**Teori**

**Oppgave 1.1**

Lag deg en oversikt over hva følgende ord/begreper betyr, *med egne ord* (lurt å gjøre dette bra nå, kan være nyttig til eksamen):

* Class

Class er en slags oversikt over hva som skal bli laget. Den er satt opp av instansevariabler, som det det som styrer get-og set metodene som blir laget.

Det som bruker classen til å lage noe, er objektet. Om vi har en class med instansevariabler, kan vi gå i main og lage et objekt for denne classen. Da har objektet brukt oversiktet til classen og kan bruke variablene som er laget.

* Object (konseptet, ikke klassen)

Object er det du lager når du har opprettet en Class.

Ser slik ut classNavn objekt = new classNavn();

Dette blir da laget i main, og du kan hente inn get-og set metoder som du kan bruke.

* Instansvariabel

Instansevariabel er de variablene du tildeler classen du lager. De kan være public eller private.

Eks:

Class variabel{

Private String name;

Public int tall;

}

Dette vil si når du lager et objekt av Class variabel kan du benytte deg av disse variablene. Private må du lage get og set metoder, mens public så kan du bruke de med engang. public 🡪 variabel.int = 8;

Private 🡪 variabel.setName(‘’Tom’’);

* Overloading

Overloading vil si at du lager metoder med samme navn, men med forskjellige argumenter.Altså at metodene skal så å si gjøre det samme, men de blir forskjellige på hvor mange argumenter de tar inn. Dette er for å forenkle programmet slik at du ikke trenger å huske på mange navn.

Eks:

public int add(int a, int b){return a+b};

public int add(int a, int b, int c){return a+b+c};

* Overriding

@Override er en måte å indikere programmet at du vil overskrive en funksjon (get eller set metode, toString) som allerede finnes. Kanskje du har lyst til å endre slik at den skal skrive ut mer informasjon eller liknende.

* Extends

Extends bruker du om du skal arve fra en Class til en annen Class.

Du skriver da Class classNavn extends classNavnDuArverFra.

Dette brukes for å slippe å lage gjentatte variabler som er likt for begge. Det vil si at alle get og set metodene som blir laget i Class som blir arvet fra, kan brukes i Class som arver.

* Polymorphism

Polymorphism vil si at når en Class arver fra en annen Class så kan man lage et objekt som inneholder begge Classene.

Eks:

Parent navn = new barn(); // Parent er super classen, mens barn arver fra Parent.

Det dette gjør er at barn Classen kan bruke metodene som er i parent Classen.

Så om de to Classene har en toString metode, så vil barn sitt bli benyttet.

* private,public,(protected) (klasse,variabel,metode)

Private: Private metoder og instansevariabler kan bare blir brukt i den instansen av klassen du lager. Eksempelvis om du lager et objekt: objekt h = new objekt();. Når du bruker metodene for å legge inn variabler, så vil de variablene bare telle for dette objektet h. Er en metode private, så kan bare klassen som opprettet metoden bruke det.

Public: Når en instansevariabel/metode er public, så hvem som helst gå inn og endre/bruke den.

Protected: Protected metoder kan man bare komme innpå om man er i classen som lagde den, eller om andre classer arver fra den.

Eks: Class parent{kode}, Class barn extends Parents{kode}, Class fremmed{kode}

Koden som er protected i Parent kan bli brukt i barn, men om det blir laget et objekt av Parent i fremmed, så kan ikke metoden blir brukt.

* this og super

Når vi skriver this så refererer det til det som gjelder for dette objektet vi holder på med.

Public void setName(String name){this.name = name};

Objekt h = new objekt();

h.setName(‘’Tom’’);

Da blir String name satt til Tom for dette objektet h.

Super: Blir brukt når du arver fra en annen class. Da blir Classen som arver noe super classen til de som får arven.

Om det er en kostruktør i Super classen, så kan man bruke super(variabler) for å slippe this.

Eks: Class h{

private String h;

public h(String h){

this.h= h;

}

}

class b extends h{

private String k;

public b(String h, String k){

super(h);

this.k =k;

}

}