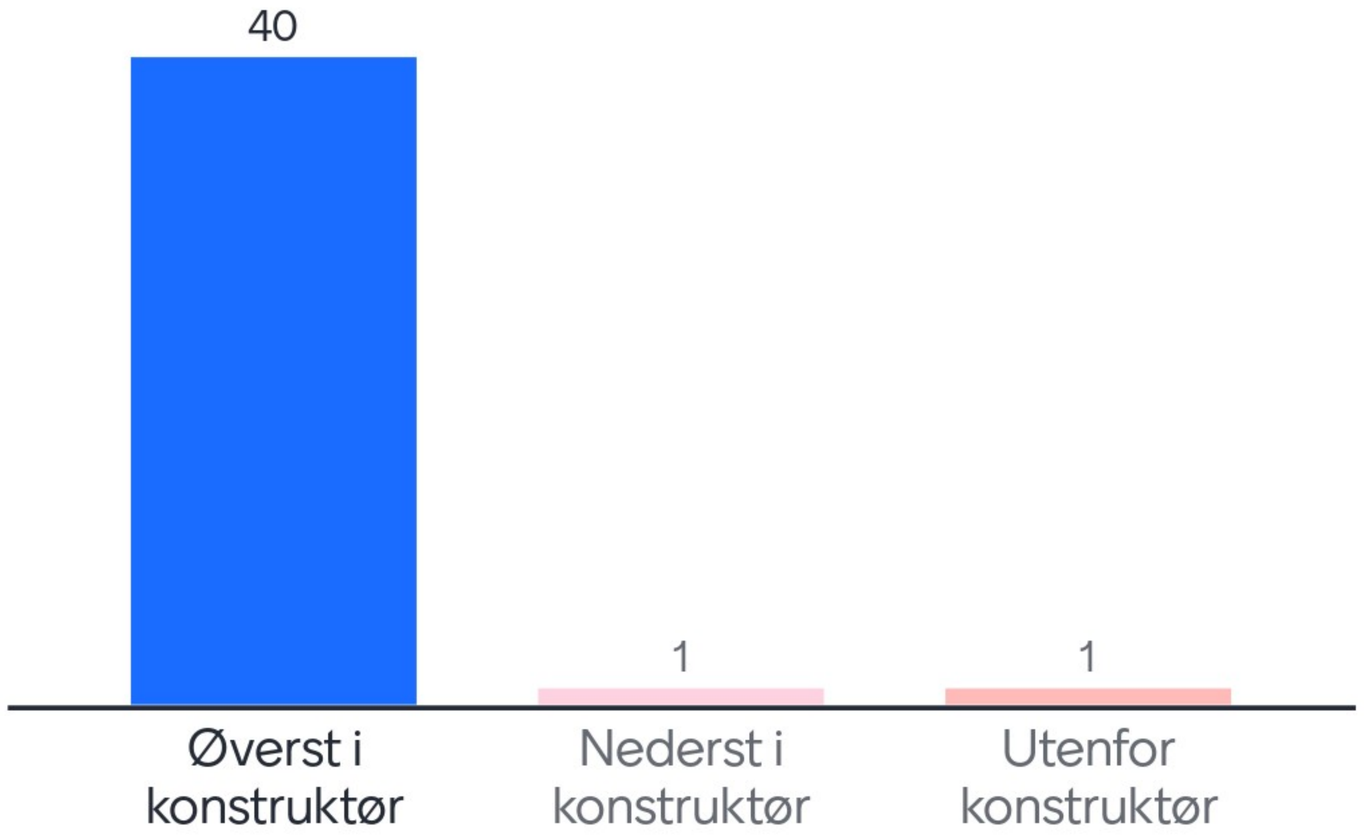
Hvordan systemet vet hvilken metode den skal benytte seg av dersom det er flere polymorfe metoder

Mentimeter



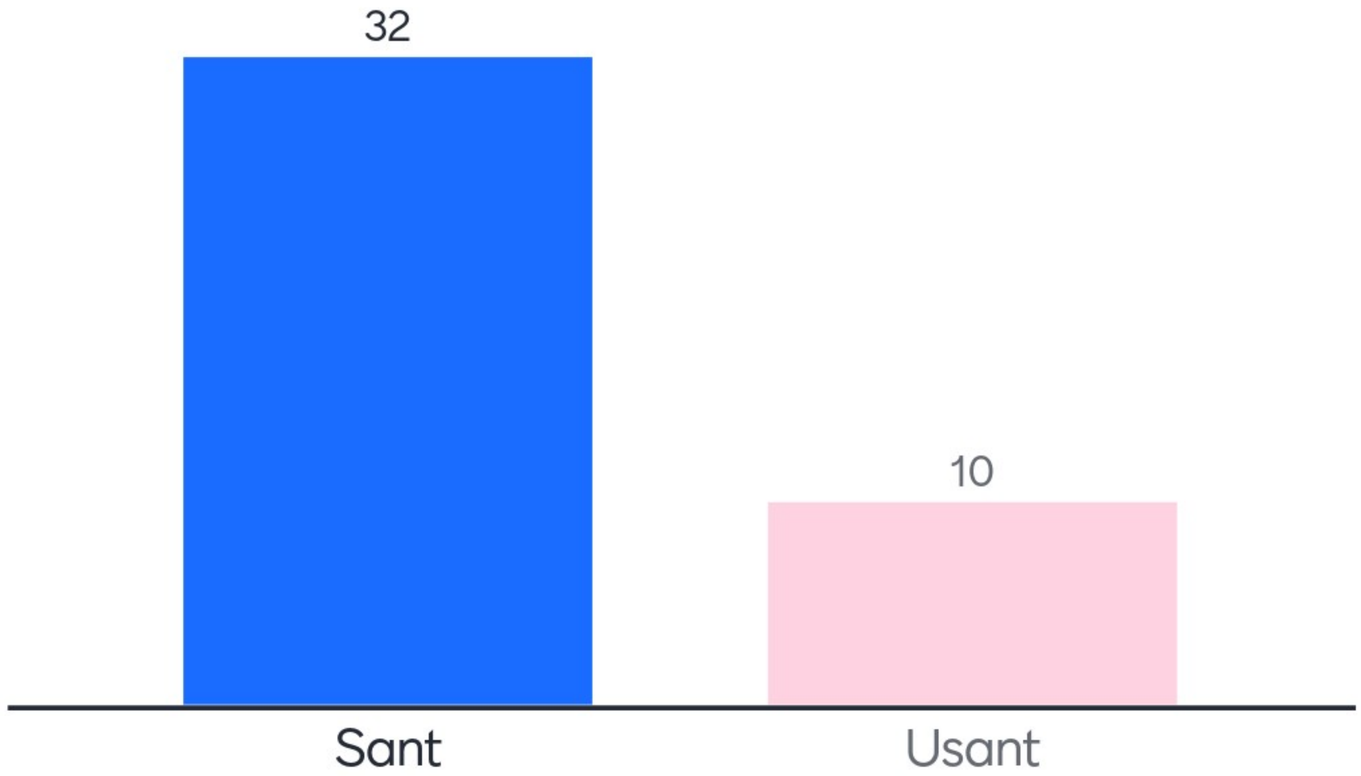
Et kall på super() må legges ...

Mentimeter



"List" er et interface i java

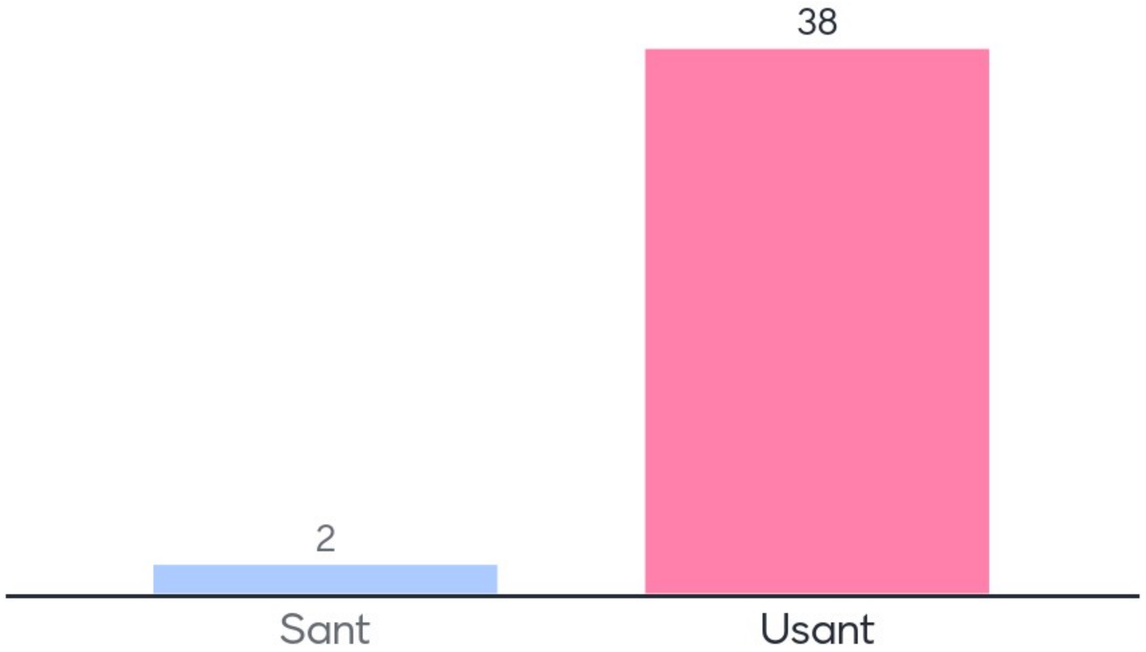
Mentimeter



42

En klasse kan arve fra flere superklasser i Java

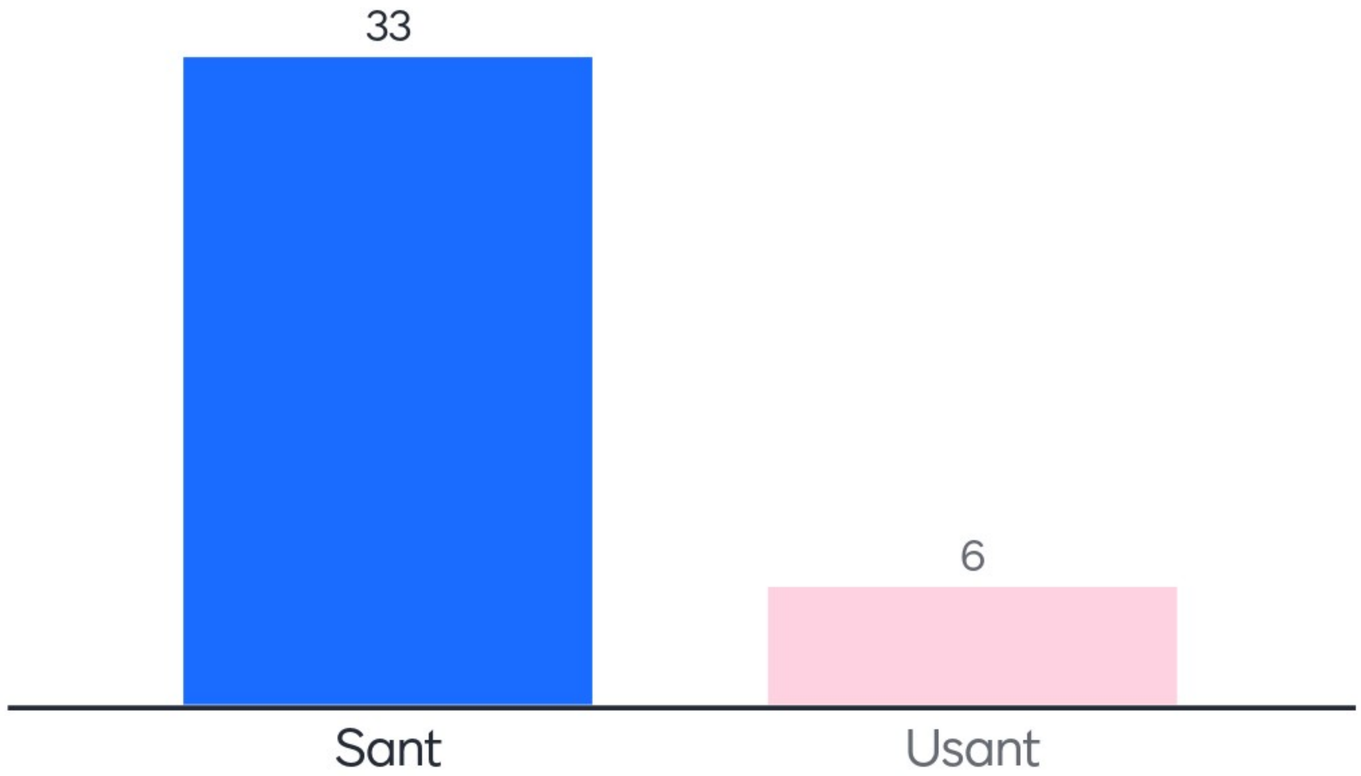
Mentimeter



40

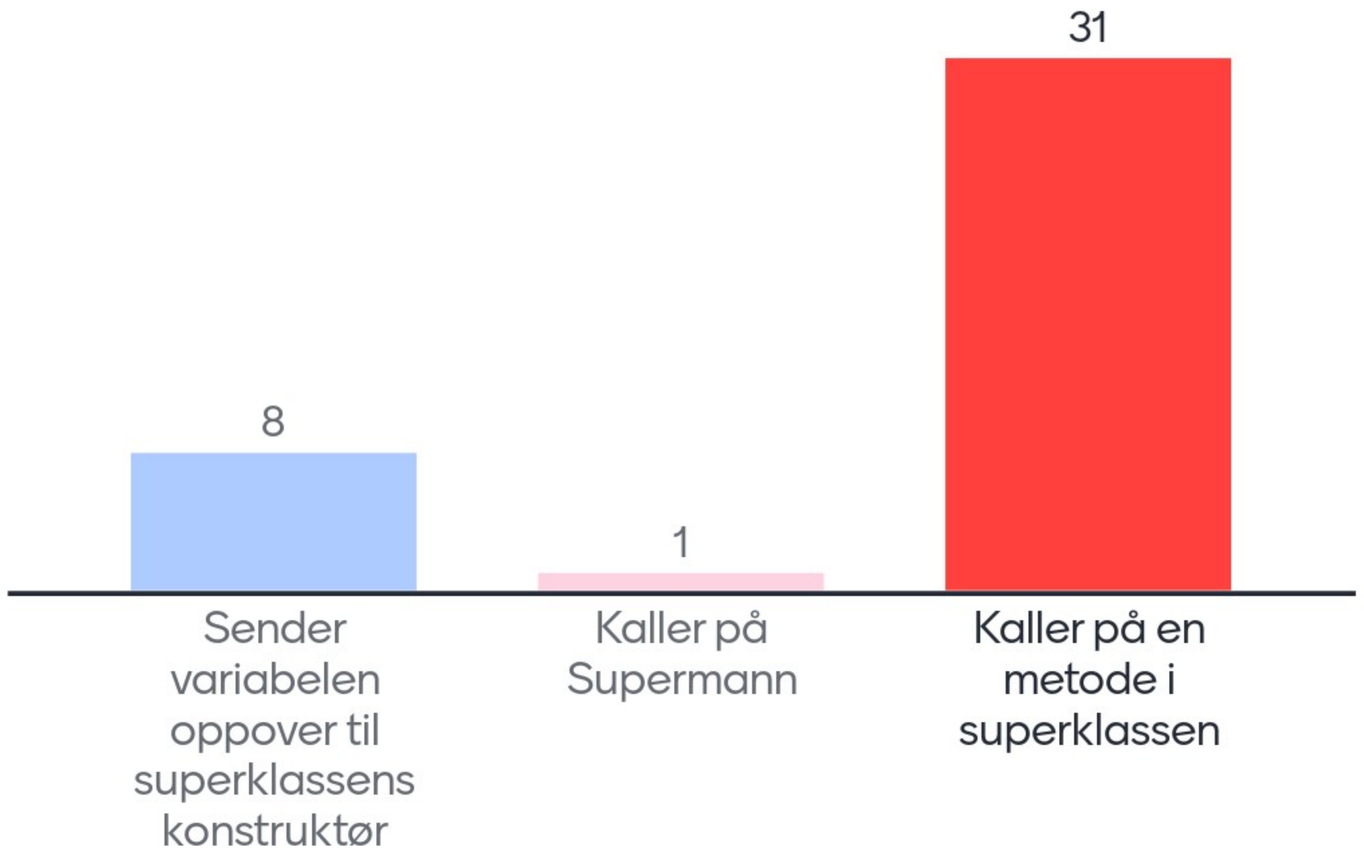
Et interface kan brukes som referansetype

Mentimeter



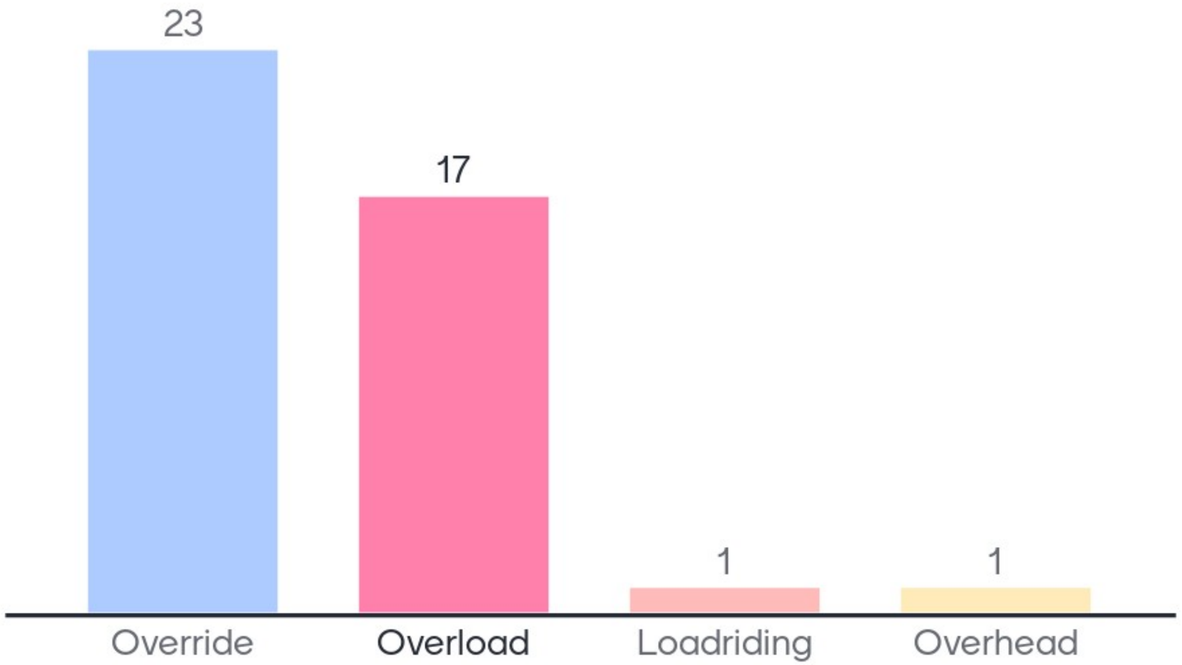
Å skrive "super."

Mentimeter



Å definere metoder med like navn men ulike signaturer kalles

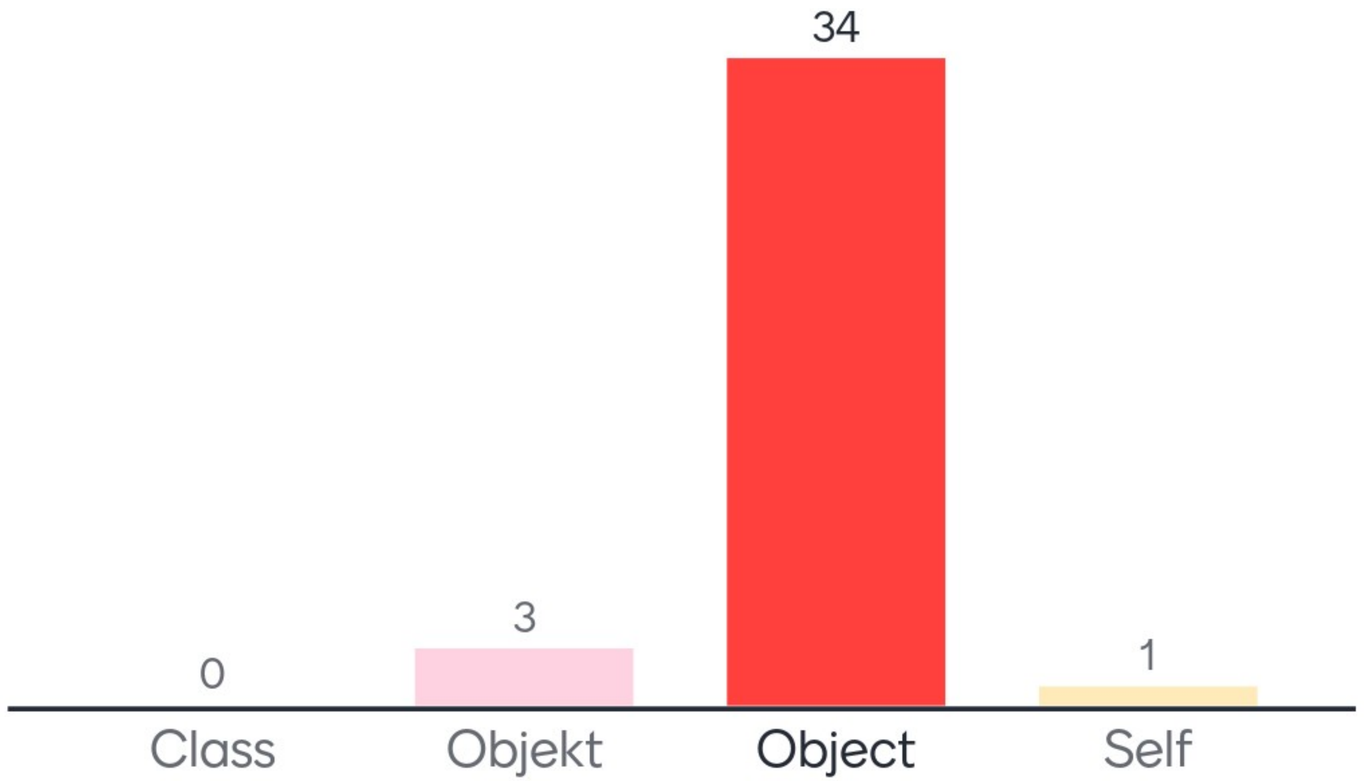
Mentimeter



42

Alle klassers mor

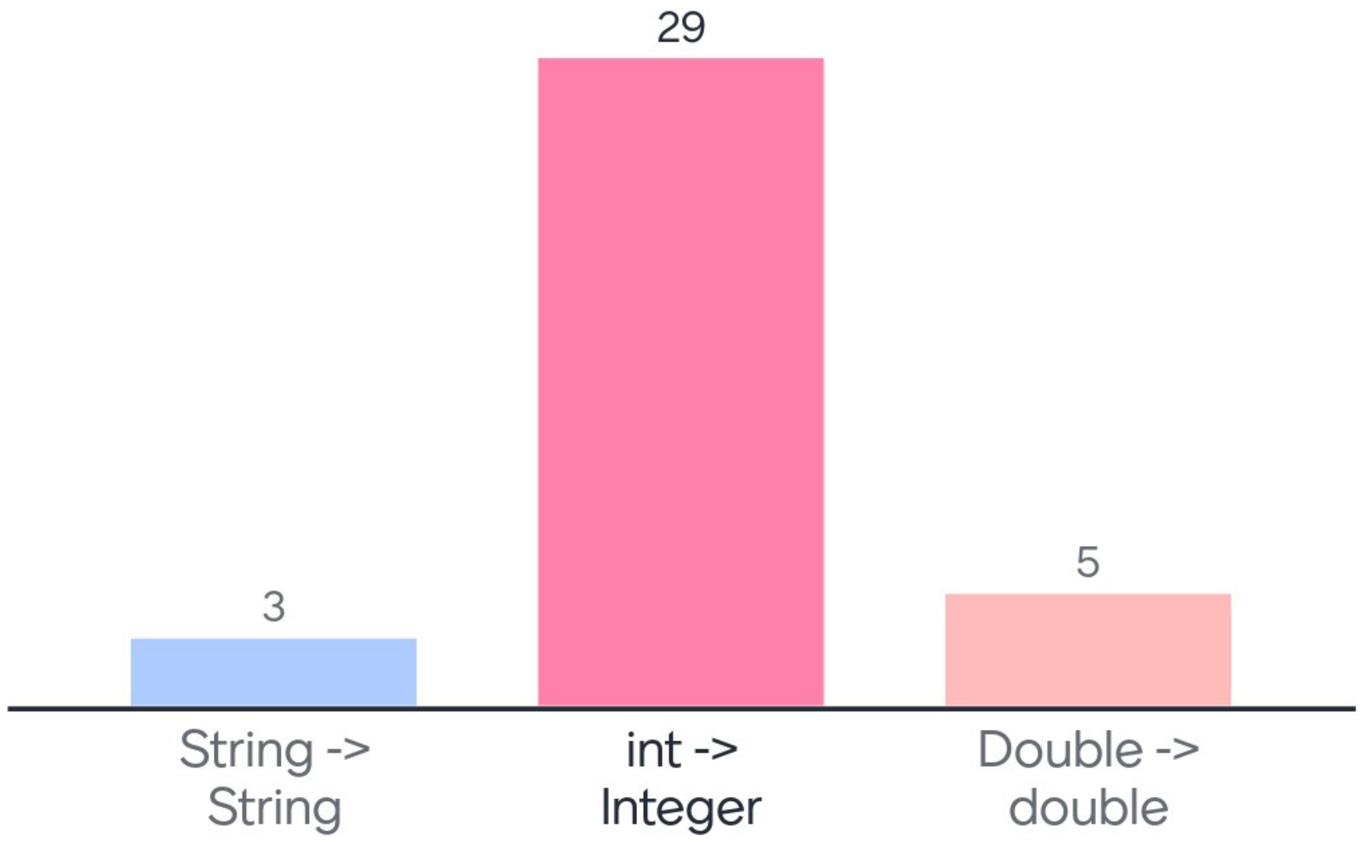
Mentimeter



38

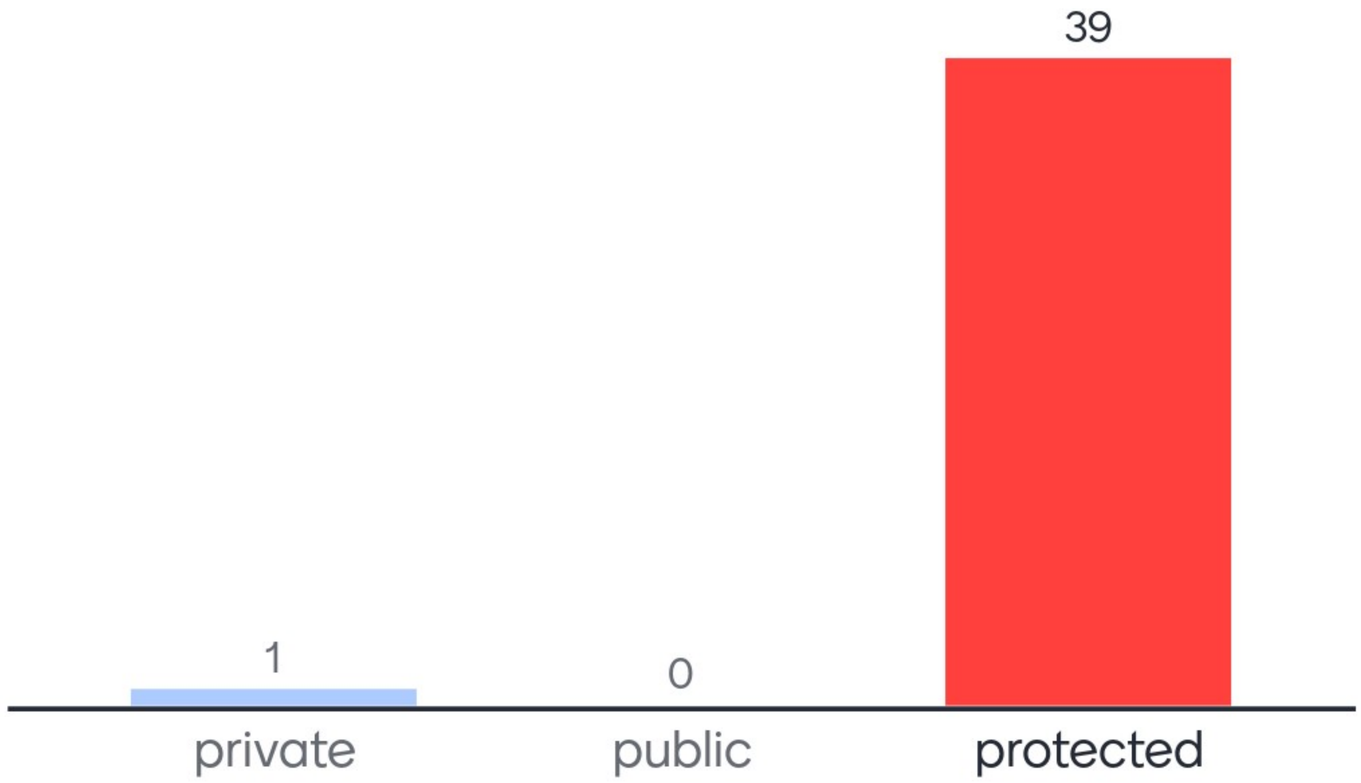
Et eksempel på "boxing"

Mentimeter



Gjør at kun subklasser kan se egenskapen

Mentimeter



Forskjellen mellom en abstrakt klasse og interface

- → Abstrakte klasser kan inneholde fullstendige implementasjoner
- → Grensesnitt: kun signaturer
- → Ingen innmat i grensesnitt (lite unntak)

Abstrakte klasser på eksamen

- Står sjeldent "skriv den abstrakte klassen ..."
- Lese mellom linjene
- Eks. "Alle stier er enten kjerreveier eller naturstier..."
- Nøkkelord som: *enten, kun, det ene eller det andre, skal ikke gå an å lage objekter av ...*

Oppgave

Du har blitt ansatt av en møbelforhandler for å modellere ulike typer møbler. I dette programmet bryr vi oss kun om senger og sofaer. Alle senger er enten Enkelseng, Vannseng eller Dobbeltseng. For Sofa har vi vanlig sofa (klassen skal da hete Sofa), Vannsofa og Hjørnesofa. I og med at både Vannseng og Vannsofa inneholder vann, så skal de også implementere interfacet Vannholdig som har metoden hentVanninnhold. Tegn klasssehierarkiet og implementer klassene beskrevet over.

Mentimeter

Hva nå?

- Jobb med tidligere ukesoppgaver
- Se på trix
- Gjør tidligere eksamensoppgaver