Løsningsforslag ukeoppg. 2: 31. aug - 6. sep (INF1000 - Høst 2011)

Variabler, uttrykk, fogreninger (if-else) *(kapittel 2 og 4.1-4.2 i læreboka, "Rett på Java"* ***3. utg.****)*

Mål
Få trening i teorien du trenger for å løse Oblig 1, dvs. bruk av variabler og uttrykk av forskjellige typer (int, double, og boolean), if-else-setninger (også kalt forgreninger), og feilmeldinger (både kompileringsfeil og kjørefeil).

Oppgave merket med nøkkelsymbol er plukket ut som spesielt representativ for de viktigste temaene fra ukens forelesning, og alle bør ha som minimumsmål å løse denne selvstendig.

Oppgaver

1. **Areal av rektangler:** *kap. 2, oppg. 1 (side 45)*
   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 oppgaven kan løses ved å følge oppskriften fra programmet på side 32 og skrive et lignende program, men med litt andre variabelnavn og beregninger.

```
class Rektangel {
   public static void main(String[] args) {
       int bredde;
       int lengde;
       int areal;

       bredde = 3;
       lengde = 5;
       areal = lengde * bredde;
       System.out.println("Areal av et rektangel med bredde " + bredde
                          + " og lengde " + lengde + " er " + areal);
       bredde = 7;
       lengde = 3;
       areal = lengde * bredde;
       System.out.println("Areal av et rektangel med bredde " + bredde
                          + " og lengde " + lengde + " er " + areal);
       bredde = 4;
       lengde = 9;
       areal = lengde * bredde;
       System.out.println("Areal av et rektangel med bredde " + bredde
                          + " og lengde " + lengde + " er " + areal);
   }
}

Kjøreeksempel:
> java Rektangel
Areal av et rektangel med bredde 3 og lengde 5 er 15
Areal av et rektangel med bredde 7 og lengde 3 er 21
Areal av et rektangel med bredde 4 og lengde 9 er 36
```

2. **Deklarasjon og initialisering av variabler:** *kap. 2, oppg. 2 (side 45)*
   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);
    }
}
```

```
linje 3: - heltallstypen heter "int", ikke "integer".
linje 5: - "volum" er ikke deklarert.
         - "bredde" er ikke initialisert.
         - "høyde" er heller ikke initialisert.
```

Disse feilene kan fikses ved å endre linje 3-5 til f.eks.:

```
    int lengde, bredde = 1, høyde = 2;
    lengde = 3;
    int volum  = lengde * bredde * høyde;
```

3. **Lovlig eller ikke?** *kap. 2, oppg. 5 (side 46)*
Hvilke av disse programsetningene er lovlige, og for de som er lovlige: hvilken verdi får variabelen? Du kan i hvert av punktene regne med at variablen som deklareres ikke er deklarert tidligere i programmet.

```
(a)  int x = 4;                     // lovlig, x får verdien 4.
(b)  int x = 4.56;                  // ikke lovlig, tap av informasjon.
(c)  int x = (int) 4.56;            // lovlig, x får verdien 4.
(d)  int z = 1/2;                   // lovlig, z får verdien 0.
(e)  double x = 5.7723;             // lovlig, x får verdien 5.7723.
(f)  double a = (int) 5.7723;       // lovlig, a får verdien 5.
(g)  double x = 6;                  // lovlig, x får verdien 6.
(h)  double tall = true;            // ikke lovlig, true er ikke en double.
(i)  char c = "hei på deg";         // ikke lovlig, en char-variabel kan ikke inneholde en tekst.
(j)  char c = "&";                  // ikke lovlig, "&" er en tekst.
(k)  char c = '&';                  // lovlig, c får verdien &.
(l)  boolean a = sann;              // ikke lovlig, "sann" er ikke en boolsk verdi.
(m)  boolean verdi = true && false; // lovlig, verdi får verdien false.
(n)  boolean a = (true == true);    // lovlig, a får verdien true.
(o)  boolean a = (false == false);  // lovlig, a får verdien true.
(p)  boolean a = (a == a);          // ikke lovlig, a er ikke definert på høyre side av likhetstegnet.
(q)  String t = 'hei på deg';       // ikke lovlig, tekst skal angis i doble anførsel.
(r)  String t = "hei på deg";       // lovlig, t får verdien "hei på deg".
(s)  String t = "" + 7.33;          // lovlig, t får verdien "7.33" (som tekst).
(t)  String tekst = 7.33;           // ikke lovlig, 7.33 er ikke en String, men en double.
```

4. **Uttrykk med int og double**
Avgjør i hvert av disse tilfellene, uten å bruke datamaskin, hvilken verdi som blir skrevet ut på skjermen. Husk at reglene for divisjon i Java avhenger av om vi jobber med heltall eller flyttall (=desimaltall) (se side 38-39 i læreboka).

```
 1.  System.out.println(10 / 9);            //  1
 2.  System.out.println(10 / 10);           //  1
 3.  System.out.println(10 / 11);           //  0
 4.  System.out.println((int) 3.65);        //  3
 5.  System.out.println((int) 3.65 - 4);    //  -1
 6.  System.out.println((int) (3.65 - 4));  //  0
 7.  System.out.println(Math.round(3.65));  //  4
 8.  System.out.println(1 / 2);             //  0
 9.  System.out.println(1 / 2.0);           //  0.5
10.  System.out.println(1.0 / 2);           //  0.5
11.  System.out.println((double) 1 / 2);    //  0.5
12.  System.out.println((double) (1 / 2));  //  0.0
13.  System.out.println(73 / 10);           //  7
14.  System.out.println(73 % 10);           //  3
```

5. **If-else med logisk uttrykk:** *kap. 4, oppg. 1 (side 84) med en endring*
Vi gjør en liten endring på oppgaven: siden innlesing av input fra tastatur mens programmet kjører er tema for neste uke, så setter vi ønsket verdi for `alder` i en variabel i begynnelsen av programmet. Bruk f.eks. alder = 20 og kjør programmet, og prøv deretter med en annen verdi.
Lag et program som avgjør, basert på alderen til en person, om personen kan få reise med trikken til halv pris. Resultatet skal skrives ut på skjermen. Vi antar at reglene er slik at alle under 12 år og alle over 64 år får reise for halv pris, mens alle andre må betale full pris. *Programmet skal bruke en variabel for å holde på alderen.*

Tips: Se eksemplet nederst på side 78. I denne oppgaven trenger du også logiske uttrykk (side 41).

```
import easyIO.*;

class Trikken {
    public static void main(String[] args) {
        In tast = new In();

        int alder;
        int alderBarn = 12;
        int alderPensjonist = 64;

        alder = 50; // Test også med 10 og 70
        System.out.println("Alder: " + alder);

        //  Logisk uttrykk som gir "true" hvis alder er mindre
        //  enn "alderBarn" ELLER større enn "alderPensjonist":
        if (alder < alderBarn || alder > alderPensjonist) {
            System.out.println("Du kan betale halv pris.");
        } else {
            System.out.println("Du må betale full pris.");
        }
    }
}
```

```
KJØREEKSEMPEL:
> java Trikken
Alder: 50
Du må betale full pris.
```

6. **Beregning av skatt i Ruritania:** *kap. 4, oppg. 2 (side 84) med en endring*
   Vi gjør en liten endring på oppgaven: siden innlesing av input fra tastatur mens programmet kjører er tema for neste uke, så setter vi ønsket verdi for `inntekt` i en variabel i begynnelsen av programmet. Bruk f.eks. inntekt = 10000 og kjør programmet, og prøv deretter med en annen verdi.
   I det fiktive landet Ruritania er skattereglene slik at hvis en person har inntekt < 10 000, så betaler man 10% skatt på hele inntekten, og hvis inntekten ≥ 10 000, så betaler man 10% skatt på de første 10 000 kronene og 30% skatt på resten av inntekten. Lag et program som regner ut og skriver ut hvor mange kroner som skal betales i skatt. *Programmet skal bruke en flyttalls-variabel for å holde på inntekten.*

```
import easyIO.*;

class Skatt {
    public static void main (String[] args) {
        In tast = new In();
        double inntekt;
        double skatt;

        inntekt = 11000; // Test også med 2000 og 20000
        System.out.println("Inntekt: " + inntekt);

        if (inntekt < 10000) {
            skatt = inntekt * 0.10;
        } else {
            skatt = (10000 * 0.10) + (inntekt - 10000) * 0.30;
        }

        System.out.println("Skatten blir: " + skatt);
    }
}

KJØREEKSEMPEL:
> java Skatt
Inntekt: 11000
Skatten blir: 1300.0
```

7. ***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 println(int)
location: class java.io.PrintStream
   System.out.println(erna);
              ^
```

a. Feilmeldingen i a. ovenfor ("**reached end of file while parsing**") betyr at Java-kompilatoren kom til slutten av filen men likevel ikke fant avslutning av elementene den holdt på å kompilere. I dette tilfellet er det en krøllparentes som ikke var avsluttet, fordi som du ser er det to åpnings-krøllparenteser ("{") i programmet, men bare én avslutnings-krøllparentes ("}"). Denne feilen retter vi ved å legge til en avslutnings-krøllparentes ("}") til slutt i filen.

b. "**Cannot find symbol**" er den vanligste feilmeldingen i Java, og **betyr at noe ikke er deklarert**. I dette tilfellet viser feilmeldingen at variabelen "Jens" med stor "J" ikke er deklarert i programmet. Denne feilmeldingen får man så ofte at det er lurt å lære seg å lese alle detaljer i feilmeldingen.

Første linje i feilmeldingen angir hvilken linje i programet Java-kompilatoren oppdaget feilen i, i dette tilfellet ser vi at det er linje 4 (fordi det står Test.java:4 i feilmeldingen). Neste linje i feilmeldingsteksten sier hva som ikke var deklarert, i dette tilfellet ser vi at det er "variabelen Jens". Og de siste 3 linjene i feilmeldingen angir nøyaktig hvor i koden det udeklarerte symbolet er forsøkt brukt. I dette tilfellet ser vi at det er i klassen Test, og akkurat der identifikatoren "Jens" står (se det lille hatt-tegnet "^" under Jens).

Denne feilen retter vi ved å endre variabelnavnet til "jens" med liten "j" på linje 4, slik at navnet blir likt med navnet vi deklarerte variabelen med på linje 3 (int jens;).

c. Feilmeldingen "**variable Jens might not have been initialized**" betyr at Java-kompilatoren mener vi bør gi en startverdi til variabelen Jens. Dette fikser vi ved å endre linje 3 til f.eks.: (husk også at vi endret variabelnavnet til "jens" med liten "j")

```
    int jens = 1;
```

d. "**) expected**" betyr at Java-kompilatoren fant noe uventet i koden på et sted der det hadde passet bedre med en ")"-parentes. Legg merke til hvor "hatten" ("^") er plassert i feilmeldingen. Undersøker vi nærmere det stedet i linjen, så skjønner vi fort at det skulle stått en "+" der, så vi korrigerer linje 5 til:

```
    System.out.println("Svar: " + siv);
```

e. Et annet eksempel på "**cannot find symbol**". Som vi så i punkt b. over så betyr denne feilmeldingen at et symbol (angitt på linje 2 i feilmeldingen) ikke er deklarert. Videre ser vi at sombolet som ikke er deklarert er "variable erna". Da vet vi at løsningen er å deklarer typen til erna, f.eks. ved å legge til nøkkelordet "int" foran på linjen:

```
    int erna = siv - jens;
```

f. En annen vri på "**cannot find symbol**". Denne gangen ser vi at det er symbolet "**method** println(int)" som ikke er deklarert. Ser vi nøye på hvor "hatten" peker nå, så skjønner vi raskt at det er en stavefeil i metodenavnet println. Vi fikser det ved å endre "println" til "println" på linje 7:

```
    System.out.println(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);
                      ^
```

g. Problemet er at når vi tar bort "{"-krøllparentesen, så har vi tatt bort angivelsen av at kroppen til metoden main startet der (fordi innmaten til alle metoder må starte med "{" i Java). Dermed går kompilatoren i surr og skjønner ikke noe av det som kommer etterpå. Den venter seg bare deklarasjoner, som er det eneste som kan stå utenfor metoder, og gir feilmelding for alle andre typer setninger (de som ikke er deklarasjoner).

8. ***Tips om tilpasning av Emacs:***
NB! Emacs er ikke pensum, det er bare ett av mange mulige redigeringsprogrammer du kan bruke for å skrive dine Java-programmer. Her kommer et nyttig lite tips for de som er interessert i Emacs, og link til mange flere tips. Emacs kan konfigureres på veldig mange måter slik at programmering blir mer behagelig. Dette gjøres ved å opprette og redigere en spesiell fil i ditt hjemmeområde, kalt `.emacs`. Denne oppgaven viser én nyttig konfigurasjonsmulighet i Emacs, for å fjerne velkomstskjermen som Emacs viser ved oppstart, slik at du slipper å ta den vekk hver gang du skal starte arbeidet.

Start Emacs for å se velkomstskjermen, f.eks. ved å taste kommandoen `emacs&` i et kommandovindu. Hvis Emacs-vinduet ikke viser noen "velkomstside" (hvor det står Emacs i store grafiske bokstaver, og diverse tekst i forskjellige farger) så er tilpasningen allerede gjort på ditt hjemmeområde og du trenger ikke gjøre noe mer. Hvis du derimot fikk opp velkomstsiden til Emacs, så kan du fortsette med følgende steg og bruke denne samme Emacs-en du startet nå for å ta bort velkomstskjermen, slik at den ikke forstyrrer neste gang du åpner Emacs. Trykk Ctrl-x etterfulgt av Ctrl-f og så taster du navnet på konfigurasjonsfilen:

```
Find file: ~/.emacs
```

Sjekk at nederste linje i Emacs ser akkurat ut som dette, med tilde-tegn ("~"), skråstrek, og punktum rett etter `"Find file: "` og før `emacs`. Trykk Enter. Dette vil åpne (og hvis nødvendig opprette) konfigurasjonsfilen og flytte tekstmarkøren inn i den, slik at du kan redigere innholdet. Skriv (eller legg til) følgende linje i filen:

```
(setq inhibit-splash-screen t)
```

Deretter lagrer du filen ved å trykke Ctrl-x Ctrl-s Ferdig! Nå kan du avslutte Emacs-en, og neste gang du starter programmet vil du ikke få opp den gamle forstyrrende "velkomstsiden".

Du kan finne mange andre tips til Emacs i lab-oppgavene til Forkurs i informatikk, blant annet tips om klipping/liming (Ctrl-w og Ctrl-y), åpning av flere samarbeidende Emacs-vinduer (Ctrl-x 52), automatisk korrigering av innrykk i et helt Java-program (Java > Indent Line or Region), og enkle tastetrykk som gir tekster som `"public static void main(String[] args)"` og `"System.out.println()"` (abbrevs).

***Tibakemelding** om dette oppgavesettet kan du skrive i bloggen eller sende på mail til josek [a] ifi.uio.no*