Rekursjon

Repetisjonskurs IN1010

Mai 2018

1 Om rekursjon

Kort fortalt er en rekursiv metode en metode som *kaller seg selv*.

```
public void rekursivMetode() {
    rekursivMetode();
}
```

Uten noe mer på plass vil ikke dette gå bra, metoden vil kalle seg selv uten noen måte å stoppe. Det vi trenger er et **basistilfelle**, et tilfelle hvor metoden slutter å kalle seg selv, og vi trenger at hvert kall på metoden går mot et basistilfelle.
Metoden gjør heller ingenting. Så når er det egentlig vi får bruk for at en metode kaller seg selv? Rekursjon kan brukes når en oppgave kan løses ved å løse mindre deler av samme oppgave. Fakultet er et godt eksempel. For et gitt tall n er $n! = n \cdot (n-1)!$ Vi kan skrive dette som en metode:

```
public int fakultet(int n) {
    if (n == 1) return 1;
    return n*fakultet(n-1);
}
```

Her løses $n!$ ved å regne ut $(n-1)!$, som er en enklere versjon av samme problem, og bruke at $n! = n \cdot (n-1)!$ Her er $n = 1$ basistilfellet vårt, og for hvert kall minkes n med 1 slik at vi kommer nærmere basistilfellet.

2 Hvordan skrive rekursive metoder

Hva må vi tenke på når vi skal skrive rekursive metoder? Det kan være greit å stille seg disse spørsmålene:

1. Hvordan kan oppgaven deles opp slik at den kan løses rekursivt?
2. Hva er basistilfellet?
3. Hvordan når vi basistilfellet?
4. Hva trenger metoden å "vite"? Hvilke parametre trenger den?

2.1 Eksempel - telle rekursivt

Vi ønsker å skrive en metode som gitt et heltall n først teller ned til 0 og så teller opp igjen til n. La oss se på nedtellingen først, og bruke spørsmålene over:

1. Dele opp oppgaven: Vi lar hvert kall på metoden skrive ut et tall og så kalle seg selv for å skrive ut resten.
2. Basistilfellet: Når tallet er 0 skal vi slutte å telle nedover
3. Hvordan nå basistilfellet: Vi må minke tallet som skal skrives ut for hver gang.
4. Parametre vi trenger: Vi må vite tallet som skal skrives ut for hver gang.

Altså: Om vi kaller metoden vår tellerRekursiv(int n) så skal metoden skrive ut tallet n, og dersom n ikke er 0 skal den kalle på $tellerRekursiv(n-1)$.
Hvordan telle oppover? Vi kan skrive ut tallet n igjen etter kallet på tellerRekursiv(n-1). Utskriften vil da utføres når metodekallet returnerer. Prøv gjerne selv!

```
public void tellerRekursiv(int n) {
    System.out.println("Tallet er " + n);
    if (n == 0 ) {
        return;
    }
    tellerRekursiv(n-1);
    System.out.println("Tallet er " + n);
}
```

3 Andre tilfeller

3.1 Hjelpemetoder

Noen gangen kan det være nyttig eller nødvendig å ha med hjelpemetoder, f.eks. for å starte rekursjonen, eller for å gjøre delberegninger.

3.2 Rekursjon og objekter

En rekursiv metode kan også kalle på samme metode i et annet objekt, som f.eks. i oblig 5 hvor gaa() metoden inneholder kall på nabo-rutene sine gaa() metoder.

4 Oppgaver

Oppgave 1

Skriv en rekursiv metode *int sumTilN(int n)* som tar inn et positivt heltall n, og regner ut og returnerer summen av alle tallene fra 1 til og med n.

Oppgave 2

Skriv en rekursiv metode *int produktTilN(int n)* som tar inn et positivt heltall n, og regner ut og returnerer produktet av alle tallene fra 1 til og med n.

Oppgave 3

Skriv en rekursiv metode *boolean finnesITekst(String tekst, char bokstav)* som returnerer true eller false utfra om bokstaven finnes i teksten eller ikke.

Oppgave 4

Skriv en rekursiv metode som gitt et array av int'er returnerer det minste tallet i arrayet.

Oppgave 5

Skriv en rekursiv metode som tar inn en String og skriver ut hver bokstav i stringen rekursivt.

Oppgave 6

Skriv en metode som tar inn en mappe og skriver ut navnet på alle filene i mappen og dens undermapper rekursivt.
Både filer og mapper representeres som File-objekter. Noen metoder i File klassen som kan være nyttige er isDirectory() (returnerer true dersom File-objektet er en mappe), listFiles() (returnerer et array av File objektene i mappen) og getName().

Relevante eksamensoppgaver

Tidligere eksamensoppgaver kan finnes på semestersiden til INF1010 V15, under Ressurser og htmleksamen2.2.html