Scope

Scopet til en variabel er den delen av programmet hvor variablen er tilgjengelig. For tilgjengelighetsnivå i klasser og for klassevariabler, les _Modifikatorer.md_.

Når vi deklarerer en variabel i Java, er det ofte intensjonen å bruke eller modifisere denne variabelen et annet sted i programmet. Hvor vi kan bruke eller modifisere denne variabelen avhenger av hvor den ble deklarert, da vi kun kan bruke denne i et *scope*.

Se på følgende program:

```
class Main {
    void metode() {
        // Lokal variabel (Metode-nivå/scope)
        int x;
    }

    void add() {
        x += x;
        // Dette vil gi en error
    }
}
```

I dette programmet ble `x` deklareret i metoden `metode()`, og forsøkt endret i metoden `add()`. Dette vil gi feilen `error: cannot find symbol` da `x` ikke er tilgjengelig i metoden: `x` er ikke i samme *scope*. Det eneste området `x` er tilgjengelig i her, er i metoden `metode()`.

For å løse dette kan vi gjøre følgende:

```
class Main {
    // Instantsvariabel (Klasse-nivå/scope)
    int x;

    void add() {
        x += x;
    }
}
```

Her har vi flyttet `x` hakket opp, og `x` er nå tilgjengelig på klassenivå, eller *scopet* til klassen. For å gjøre det enklere å forstå, kan vi se på et par krøllparanteser `{ }` som et *scope*. Se på følgende program:

```
class Main {
    public static void main() {
        int x = 10;
        System.out.print(x);
```

```
        // 10

        {
            int y = 5;
            System.out.print(x+y);
            // 15

            {
                System.out.print(y);
                // 5
            }
        }

        System.out.print(y);
        // error: cannot find symbol
    }
}
```

Her har variabelen `y` *scope* innenfor krøllparantesene. Vi kan da bruke `y` innenfor blokken vi har definert, og i alle *scopes* som er en del av *scopet* y ble deklareret i. Variabelen `x` ble deklarert i *scopet* over blokken igjen, og vil derfor være tilgjengelig i alle blokkene. Siden `y` ikke er en del av *scopet* hvor vi prøver å kalle `System.out.print(y)`, vil vi få en feil når vi prøver å hente variabelen, da den ikke finnes.

Vi kan også bruke et eksempel som involverer å printe til konsollen inne i løkker. Se på følgende program:

```
class Main {
    public static void main() {
        for(int i = 0; i < 4; i++) {
            System.out.print(i);
        }
        // 123

        System.out.print(i);
        // error: cannot find symbol
    }
}
```

Her er variabelen `i` kun tilgjengelig inne i løkken, som er et eget *scope*. En for-løkke er en kodeblokk, definert med to krøllparanteser, og variabler som blir deklareret inne i denne blokken vil ikke være tilgjengelig utenfor denne blokken. For å rette opp i feilen kan vi deklarere variabelen utenfor denne blokken, slik at den er en del av samme *scope* som kallet på `System.out.print()`:

```java
class Main {
    public static void main() {
        int i;
        for(i = 0; i < 4; i++) {
            System.out.print(i);
        }
        //123

        System.out.print(i);
        // 4
    }
}
```

Kort oppsummert

- Generelt vil et par krøllparanteser `{ }` definere et *scope*.
- I java har vi som oftest tilgang til en variabel så lenge den er deklarert i samme kodeblokk (sett med krøllparanteser) eller i andre kodeblokker inni samme blokk variabelen ble deklarert i.
- En variabel definert på klasse-nivå kan bli brukt av alle metodene i klassen.
- Når en metode har samme lokalvariabel som en klassevariabel, kan vi bruke `this` for å referere til klassevariabelen.
- For at en variabel skal kunne bli brukt etter slutten av en løkke, må den bli deklarert før kodeblokken til løkken.