Oppgave 1.1:

Class

- En klasse en slags «blueprint» for å lage objekter. Den definerer egenskaper (variabler) og adferd (metoder) som objekter av klassen vil ha.

Object (konseptet, ikke klassen)

- Et objekt er en instans av en klasse som inneholder tilstand (variabler) og oppførsel (metoder) Eks på objekt:

```
Planet mercury = new Planet("Mercury", 0.03412549655905556,
1.7297154899894627E-4);
```

Instansvariabel

- En instansvariabel er en variabel som er assosiert med en instans (objekt) av en klasse.
- Den er deklarert i klassen, men utenfor noen metode, vanligvis øverst i klassen.
- Hver instans (objekt) av klassen har sin egen kopi av instansvariabelen.
- Den kan endres på gjennom hele levetiden til objektet.

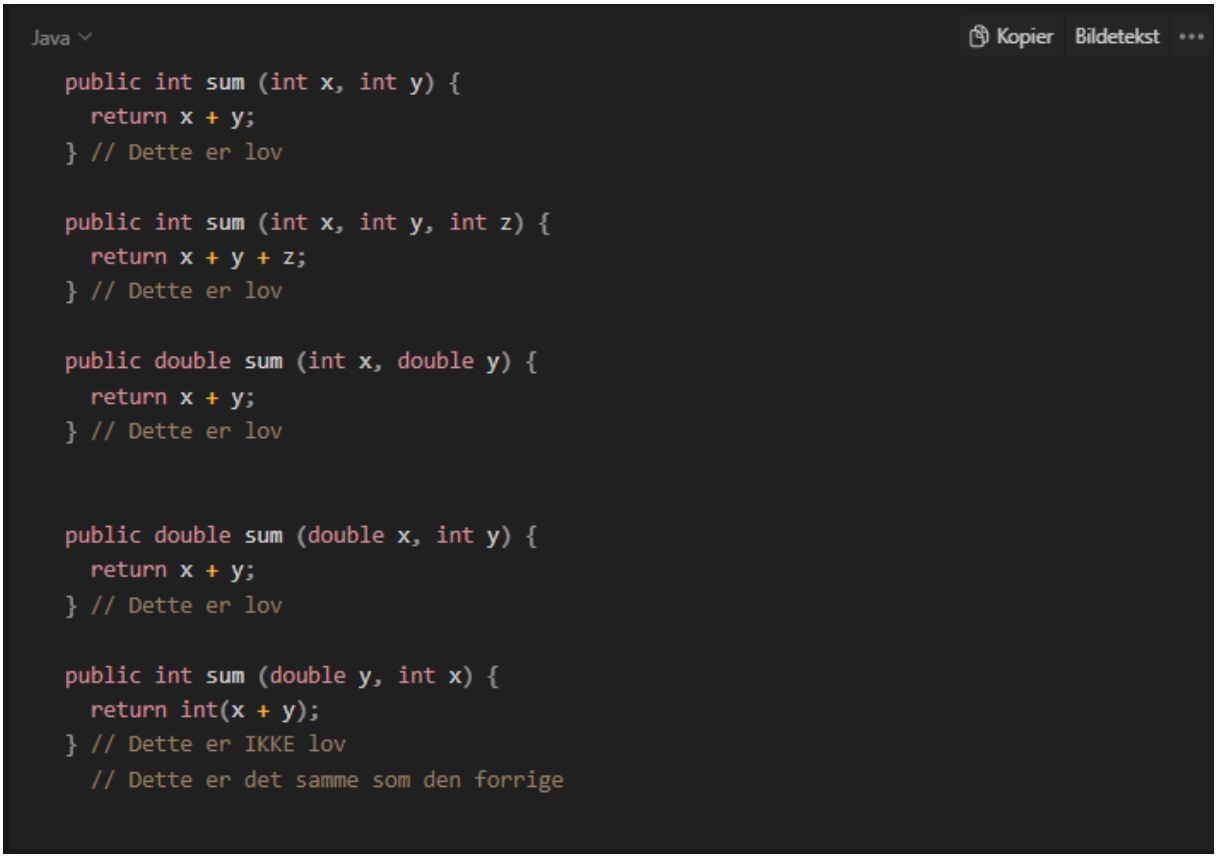
```
public class Person {
    // Instansvariabel
    String navn;

    // Konstruktør
    public Person(String navn) {
        // Tilordner verdien av lokal variabel "navn" til
instansvariabelen "this.navn"
        this.navn = navn;
    }

    // Metode som bruker instansvariabelen
    public void skrivUtNavn() {
        System.out.println("Navn: " + navn);
    }
}
```

Overloading

- Overloading gir oss muligheten til å definere flere metoder med samme navn i en klasse, men med forskjellige parametertyper eller antall parametere. Når du overloader en metode, oppretter du forskjellige versjoner av den samme metoden som kan ta forskjellige typer argumenter. Ved overloading skal metoden ha samme navn, men man må skille basert på parametere (antall og rekkefølge) og ulike returtype.
- Eks:
-



Overriding

- Overriding brukes for å overstyre og omdefinere funksjonalitet for en gitt barneklasse. Metoden toString er ofte relevant i denne sammenhengen. Når man benytter seg av overriding, er det anbefalt å bruke nøkkelordet «@override». Dette forsikrer oss at en arvet metode faktisk blir overridet. Er det noe som ikke blir overridet, får vi error. Override har derimot noen

betingelser. Returtype og navn må være den samme, mens parametere og kodelogikk kan forandres på.

Eks:

```
@Override
public String toString() {
      return "Carpenter " + firstName + " " + lastName + " has built " + housesBuilt + " houses." + " He is " + age + " years old.";
}
```

Extends

- Extends et nøkkelord som brukes for å opprette en arvssammenheng mellom to klasser. Når en klasse utvider en annen klasse ved å bruke extends, arver den alle ikke-private egenskaper og metoder fra den overordnede klassen.

  Eks:

```
public class Carpenter extends Person{
protected int housesBuilt;
```

Polymorphism

- Ikke lært per i dag (09.02.24)

Private og public:

- Private og public er tilgangsnivåene i Java. Private vil si at klassen, metoden eller variabelen kun er tilgjengelig innenfor samme klasse. Er den public er den tilgjengelig fra hvilken som helst klasse. At noe er «protected» vil si at det er tilgjengelig innenfor samme pakke og underklasser.

this og super

- «this» refererer til gjeldende objekt i en metode eller konstruktør. Den brukes for å unngå forveksling mellom instansvariabler og lokale variabler med samme navn.
- «super» bruker man når man ønsker å arve noe fra en overordnet klasse. Dette er nyttig ettersom man kan ta i bruk metodene i foreldreklassen, og man slipper derfor å definere disse på nytt. Nøkkelordet «super» referer i seg selv kun til den klassen som er valgt.

«super» har parametere, hvor parameterverdiene skal matche med de tidligere variabelnavnene, som da er definert i foreldreklassen man vil arve fra. Ved at disse er definert som parametere slipper man å definere de på nytt som variabler.