# Teorioppgaver

## Oppgave 1.1.

#### Static (variabel, metode)

Static variable og metode tilhører klassen og er delt mellom alle objekter. Om et objekt endrer static variabel vil dette også påvirke de andre objektene i klassen. Vi kan få tilgang til en static variable uten å måtte lage en instance av klassen, dette er fordi den tilhører klassen og ikke objektet. Vi har tilgang til static variabel ved å kalle på klassen.

Static metode tilhører klassen og vi kan få tilgang til den static metode uten å måtte lage en instance av klassen, static metode kjører direkte på klassen.

#### Final (variabel, metode, klasse)

Vi bruker final når vi ønsker å sette noe konstant, der vi ikke kan endre eller modifisere på dette i etterkant. Vi kan bruke final i variabel, metode eller klasse.

* + Når vi bruker det for å sette en variabel så blir dette en konstant variabel der vi ikke kan endre på i etterkant. Det er mulig å sette en final variabel til å være tomt, men da må man initiere disse variablene i en konstruktør.
  + Bruker vi final i en metode så kan vi ikke overskrive (overriding) metoden i en barneklasse.
  + Setter vi en klasse til å være final så kan vi ikke arve dette til en annen klasse.

Om vi prøver å endre variablene, overriding av metode eller prøve å arve fra en final så vil få en kompilerings error. Enkelt sakt så er final at det er «endelig». Du kan ikke endre på variabler, du har ikke mulighet til å endre på metoder eller bruker dette i en barneklasse.

#### Abstract (klasse, metode)

Vi bruker abstract får å abstrahere egenskaper som er felles for objektene vi ønsker å lage, uten at vi må lage objektet. Ved å bruke «abstract» så vil det gi oss noen begrensninger, disse begrensningene er forskjellig mellom klasse og metode. Vi har ikke mulighet for å lage et nytt objekt ved å bruke abstract class, for å kunne gjøre dette så må en barneklasse arve det som er fra abstract klassen. I en abstract klasse så kan vi bruke både abstract metode, men også vanlig metode.

En abstract metode kan kun lages når vi er i en abstract class, uten å ha en kodekropp. For å kunne gjøre dette MÅ/SKAL alle barneklasse HA implementert @Override metoden, og definere en kodekropp.

#### Interface

Interface er en annen måte å lage en «abstract class», slik at vi kan lage metode uten kodekropp. For at vi skal kunne aksessere interface metode, må vi bruke «implements» istedenfor «extends» i en annen klasse. Kodekroppen blir da implementert i «implements» klassen.

Siden Interface på en måte, er «abstract class» har vi heller ikke mulighet til å lage et objekt med interface. Siden den ikke kan lage objekter kan den heller ikke ha en konstruktør i Interface. Når vi «bruker/implementere» Interface i en klasse, så må vi @Override alle metoder i Interface, hvis ikke vil få da få en error.

#### Klassediagram

