UiO : Universitetet i Oslo

Ukeoppgaver 2: 6. - 10. sep (INF1000 - Høst 2010)

Variabler og uttrykk *(kap. 2.3 - 2.6),* ***terminal I/O*** *(kap. 3.1-3.5),* ***if-setninger*** *og* ***løkker*** *(kap. 4.1 - 4.4), og litt om* ***arrayer*** *(kap. 5.1 - 5.2)*

Mål: Øve på bruk av variable og uttrykk av forskjellige typer (int, double, og boolean), innlesing og utskrift til terminal, if-setninger, løkker, og en liten oppgave med array.

Oppgaver til teoritimen

1. **Presedensregler – i hvilken rekkefølge utføres regneoperasjonene?**
   Avgjør i hvert av disse tilfellene, uten å bruke datamaskin, hvilken verdi som blir skrevet ut på skjermen.

   For å svare på denne oppgaven må du kjenne reglene for evaluering (utregning) av numeriske uttrykk i Java. Kortversjonen av disse er at uttrykk beregnes fra venstre mot høyre, men multiplikasjoner og divisjoner utføres før addisjoner og subtraksjoner. Du kan lese mer om dette på side 37 i læreboka.

```
1.  System.out.println(3 * 4 + 5);
2.  System.out.println(3 + 4 * 5);
3.  System.out.println((3 + 4) * 5);
4.  System.out.println(1 / 2 * 5);
5.  System.out.println(1 / 2 * 5.0);
6.  System.out.println(1.0 / 2 * 5);
7.  System.out.println(4 * 2 * 2 * 1);
8.  System.out.println(4 / 2 / 2 / 1);
9.  System.out.println(5 * 4 / 3 * 2 / 1);
```

   Hvis du ønsker mer info kan du se følgende oversikt med presedens for alle operatorene. Jo høyere oppe på denne lista operatoren står jo før blir den utført, mens de som står på samme nivå er likestilt og utføres fra venstre til høyre. Tabellen er tatt fra www.janeg.ca/scjp/oper/precedence.html, og er basert på side 378 i *The Java Programming Language*:

Operator type	**Operators**
Postfix	`[] . (params) expr++ expr--`
Unary	`++expr --expr +expr -expr ~ !`
Creation or Cast	`new (type)expr`
Multiplicative	`* / %`
Additive	`+ -`
Shift	`<< >> >>>`
Relational	`< > >= <= instanceof`
Equality	`== !=`
Bitwise AND	`&`
Bitwise exclusive OR	`^`
Bitwise inclusive OR	`|`
Logical AND	`&&`
Logical OR	`||`
Ternary	`? :`
Assignment	`= += -= *= /= %= >>= <<= >>>= &= ^= |=`

2. **Logiske uttrykk:** *(tabell 2.5, side 39)*
Avgjør hvilken verdi den boolske variabelen *b* får etter hver av disse tilordningssetningene, når vi antar at heltallsvariablene *x*, *y*, og *z* har fått verdier som følger: `int x = 3, y = 4, z = 1;`

```
1.  b = (x < y) && (y < z);
2.  b = (x < y) || (y < z);
3.  b = ! (x < y);
4.  b = x < y;
5.  b = x <= y;
6.  b = (x == y);
7.  b = (x != y);
```

Hint: Eksempel på hvordan man kan bruke slike boolske uttrykk i et program:

```
int x = 3, y = 4, z = 1;
if ((x < y) && (y < z)) {
    System.out.println("Det stemmer at x < y < z."); // Blir dette skrevet ut?
}
```

3. **Innlesing fra terminal:** *kap. 3, oppg. 3 (side 70)*
Lag et program som ber om og leser inn to flyttall. Programmet skal deretter regne ut produktet av de to tallene og skrive ut svaret. Her er et eksempel på hvordan en kjøring av programmet kan se ut:

```
Oppgi verdien til x: 9
Oppgi verdien til y: 10
Produktet av x og y er 90.0
```

Hint: Følg malen fra eksemplet på side 56. Innlesing fra terminal bør gjøres i to steg, først en utskrift som sier til bruker hva hun skal taste inn (det kalles for *ledetekst*), og så kommer selve setningen som henter input fra tastaturet. Bruk her `tast.inDouble()` og ikke `tast.inInt()`. Disse stopper kjøringen av programmet inntil bruker har tastet inn et svar. Når bruker har gjort det, blir tallet lagret i den angitte variabelen, og kjøringen av programmet fortsetter med neste setning.

4. **Utskrift med 2 desimaler på to måter: easyIO og printf**
Ved utskrift av desimaltall er det ofte ønskelig å kontrollere hvor mange desimaler som skrives ut. Bruker vi `System.out.println(tall)` til å skrive ut en **`double`**-variabel *tall*, får vi med alle desimalene som er lagret i *tall* (med unntak av avsluttende nuller i desimalene). To måter å kontrollere antall desimaler som skrives ut er:

   printf: I nyere versjoner av Java (5.0 og 6, også kalt 1.5 og 1.6) finnes en metode som gir oss god kontroll over utskriften. Hvis vi skriver:

   ```
   System.out.printf("Tall: %.2f\n", tall);
   ```

   så får vi skrevet ut verdien i variabelen *tall* med 2 desimaler, og hvis vi skriver:

   ```
   System.out.printf("Tall: %7.2f\n", tall);
   ```

   så får vi skrevet ut *tall* på 7 plasser (eller så mange som er nødvendig) og med 2 desimaler. Om nødvendig fyller Java på med blanke tegn til venstre slik at hele utskriften tar 7 plasser. Koden `\n` (omvendt-skråstrek etterfulgt av n) gir et linjeskift, og kan tas bort i setningene over hvis du ikke ønsker linjeskift. Flere eksempler på bruk av `printf` kan du se her (PDF) [Y.D. Liang *"Introduction to Java Programming"* 5. utg., side 64], og på Wikipedia.

   easyIO: Pakken easyIO gir tilsvarende resultat med følgende setninger:

   ```
   skjerm.outln(tall, 2);
   skjerm.outln(tall, 2, 7);
   ```

   For å kunne bruke dette må programmet ha setningnen `"import easyIO.*;"` øverst, og setningen `"Out skjerm = new Out();"` plassert på et passende sted i programmet før de ovennevnte `skjerm.outln()`-linjene, se eksemplet på side 50 i læreboka. I tillegg må easyIO være installert på datamaskinen (mer om dette kommer på bloggen). For info om andre muligheter for formatert utskrift med easyIO se oversikten på side 52. Ledeteksten `"Tall: "` kan skrives ut vha. en egen `skjerm.out("Tall: ")`-setning som du kan plassere før de ovennevnte `skjerm.outln`-setningene.

Oppgave: Bruk dette til å endre programmet fra forrige oppgave (i punkt 3. ovenfor) til å skrive ut produktet med 2 desimaler, og igjen med 3 desimaler på 10 plasser.

5. **Beregning av skatt i Ruritania:** *kap. 4, oppg. 2 (side 82)*
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 lese inntekten (som antas å være et desimaltall) fra terminal.

6. **For-løkke:** *kap. 4, oppg. 4 (side 83)*
   Lag et program som skriver ut på skjermen omkretsene til sirkler med radiusene r = 1, 2, ..., 10 (omkretsen *O* beregnes etter formelen $O = 2\pi r$. Sett $\pi = 3.14$). Utskriften skal følge mønsteret:

```
Radius = 1 gir omkrets = 6.28
Radius = 2 gir omkrets = 12.57
..osv..
```

7. **While-løkke:** *: kap. 4, oppg. 5 (side 83)*
   (a) Gjenta forrige oppgave, men bruk while-løkke i stedet.

   (b) Som forrige oppgave, men utskriften skal nå først stoppe når omkretsen overstiger 1000. Tips: Bruk `while`-løkken til å kontrollere *O*.

8. **Løkker: Hva blir skrevet ut?**
   Anta at følgende programsetninger utføres. Hva skrives ut på skjermen?

```
class Ukeoppg2_8 {
    public static void main(String[] args) {

//(a)
       int a = 1;
       while (a < 5) {
           a = a + 1;
       }
       System.out.println("a = " + a);

//(b)
       int b = 11;
       while (b < 14) {
           b++;
           System.out.println(b);
       }

//(c)
       int c = 1;
       while (c < 10) {
           c = -2 * c;
       }
       System.out.println("c = " + c);

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

//(e)
       for (int e = 1; e <= 3; e++) {
           for (int f = 1; f <= 2; f++) {
               System.out.println(e + " " + f);
           }
       }

//(f)
       for (int ytre = 0; ytre < 2; ytre++) {
           System.out.print("[");

           for (int indre = 0; indre < 3; indre++) {
               System.out.print(".");
           }
           System.out.println("]");
       }
}}
```

9. **En enkel array:**

```
int[] a = new int[20];
```

(a) Når setningen over utføres, skjer det både en deklarasjon og en oppretting av et array-objekt. Forklar hvilken del av setningen som gjør hva, og vis hvordan setningen kunne vært splittet opp i to setninger: en deklarasjonssetning og en setning som oppretter array-objektet.

(b) Her er eksempler på bruk av ovennevnte array. Hva blir skrevet ut på skjermen?

```
int[] a = new int[20];
a[0] = 100;
a[1] = a[0] * 2;
System.out.println(a[1]);
a[0]++;
System.out.println(a[0]);
System.out.println(a.length);
```

Oppgaver til terminaltimen

1. **Innlesing fra terminal:** *kap. 3, oppg. 3 (side 70)*
(Oppgaveteksten står i punkt 3. ovenfor)

2. **Utskrift med 2 desimaler på to måter: easyIO og printf**
(Oppgaveteksten står i punkt 4. ovenfor)

3. **Valuta-omregning:** *kap. 3, oppg. 4 (side 70)*
Lag et program som leser inn et kronebeløp **fra tastaturet**, og omregner beløpet til amerikanske dollar, euro, og svenske kroner; og skriver ut resultatet på skjermen **med to desimaler**. Omregningen er etter følgende kurser: 1 USD = 6.127 kr; 1 EUR = 7.865 kr; 1 SEK = 0.8436 kr. (*NB! Vekslingskursene er oppdatert 6. sep 2010, det står andre kurs og valutaer i læreboka.*) [Valutakurser]. ***Hint:*** Hvis brukeren taster 100 skal resultatet i USD bli $16.32.

4. **If-else med logisk uttrykk:** *kap. 4, oppg. 1 (side 82)*
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 lese personens alder fra terminal.

Hint: Se eksemplet på side 76, men legg til utskrift av en ledetekst (forklart i punkt 5. over). I denne oppgaven trenger du også logiske uttrykk (side 39).

5. **Summerings-løkke:** *kap. 4, oppg. 6 (side 83)*
(a) Skriv et program som leser ett heltall n fra terminal, og som deretter summerer tallene fra 1 til n, dvs. $1 + 2 + \cdots + n$, og skriver ut hver av mellomresultatene (1; 1+2; 1+2+3; osv.), og sluttsummen. Utskriften skal følge dette mønsteret (når $n = 5$):

```
1    1
2    3
3    6
4   10
5   15
```

(b) Det finnes også en formel som gir summen av tallene 1, 2, 3, ..., n direkte: $n * (n + 1) / 2$. Utvid programmet slik at det til slutt sammenligner siste sum med resultatet av formelen, og gir en melding til bruker på om de to svarene var like eller ikke.

6. **Løkker: Hva blir skrevet ut?**
(Oppgaveteksten står i punkt 8. ovenfor)

7. Gjør ferdig Oblig 1.

8. **En enkel array:**
(Oppgaveteksten står i punkt 9. ovenfor). Mer om arrayer neste uke...

9. **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*).

Løsningsforslag

Her kan du finne løsningsforslag til disse oppgavene. Det anbefales å løse oppgavene på egen hånd før du studerer løsningsforslagene.

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