Hva du må vite om tegning i IN1010

Dag Langmyhr
dag@ifi.uio.no

27. februar 2026

I IN1010 brukes diagrammer for gi bedre oversikt over Java-programmer og hva som skjer når de kjøres. Nærmere bestemt brukes disse to formene for diagrammer:

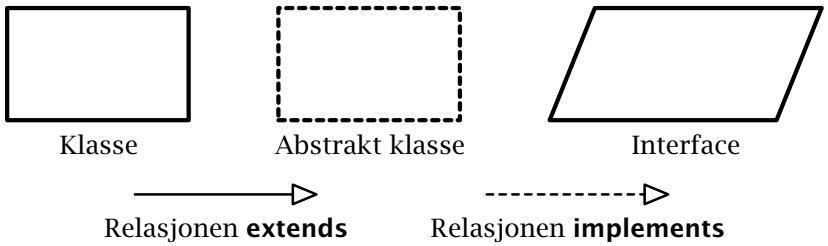
Klassediagrammer viser hvilke klasser og interfacer som forekommer i programmet og dessuten hvilke som er subklasser eller implementasjoner av hverandre. Dette er *statisk* informasjon som ikke endres under kjøringen.

Datastrukturdiagrammer viser et utvalg av objekter og variabler under kjøring for å tydeliggjøre sammenhengen mellom dem. Dette er *dynamisk* informasjon som endrer seg hele tiden under kjøringen, så datastrukturdiagrammet viser et øyeblikksbilde av situasjonen.

Ellers er det verdt å huske på at formålet med diagrammene er å illustrere fagstoff, så hvor mye informasjon som vises, vil variere. Det viktigste er å formidle generell kunnskap.

1 Klassediagrammer

Klassediagrammer viser hvilke klasser og interfacer vi har i programmet vårt og hvordan de er subklasser eller implementasjoner av hverandre. Til dette bruker vi elementene vist i figur 1:



Figur 1: Elementer i klassediagrammer

Klasser vises som rektangler.

Abstrakte klasser (dvs klasser vi ikke kan bruke **new** på) er stiplet.

Interfacer tegnes som skjeve parallellogrammer.

Relasjonen extends angir at en klasse er en subklasse av klassen den peker på, eller at interfacet er et subinterface av interfacet det peker på.

Relasjonen implements angir at en klasse implementerer interfacet den peker på.

Det er ellers verdt å merke seg at klassediagrammene vi tegner normalt ikke inneholder noe om innmaten av klassene eller interfacene (dvs variabler eller metoder).

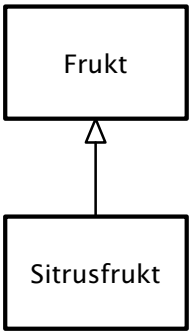
1.1 Eksempler

Anta at vi har en klasse Frukt som har en subklasse Sitrusfrukt:

```
class Frukt {
    /* ... */
}

class Sitrusfrukt extends Frukt {
    /* ... */
}
```

Dette illustrerer vi som vist i figur 2.



Figur 2: En klasse og en subklasse

I et annet eksempel har vi både vanlige klasser, abstrakte klasser og et interface:

```
abstract class Bil {
    /* ... */
}

class Lastebil extends Bil {
    /* ... */
}

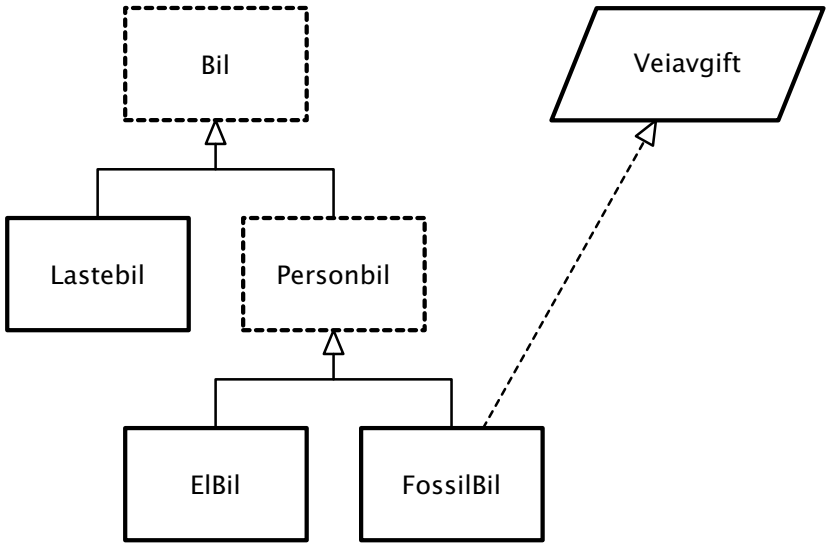
abstract class Personbil extends Bil {
    /* ... */
}

class ElBil extends Personbil {
    /* ... */
}

class FossilBil extends Personbil implements Veiavgift {
    /* ... */
}

interface Veiavgift {
    /* ... */
}
```

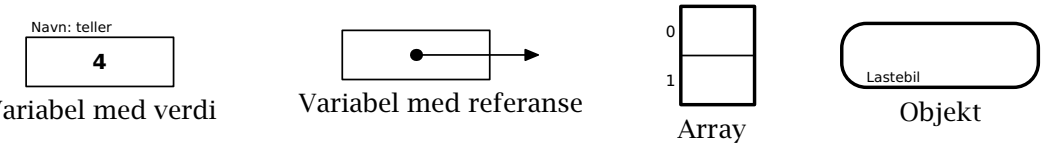
Klassestrukturtegningen for disse klassene ser vi i figur 3.



Figur 3: Klasser, subklasser og et interface

2 Datastrukturdiagrammer

Som nevnt viser datastrukturdiagrammet et øyeblikksbilde fra kjøringen så vi kan se hvilke data (dvs objekter, pekere[1] og variabler) som er interessante, og sammenhengen mellom dem. Spesielt er pekerne og objektene viktige for å forstå datastrukturen. I datastrukturdiagrammene bruker vi disse elementene:



Figur 4: Elementer i datastrukturdiagrammer

Variabler tegnes som rektangler, og de kan inneholde en verdi eller en peker. Noen ganger tar vi med variabelens navn, men bare når vi synes det er viktig.

Arrayer tegnes som en samling rektangler indeksert fra 0 og oppover.

Objekter er opprettet med **new** og de tegnes som et rektangel med avrundede hjørner. Nede til venstre skriver vi gjerne navnet på klassen objektet er laget fra. Objekter kan inneholde nye variabler.

Husk at datastrukturdiagrammer inneholder objekter, men aldri klasser eller interfacer. Vanligvis inneholder de ikke metoder, med mindre det er noe spesielt som skal illustreres.

[1] I IN1010 bruker vi betegnelsene *peker* og *referanse* om hverandre. De betyr det samme.

2.1 Eksempel

Dette Java-programmet har en datastruktur med flere klasser og en array:

```
class Kaniner {
    public static void main (String[] args) {
        Kaninbur bur = new Kaninbur(3);
        bur.settInn(new Kanin("Kalle"));
        bur.settInn(new Kanin("Peter"));
        /* --- Nå tegnes datastrukturdiagrammet! --- */
    }
}

class Kaninbur {
    Kanin[] beboere;
    int antall = 0;

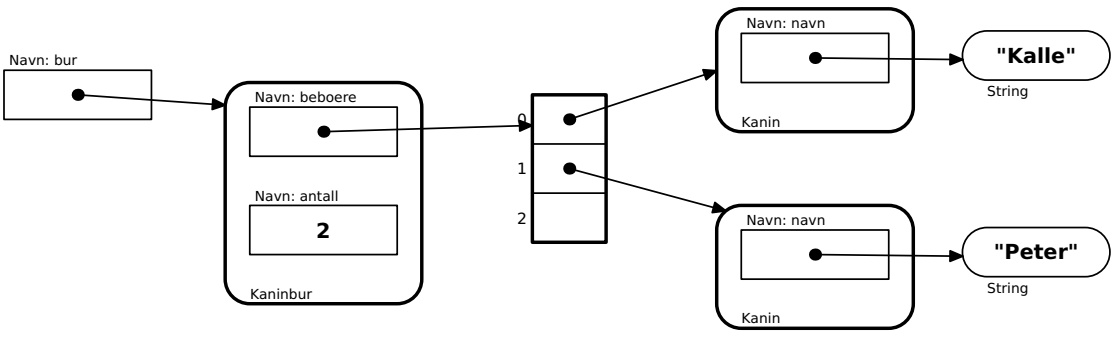
    Kaninbur (int størrelse) {
        beboere = new Kanin[størrelse];
    }

    void settInn (Kanin k) {
        if (antall < beboere.length) {
            beboere[antall] = k;  ++antall;
        }
    }
}

class Kanin {
    String navn;

    Kanin (String s) { navn = s; }
}
```

Når programmet har startet opp og generert ett Kaninbur- og to Kanin-objekter, har vi en datastruktur som vist i figur 5:



Figur 5: Datastruktur med kaninbur og kaniner

- Variabelen bur peker på et Kaninbur-objekt.
- Dette objektet har en instansvariabel beboere som peker på en array.
- Element nr 0 i denne arrayen peker på et Kanin-objekt med navn *Kalle*.[2]
- Element nr 1 i arrayen peker på et annet Kanin-objekt og dette har navn *Peter*.

[2]Legg merke til at tekster ikke er verdier men objekter av klassen String.