Java for deg som kan Python

del 2

Dag Langmyhr
(dag@ifi.uio.no)

Hva er formålet med programmet vi skal lage?

Jeg skal vise et lite eksempel som demonstrerer

- Programparametre
- Lesing fra fil og fra tastaturet
- Java-biblioteket
- Unntak («exceptions»)
- Metoder
- Tegn og tekster (**`char`** og **`String`**)
- For-løkker
- Utskrift

Oppgave

Jeg skal lage et Java-program `WC` («word count») som leser en fil og skriver ut tre tall:

- antall linjer i filen
- antall ord i filen (dvs tegnsekvenser adskilt med blanke)
- antall tegn i filen

Eksempel

`datafil.txt`

```
Denne filen
har 2 linjer.
```

Da skal vi kunne kompilere og kjøre programmet vårt slik:

```
$ javac WC.java
$ java WC datafil.txt
2 linjer, 5 ord, 24 tegn
```

Løsningen

```
import java.io.File;
import java.util.Scanner;

class WC {
    public static void main (String[] args) {
        if (args.length != 1) {
            System.out.println("Bruk: java WC filnavn");
            System.exit(1);
        }

        try {
            Scanner f = new Scanner(new File(args[0]));
            tellFil(f);
        } catch (Exception e) {
            System.out.println("WC: Kan ikke lese " + args[0]);
            System.exit(2);
        }
    }
```

Løsningen (del 2)

```
private static void tellFil (Scanner f) {
    int antLinjer = 0, antOrd = 0, antTegn = 0;

    while (f.hasNextLine()) {
        String linje = f.nextLine();
        antTegn += linje.length();  ++antLinjer;

        boolean iEtOrd = false;
        for (int i = 0;  i < linje.length();  i++) {
            char c = linje.charAt(i);
            if (Character.isWhitespace(c)) {
                iEtOrd = false;
            } else {
                if (! iEtOrd) antOrd++;
                iEtOrd = true;
            }
        }
    }
    System.out.println(antLinjer + " linjer, " +
        antOrd + " ord, " + antTegn + " tegn");
}
}
```

Programparametre

Når vi kjører et program, kan vi gi det parametre:

```
$ java MittProg  292  min_mappe/min_fil
```

Disse finner vi i `String[]`-parameteren til `main`-metoden:

```
public static void main (String[] args) {
```

I dette tilfellet har arrayen `args` to elementer:

`args[0]: "292"` `args[1]: "min_mappe/min_fil"`

Lesing

Når vi skal lese fra en fil eller tasturet, trenger vi klassen `Scanner`.

Et `Scanner`-objekt opprettes slik:

```
Scanner tastatur = new Scanner(System.in);
Scanner fil = new Scanner(new File(filnavnet));
```

Husk

- Klassens `Scanner` og `File` må importeres:
  `import java.util.Scanner;` `import java.io.File;`
- Når vi skal lese en fil, må vi sjekke på unntak.

Lesing linje for linje

Det vanligste er å lese filen linje for linje; da trenger vi

```
Scanner s = ...;
while (s.hasNextLine()) {
   ... = s.nextLine();
}
```

`s.hasNextLine()` forteller om det er flere linjer igjen å lese; svaret er altså en **`boolean`**-verdi.

`s.nextLine()` leser neste linje; svaret er altså en tekst (et `String`-objekt)

Java-biblioteket

Java har et kjempebibliotek. Det finnes en interaktiv dokumentasjon på

https://docs.oracle.com/javase/8/docs/api/index.html?overview-summary.html

Dessuten finnes en oversikt over de viktigste klassene bakerst i læreboken.

Import fra biblioteket

Hvis vi skal bruke klasser i biblioteket (unntatt de i `java.lang`), må vi importere dem:

```
import java.io.File;
import java.util.Scanner;
```

Unntak («exceptions»)

Selv om kompilatoren finner de fleste feilene vi gjør, vil det kunne oppstå feil under kjøring:

- Vi ber om et array-element som ikke finnes:

```
if (a[i] > 0) {
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
    Index 10 out of bounds for length 10
        at ArrayFeil.main(ArrayFeil.java:6)
```

- Vi prøver å bruke en referanse som er **`null`**:

```
int i = minRef.verdi();
```

```
Exception in thread "main" java.lang.NullPointerException:
    Cannot invoke "MinKlasse.verdi()" because "<local1>" is null
        at RefFeil.main(RefFeil.java:4)
```

Noen ganger *krever* Java-kompilatoren at vi sjekker om feilen (unntaket) oppstår:

- Vi prøver å lese fil som ikke finnes:

```
Scanner s = new Scanner(new File("minfil.data"));
```

```
FilFeil.java:6: error: unreported exception FileNotFoundException;
    must be caught or declared to be thrown
        Scanner s = new Scanner(new File("minfil.data"));
                    ^
```

Hvordan tar vi hånd om unntak?

Slik sjekker vi for mulige unntak:

```
try {

  ... kode ...

} catch (Exception e) {
  ... feilhåndtering ...
}
```

Hvis unntaket oppstår i koden, hopper vi ut til feilhåndteringen.

Hvordan vet vi hvilke metoder som krever at vi håndterer feilen?

Hint!

Bruk kompilatoren.

Skriv først koden uten å tenke på mulige krav om å håndtere unntak. Legg dem inn når kompilatoren krever det.

Metoder

Python

```
def max (a, b):
    if a > b:
        return a
    return b
```

Java

```
int max (int a, int b) {
    if (a > b) {
        return a;
    }
    return b;
}
```

Metoder i Java deklareres og brukes nesten som i Python, men

- Alle metoder må ligge i en klasse.
- Innmaten i metoden angis med krøllparenteser ({ }).
- Vi må angi typen til verdien som returneres (eller **`void`** hvis ingen).
- Vi må angi typen til alle parametrene.

Tegn og tekster

I Java skiller vi mellom

- Enkelttegn er en verditype **`char`**
- Tekster lagres i objekter av klassen **`String`**

Enkelttegn

Eksempler:

```
char storA = 'A';  // NB! Enkeltfnutter ' '
char c;
c = '+';
```

Tekster

Tekster er objekter av bibliotekklassen **`String`**.

```
String s = "En tekst"  // NB! Dobbeltfnutter "..."
```

Vanligste operasjoner på tekster

`s1 + s2` skjøter sammen to tekster og lager en ny
NB! Fungerer også med **`s + v`** der `v` er hva som helst; `v` blir omdannet til en `String` før de to skjøtes sammen.

`s.length()` forteller hvor lang teksten `s` er.

`s.trim()` returnerer ny tekst uten innledende og avsluttende blanke.

`s.split("x")` deler teksten med angitt skilletegn; svaret er en `String`-array.

`s.charAt(i)` henter tegnet (en `char`) i posisjon nr `i`.

`s1.equals(s2)` sjekker om `s1` og `s2` er like.

NB! Bruk `.equals` for å sjekke likhet

Testen `s1==s2` sjekker om `s1` og `s2` er *samme `String`-objekt*. Det er noe annet.

```
String s1 = "Ab";
String s2 = "A";  s2 += "b";  // Nå inneholder s2 "Ab".

System.out.println("s1.equals(s2) gir " + (s1.equals(s2)));
System.out.println("s1 == s2 gir " + (s1==s2));
```

```
s1.equals(s2) gir true
s1 == s2 gir false
```

For-løkker

For-løkker ser ganske anderledes ut i Java enn i Python:

Python

```
for i in range(3):
    print(i)
```

Java

```
for (int i = 0;  i < 3;  i++) {
    System.out.println(i);
}
```

Generelt består Javas for-setninger av disse elementene:

```
for (initiering; test; oppdatering) {
    ...
}
```

Dette er egentlig en kompakt form for

```
{
    initiering
    while ( test ) {
        ...
        oppdatering
    }
}
```

Dette gir oss ganske frie tøyler:

```
for (int i = 256;  i > 0;  i = i/2) {
    System.out.print(i + " ");
}
System.out.println();
```

```
256 128 64 32 16 8 4 2 1
```

Utskrift

Utskrift til skjermen skjer med

```
System.out.println("Svaret er " + resultat);
```

Husk at vi kan skjøte sammen mange deler av ulike typer til en samlet `String` med `+`.

Utskrift til fil

For å kunne skrive til en fil trenger vi klassen `PrintWriter`:

```
PrintWriter utFil = new PrintWriter(new File(filnavn));
utFil.println("Svaret er " + resultat);
```

Husk

- Klassene `PrintWriter` og `File` må importeres.
- Instansieringen (dvs `new PrintWriter(new File(...))`) må legges i en **`try-catch`**-omgivelse.

Oppsummering

- Det er mye nyttig informasjon i feilmeldingene fra kompilering og kjøring.
- Programparametre havner i `String[]`-parameteren til `main`.
- Til lesing bruker vi en `Scanner` og metodene `hasNextLine()` og `nextLine()`.
- Vi kan håndtere unntak med et **`try`**...**`catch`**-oppsett.
- Enkelttegn lagres i **`char`**-variabler og tekster av vilkårlig lengde i `String`-objekter.
- Vi må bruke **`s1.equals(s2)`** for å sjekke om to tekster er like.
- Ofte trenger vi klasser fra standardbiblioteket; de må importeres.