Ukeoppgaver 1: 24.–28. aug (INF1000 - Høst 2009)

Et første møte med Java *(kapittel 1 og 2.1–2.3 i læreboka, "Rett på Java" 2. utg.)*

Mål
I første terminalstue-time skal du greie å skrive dine første Java-programer og få de til å fungere. På timene med tavleundervisning (teoritimene) får du mer informasjon om Java og kurset generelt, og gjennomgang av noen av disse oppgavene.

Oppgaver til teoritimen

1. **Det første programet:** *kap. 1, oppg. 1 (side 23 i læreboken)*
   Lag et program som skriver følgende tekst på skjermen:

   ```
   Rachmaninovs 3. klaverkonsert har et vakkert åpningstema
   ```

   Velg selv hvilket navn du vil gi klassen.

   Tips: Fremgangsmåte for å lage ditt første program:

   a. **Innlogging:** Du kan bruke din egen datamaskin eller en maskin på en av «terminalstuene» på universitetet. For å logge inn på terminalstuen trenger du bare å skrive ditt brukernavn og passord på velkomstskjermen til maskinen. Hvis du har med deg en bærbar pc til UiO kan du koble den til det trådløse nettet ved UiO *(klikk på teksten hvis du ønsker mer informasjon).* Hvis du sitter på en egen pc hjemme sjekk at du har installert en Java-kompilator før du går videre — Mac og Linux kommer som regel med Java-kompilator, men på Windows må du vanligvis installere kompilatoren (PDF) selv.

   b. **Redigering:** Deretter starter du et tekstredigerings-program. Det finnes mange slike å velge mellom, for eksempel Emacs (som finnes på Linux-maskinene i terminalstuen) eller TextPad (på Windows-maskinene). Skriv programmet ditt i redigerings-programmet. Se følgdende eksempel for gode tips til hvordan du kan bygge opp programmet ditt — det eneste du trenger å endre er teksten som står i anførselstegn! Lagre programmet i en fil med riktig filnavn. Filnavnet skal være likt klassenavnet etterfulgt av `.java`, f.eks. følgende program (fra side 16 i læreboken) skal lagres med filnavnet `Utskrift.java` fordi klassen i programmet heter `Utskrift`. Husk at store og små bokstaver har betydning her.

   ```
   class Utskrift {
       public static void main(String[] args) {
           System.out.println("Beethoven komponerte Skjebnesymfonien");
       }
   }
   ```

   c. **Kompilering:** Videre bruker du et «kommandovindu», også kalt terminalvindu, for å kompilere programet. På *Linux*-maskinene på Ifi åpnes det automatisk et kommandovindu når du logger inn, men du kan også åpne det ved å trykke høyre musknapp og velge *Open Terminal*. På *Mac* finner du også kommandovinduet under navnet «Terminal». På *Windows* kan du åpne kommandovinduet ved å holde inn Windows-tasten (tasten til venstre for Alt-tasten) mens du trykker bokstaven `r`, og så skrive `cmd` (og trykke `Enter`).

   For å kompilere programmet skriver du på kommandovinduet kommandoen `javac` etterfulgt av mellomrom og filnavn, for eksempel:

   ```
   > javac Utskrift.java
   ```

   NB! Husk at du ikke skal taste inn ">"-tegnet, dette tegnet kalles *prompt* og brukes bare for å angi at det som følger er en kommando som skal tastes inn på kommandovinduet. Promptet du ser på skjermen avhenger bl.a. av operativsystem, på Unix (Linux) kan det f.eks. se slik ut:
   bruker@maskin ~ $

d. **Debugging:** Hvis kompilatoren oppdager feil i programmet, retter du disse i tekstredigerings-programmet, lagrer, og kompilerer igjen. Feilmeldingen fra kompilatoren sier ofte hvor i programmet feilen sannsynligvis ligger, bl.a. vha. linjenummer. Bruk dette til å finne feilen, og husk at store og små bokstaver har betydning, både i programkoden og filnavnet. Mer info om kompilering og debugging kan du finne på side 17-18 i læreboken, og på Terminalvaktenes hjelpesider.

e. **Kjøring:** Når du har rettet skrivefeilene slik at kompilatoren ikke gir noen feilmelding kan du «kjøre» programmet ved å skrive `java Klassenavn` på kommandovinduet, for eksempel:

```
> java Utskrift
```

2. **Finn fem syntaksfeil:** *kap. 1, oppg. 3 (side 24)*
Finn feilene i dette programmet:

```
class Utskrift {
    public stitac void main(String args) (
        System.out.println("Beethoven skrev Skjebnesymfonien")
        System.out.println("og åtte andre symfonier.);
    }
}
```

3. **Enkel formatering av utskrift:** *kap. 1, oppg. 5 (side 24)*
Vi ønsker å skrive ut følgende tekst på skjermen (med samme pyramideformatering):

```
    a
   a a
  a b a
 a c c a
a d f d a
```

Lag et Java-program som gjør dette, ved å bruke flere utskriftssetninger.

Tips:
Idéen her er å finne hvor og hvordan man kan legge inn mellomroms-tegn i programmet for å få til pyramideformateringen. Senere i kurset, i kapittel 3 (side 52) vil du lære en annen måte å løse denne oppgaven på hvor Java tar seg av pyramideformateringen.

4. **Areal av rektangler:** *kap. 2, oppg. 1 (side 43)*
Skriv et program som beregner arealet av rektanglene med disse sidestørrelsene: 3 og 5, 7 og 3, samt 4 og 9, og som skriver ut resultatet med en passende fortekst. Kompier og kjør programmet.

Tips:
Denne kan løses ved å følge oppskriften fra programmet på side 30 og skrive et lignende program, men med litt andre variabelnavn og beregninger.

5. **Deklarasjon og initialisering av variabler:** *kap. 2, oppg. 2 (side 43)*
Finn feilene i dette programmet:

```
class Volum {
    public static void main (String[] args) {
        integer lengde, bredde, høyde;
        lengde = 3;
        volum  = lengde * bredde * høyde;
        System.out.println("Volumet er: " + volum);
    }
}
```

6. **Typiske feilmeldinger:**
Hva betyr følgende feilmeldinger, som kompilatoren spytter ut når vi prøver å kompilere og debugge dette programmet:

```
1  class Test {
2      public static void main(String[] args) {
3          int jens;
4          int siv = Jens * 3;
5          System.out.println("Svar: " siv);
6          erna = siv - jens;
7          System.out.printn(erna);
8      }
```

a. `Test.java:8: reached end of file while parsing`
```
        }
         ^
```

b. `Test.java:4: cannot find symbol`
```
symbol  : variable Jens
location: class Test
   int siv = Jens * 3;
             ^
```

c. `Test.java:4: variable Jens might not have been initialized`
```
   int siv = Jens * 3;
             ^
```

d. `Test.java:5: ')' expected`
```
   System.out.println("Svar: " siv);
                              ^
```
`Test.java:5: illegal start of expression`
```
   System.out.println("Svar: " siv);
                                 ^
```

e. `Test.java:6: cannot find symbol`
```
symbol  : variable erna
location: class Test
   erna = siv - jens;
   ^
```

f. `Test.java:7: cannot find symbol`
```
symbol  : method printn(int)
location: class java.io.PrintStream
   System.out.printn(erna);
             ^
```

Hvis vi fjerner "{"-krøllparentesen på linje 2 får vi ca. 10 nye feilmeldinger, bl.a. følgende. Hva tipper du er grunnen til at så mange feilmeldinger forårsakes av bare denne enkle lille feilen?

g. `Test.java:2: ';' expected`
```
    public static void main(String[] args)
                                           ^
```
`Test.java:5: <identifier> expected`
```
   System.out.println("Svar: " + siv);
                     ^
```
`Test.java:5: illegal start of type`
```
   System.out.println("Svar: " + siv);
                      ^
```

Oppgaver til terminaltimen

1. **Det første programet:** *kap. 1, oppg. 1 (side 23 i læreboken)*
   (Se punkt 1. ovenfor)

2. **Kompilerings- vs. kjørefeil:** *kap. 1, oppg. 2 (side 23)*
   Med utgangspunkt i programmet nedenfor skal du utføre de angitte endringene en for en. Prøv først å **kompilere** programmet og studér eventuelle feilmeldinger. Dersom programmet lar seg kompilere, så forsøk å **kjøre** det. For hvert forsøk skal du rette programmet tilbake til utgangspunktet og kontrollere at det virker. Her er programmet (som skrives inn i en fil med navnet `Studie.java`):

```
class Studie {
    public static void main(String[] args) {
        System.out.println("Rett på Java");
    }
}
```

a. Fjern ordet *static*.
b. Fjern ordet *void*.
c. Skriv *studie* i stedet for *Studie*.
d. Skriv *Skrått* i stedet for *Rett*.
e. Fjern den siste krøllparentesen.

3. **Enkel formatering av utskrift:** *kap. 1, oppg. 5 (side 24)*
(Oppgaveteksten står i punkt 3. ovenfor)

4. **Areal av rektangler:** *kap. 2, oppg. 1 (side 43)*
(Oppgaveteksten står i punkt 4. ovenfor)

5. **Typiske feilmeldinger:**
(Oppgaveteksten står i punkt 6. ovenfor). Korrigér feilene slik at programmet kompilerer.

6. Hvis du har tid til øvers kan du begynne med Oblig 1 (PDF).

7. Hvis du har enda mer tid kan du også begynne å titte litt på noen av disse kildene med nyttig informasjon. Det er bare de 2 første som er viktigst, men spesielt interesserte kan fortsette nedover i lista og finne enda mer informasjon.
   - **Kursets hjemmeside:** ifi.uio.no/inf1000/h09 er din viktigste kilde til informasjon om kurset. Du bør bli godt kjent med disse websidene, og det forutsettes at du følger med på beskjedene som publiseres der (særlig i forbindelse med de 4 obligatoriske oppgavene). Kurshjemmesiden har bl.a. lenker til forelesningsnotater, obliger, ukeoppgaver, innleveringssystemet Joly, Godkjentsystemet wwws, osv.
   - **Webmail:** webmail.uio.no Her finner du din e-post-konto på universitetet. Denne bør du sjekke regelmessig, eller evt. sette opp automatisk videresending av mailene dine til en annen e-post-adresse (se «UiO-instillinger» når du er inne i systemet). Du kan bruke som e-post adresse: *brukernavn*`@ulrik.uio.no`
   - **IfiDVD:** ifi.uio.no/ifidvd Finnes både på nettet og som fysisk plate du kan få gratis (på gruppene eller Ifi-ekspedisjonen). Denne DVD-ROM-platen inneholder mye nyttig programvare du kan installere i din egen datamaskin.
   - **Terminalvaktenes hjelpesider:** termvakt.uio.no har info om terminalstuene; programvare tilgjengelig i disse, bl.a. om Emacs og Linux; og mye mer.
   - **USIT:** usit.uio.no har offisiell info om student-IT ved universitetet, bl.a. om tilkobling og bruk av egen bærbar eller stasjonær maskin, og brukernavn og passord (husk å skifte passord!). Ifi-drifts websider har info om IT-tjenester ved Ifi.
   - **Suns offisielle websider om Java:** Her finner du bl.a. siste versjon av Java (for tiden JDK 6 Update 16), og mye dokumentasjon, f.eks. om API og en *tutorial*.
   - **Lærebokens hjemmeside:** Har bl.a. løsningsforslag til noen av oppgavene i boken, og lenke til en veiledning (PDF) om hvordan du kan installere Java og easyIO på egen Windows-pc (selv om noen av lenkene i websiden ikke fungerer akkurat nå).
   - **Informatikk-forkurset:** Har forelesningsnotater og lab-oppgaver om Linux og Java fra forkurset som foregikk 12. - 14. aug. Der finner du også mange lenker til mer informasjon om Linux, og om hvordan du kan **koble deg til UiO** hjemmefra eller overføre filer.
   - Wikipedia: Har mye nyttig informasjon om Java og relaterte teknologier, og mye mer!
   - Google: Hvis du ikke skjønner en feilmelding fra kompilatoren kan du lime inn feilmeldingen i søkefeltet til Google og finne hjelp!
   - **Gruppelærer:** Du kan alltid sende mail til din gruppelærer hvis du har spørsmål, kommentarer, e.l. *NB! Det kommer snart egne websider for hver av gruppene...*

Løsningsforslag

Du får sannsynligvis mest utbytte av løsningsforslagene hvis du har forsøkt å løse oppgavene på egen hånd først.