Gruppetime 3

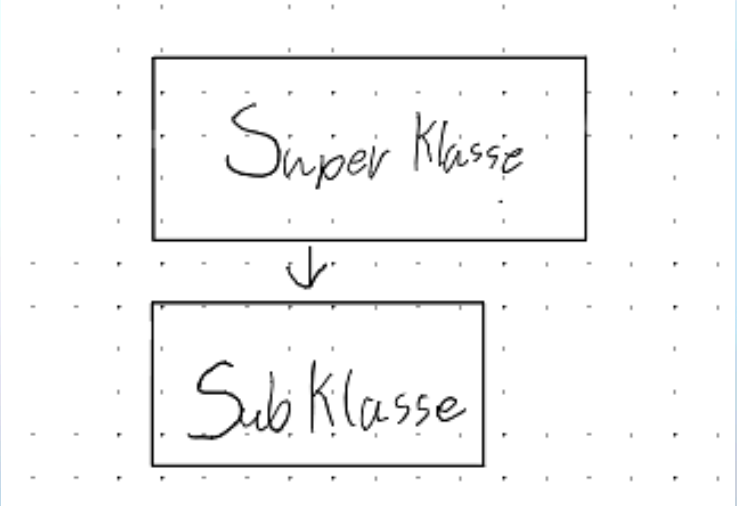
Litt repetisjon

Extends & Arv

```
class SuperKlasse {..}

class SubKlasse extends SuperKlasse {..}
```

- Klasser kan arve fra superklasser
- De får alle instansvariabler og metoder fra superklassen
- Kan skrives som klassehierarkier



Pekere og subklasser

- En peker mot en superklasse kan også peke på alle subklasser (implisitt type casting)
- instanceof sjekker hva klassen *faktisk* er
- Et objekt kan eksplisitt typekonverteres tilbake

```
class A {}
class B extends A {}

class Main {
  public static void main(String[] args) {
    A a = new A();
    B b = new B();
    A c = new B();

    System.out.println(b instanceof A);
    System.out.println(c instanceof A);
}
```

```
class A {}
class B extends A {}

class Main {
  public static void main(String[] args) {
    A a = new A();
    B b = new B();
    A c = new B();

    B d = (B) c; // Dette går fint.
    B e = (B) a; // Dette kræsjer.
  }
}
```

En liten oppgave

Hva printes her?

```java
class A {}
class B extends A {}
class C extends A {}
class D extends C {}

class TrueOrFalse {
    public static void main(String[] args) {
        A a = new A();
        B b = new B();
        C c = new C();
        D d = new D();
        A e = new D();
        A f = new B();
        System.out.println("a) " + (b instanceof A));
        System.out.println("b) " + (c instanceof D));
        System.out.println("c) " + (d instanceof A));
        System.out.println("d) " + (e instanceof C));
        System.out.println("e) " + (f instanceof C));
    }
}
```

Hvilket tilfelle går fint?

```java
class A {}
class B extends A {}
class C extends A {}
class D extends C {}

class TrueOrFalse {
    public static void main(String[] args) {
        A a = new A();
        B b = new B();
        C c = new C();
        D d = new D();
        A e = new D();
        A f = new B();

        //Tilfelle 1
        B ab = (B) a;

        //Tilfelle 2
        if( e instanceof B ){
            B eb = (B) e;
        }

        //Tilfelle 3
        D ed = (D) e;

        //Tilfelle 4
        if( e instanceof D ){
            D ed = (D) e;
        }

        //Tilfelle 5
        B fb = (B) f;

        //Tilfelle 6
        C ec = (C) e;
```

Polymorfi

Metoder med samme signatur har ulik oppførsel

- Man kan overskrive metoder i subklasser med @Override
- Når man kaller en overskrevet metode, blir alltid den *laveste* implementasjonen kalt
- Man kan kalle på supermetoden med *super.metodekall();*
- Vanlige metoder å overskrive er *toString()* og *equals()*

- Man har også *Overloading* å lage en metode med samme navn og type, men med annet antall parametere.

Parkeringshus med motorsykler og biler

Parprogrammering

Dere skal nå lage et program som visualiserer et parkeringshus. Et parkeringshus har parkeringsplasser hvor både Motorsykler og Biler kan parkere. Motorsykler og Biler har en del til felles, bl.a. har de begge en sjåfør og et registreringsnummer. Mens en bil også skal kunne holde på alt fra 2 til 5 personer (spesifisert i det bilen lages), derav en av de må være sjåføren.

- Tegn klassehierarkiet

- Lag alle klassene med instansvariabler. Inntil videre skal de ikke ha noen metoder. Det kan være lurt at Motorsykkel og Bil arver av samme klasse - hvorfor tror du det?

- Parkeringshus skal ha to metoder: en for å sette inn en bil eller motorsykkel og en for å fjerne en bil eller motorsykkel på en gitt plass. Vi vil at metoden skal returnere bilen/motorsykkelen som står på plassen, og returnere null hvis det hverken finnes bil/motorsykkel på plassen. Når dere skal sette inn en bil eller motorsykkel, skal det gjøres på en ønsket plass. Lag unntakshåndtering til parkeringshuset ditt, for å håndtere tilfeller der man prøver å sette inn når det er fullt, eller på en ugyldig plass. Hint: ArrayIndexOutOfBoundsException.

- Lag en metode skrivInformasjon(), som skriver ut all informasjon om hvert av kjøretøyene. Husk at motorsykler og biler har noe felles informasjon, siden begge er kjøretøy, men de har også noe særegen tilleggsinformasjon. Når du itererer gjennom kjøretøyene, skal all informasjon om hvert enkelt kjøretøy skrives til terminalen. Du kan ikke bruke instanceof i denne oppgaven, og du skal unngå å repetere kode unødig.