Invarianter

IN1010 uke 10

I dag

- Datainvarianter/klasseinvarianter
- Løkkeinvarianter
- «invarianten holder»
- Preconditions
- Postconditions

Datainvarianter/klasseinvarianter



- Invarianter er betingelser i programmene våre.
- Betingelsene er ofte knyttet til datastrukturen i objektene
  - → *datainvarianter* eller *klasseinvarianter*.
- Må **alltid være sann** for en instans av en klasse.
- Bidrar til at klassens datastruktur forblir **gyldig** og **konsistent** gjennom hele objektets livssyklus.
- Metoder kan foreta endringer på datastrukturen når de utfører operasjoner, men invarianten må holde etter at de terminerer.

Klasseinvariant eksempel:

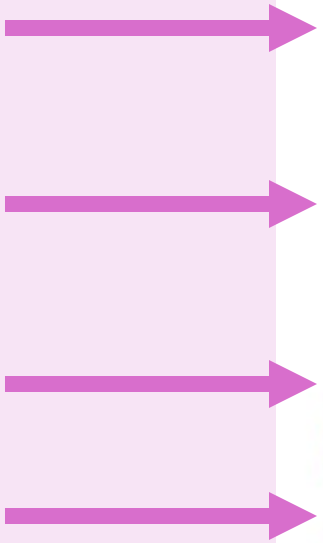


```java
public class StorreEnnNull {
  // Invariant: tall er alltid et heltall storre enn 0
  private int tall;

  public StorreEnnNull(int tall) {
    if (tall <= 0) {
      throw new IllegalArgumentException("[ERROR] Tallet ma vaere storre enn 0");
    }
    this.tall = tall;
  }

  public void settTall(int tall) {
    if (tall <= 0) {
      throw new IllegalArgumentException("[ERROR] Tallet ma vaere storre enn 0");
    }
    this.tall = tall;
  }
}
```

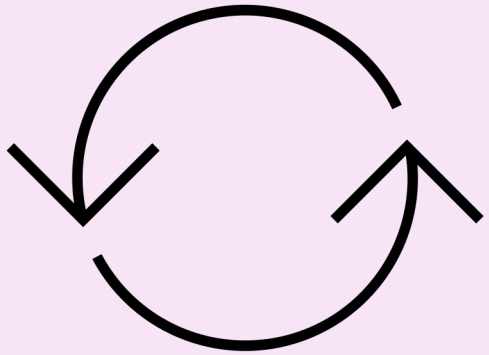
Datainvarianter i dobbeltlenket lenkeliste



```
public class Lenkeliste {
  class Node {
    /* Invariant: 'neste' peker til neste node i lista (eller er null hvis noden er den
    siste). 'neste' i 'forste' peker alltid paa en node hvis lista har >1 element. */
    Node neste;
    /* 'forrige' peker til forrige node i lista (eller er null hvis noden er forste).
    'forrige' i 'siste' peker alltid paa en node hvis lista har >1 element. */
    Node forrige; }
  // Invariant: hvis lista er tom, er 'forste' null. Ellers peker 'forste' paa en node.
  Node forste;
  // Invariant: hvis lista er tom, er 'siste' null. Ellers peker 'siste' paa en node.
  Node siste;
}
```

Løkkeinvarianter

- Når vi har betingelser knyttet til løkker, kaller vi dem for *løkkeinvarianter*.
- En løkkeinvariant er en betingelse som forblir **sann før og etter hver iterasjon**.
- Invarianten brukes til å bevise at en løkke fungerer som forventet og at den til slutt terminerer med ønsket resultat.



Løkkeinvariant i løkke som summerer partall



```java
public class SumPartall {
  public static void main(String[] args) {
      int sum = 0; int ovreGrense = args[0];
      // Invariant: I forkant av hver iterasjon, er 'sum' lik summen av partallene
      opp til og med tallet paa indeks i-1 (dersom tallet paa indeks i-1 er et partall).
      for (int i = 0; i <= ovreGrense; i++) {
          if (i % 2 == 0) sum += i;
      }
  }
}
```

Løkkeinvariant i «rett frem»-søk

```
public class SosialeMedier {
  String[] sosialeMedier = {"Snapchat", "TikTok", "Instagram", "Facebook"};

  public boolean finnesSosialtMedium(String medium) {
    /* Invariant: Før hver iterasjon av for-lokken ved indeks i (hvor 0 <= i <
      sosialeMedier.length), har elementene i arrayet sosialeMedier med indekser fra 0
      til i-1 blitt sjekket, og ingen av disse elementene matcher strengen medium.
      Derfor, hvis strengen medium er til stede i arrayet, maa den befinne seg paa en
      posisjon med indeks i eller hoyere. */
    for(int i = 0; i < sosialeMedier.length; i++) {
      if(sosialeMedier[i].equals(medium)) return true;
    }
    return false;
  }
}
```

«Invarianten holder»

- Vi bruker uttrykket "*invarianten holder*" før og etter et metodekall eller en løkkeiterasjon.
  - Betingelsen invarianten utgjør forblir sann, noe som bidrar til å sikre at programmet vårt fungerer som forventet.
- Å sikre at invarianter holder bidrar til
  - programmets stabilitet og pålitelighet
  - hjelper utviklere å forhindre feil og uønsket atferd.
- Håndhevelsen av invarianter er programmererens ansvar
- Krever nøye design og gjennomtenkt implementasjon av klassens metoder/løkkens logikk.



Postconditions og preconditions

I forbindelse med invarianter, snakker vi gjerne om *preconditions (førbetingelser)* og *postconditions (etterbetingelser)*.

Preconditions/førbetingelser

- = Betingelser som må være oppfylte før kjøringen av en metode eller løkke.
- De representerer et sett med krav eller forventninger som må være sanne for at en operasjon skal fungere som forventet.
- For eksempel vil en metodes spesifikke krav til en inputverdi utgjøre en førbetingelse.

Postconditions/etterbetingelser

- = betingelser som skal være oppfylt etter at metoden eller løkka er fullført.
- Beskriver den forventede tilstanden til programmet eller returverdier etter utførte operasjoner.
- Hvis en metode er ment å utføre en bestemt oppgave og produsere et bestemt resultat, så beskriver etterbetingelsen ønsket sluttresultat.

Eksempel før-/etterbetingelse

Førbetingelse:
argumentet til *finnesSosialtMedium(String medium)* må være en streng og ikke *null*. Utløser unntak hvis *medium == null*.

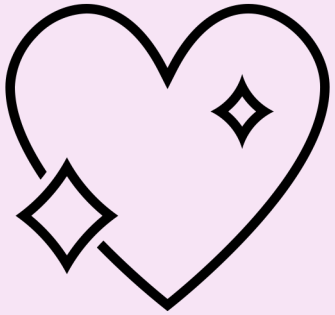
Etterbetingelse:
finnesSosialtMedium(String medium) ovenfor returnerer *true* hvis medium finnes og *false* hvis ikke.

```
public class SosialeMedier {
    String[] sosialeMedier = {"Snapchat", "TikTok", "Instagram", "Facebook"};

    public boolean finnesSosialtMedium(String medium) {
        // Precondition: 'medium' kan ikke vaere null.
        if (medium == null) {
            throw new IllegalArgumentException("[ERROR] 'medium' kan ikke vaere null.");
        }

        for (int i = 0; i < sosialeMedier.length; i++) {
            if (sosialeMedier[i].equals(medium)) {
                return true;
            }
        }
        return false;
    }
}
```

Jobbe med oppgaver



Innlevering 4 – et litt større program

Trix

Seminaroppgaver ligger på semestersiden → grupper → gruppe 3 → uke10