# ITF10611 OOP

# Oppgave 9

*Marius Mikelsen 26.01.2018*

**Class**

En klasse i objektorientert programmering er en oppskrift på hvordan et objekt skal være strukturert. Dette vil da si at dersom man skaper et objekt av en spesifikk klasse, så får den tildelt et standard sett med variabler, samt metodene som er spesifikt laget for den klassen.

**Object**

Et objekt er en samling av variabler og metoder som blir definert av klassen den tilhører. Alle objekter stammer fra en klasse, og dersom det ikke er laget en egen klasse, så benytter den en standard klasse som heter Object. Fra denne klassen arver alle objekter enkelte egenskaper og metoder, som eksempelvis toString(), som er en representasjon av objektet i form av en String.

**Member Variables**

Membervariabler, som også kalles Instansevariabler og Field, er variabler som er definert i en klasse, og ikke i main(). Motsetningen til membervariabler er lokale variabler, som da er variablene definert i main().

**Overloading**

Overloading tillater at man har flere metoder eller konstruktører med samme navn. I praksis vil dette si at man eksempelvis kan lage en konstruktør som er beregnet for ingen inputparametere, samtidig som man lager samme konstruktør, men der man inkluderer inputparametere. Når du da kaller konstruktøren vil Java (i dette tilfellet) velge den konstruktøren som passer med inputparameterne gitt.

**Overriding**

Når man lager et objekt av en tilfeldig subklasse, så får man automatisk med metoder og egenskaper fra superklassen Object. Dersom man ønsker å endre på noen av disse metodene (eksempelvis toString()), så kan man dette i subklassen. Dette kalles overriding, og uttrykkes med @Override før man lager en metode med samme navn som metoden du ønsker å endre.

**Extends**

Extends brukes da man lager en klasse som skal arve egenskaper fra en annen klasse. Dersom man eksempelvis har en Bil-klasse, ønsker man kanskje en egen klasse til hvert av bilmerkene. Når man da oppretter klassen kan man bruke extends til å indikere hvilken klasse den skal arve egenskapene til.

**Polymorphism**

Polymorphism betyr at noe har flere forskjellige former. Dette vil da si at dersom du har et objekt av klassen Tiger, og denne klassen er en subklasse av superklassen AlleDyr, så vil altså objektet være både Tiger og AlleDyr.

**Private, Public, Protected**

Disse tre tingene er eksempler på tilgangskontroll i Java. Dersom en klasse har en private variabel, så vil objektene av denne klassen kun få tilgang til denne variabelen igjennom metoder fra samme klasse, eller koden internt i klassen.

Under er en tabell jeg har laget over aksessen man får med de forskjellige tilgangene.

| Modifier | Klasse | Pakke | Subklasse (Samme pakke) | Subklasse (Annen pakke) | Verden |
|---|---|---|---|---|---|
| public | 1 | 1 | 1 | 1 | 1 |
| protected | 1 | 1 | 1 | 1 | 0 |
| ingen | 1 | 1 | 1 | 0 | 0 |
| private | 1 | 0 | 0 | 0 | 0 |

**This, Super**

Dette er eksempler på keywords i Java, noe man da bruker for å vise hvilket element du sikter til.

*This :* Ved å plassere this foran en variabel eller metode, så forteller man Java at du sikter til en variabel eller metode som tilhører objektet den i det tidspunktet jobber med.

*Super :* Fungerer på samme måte som this, men viser at du sikter til et element i nærmeste superklasse.