Ukeoppgaver 4: 14. - 20. sep (INF1000 - Høst 2011)

Metoder, flerdimensjonale arrayer *(kapittel 7.1-7.7, 5.7 i "Rett på Java"* ***3. utg.****)*

Mål
Øve på bruk av forgreninger, løkker, arrayer, og metoder.

🔑 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. **Enkel kalkulator med if-else og switch:**
   (a) Lag en kalkulator som støtter enkle regnestykker på formen: *tall operator tall*, hvor de tre elementene er adskilt med mellomrom. *Tall* er heltall, og *operator* er en av de fire regneartene: `+ - * /`. Eksempel på kjøring av programmet:

   ```
   Regnestykke: 4 + 5
   Resultat: 9
   ```

   Bruk `tast.inInt()` og `tast.inChar(" ")` for å lese tall og regneart fra tastaturet (hvis du bruker EasyIO, hvis ikke se "**Hint**" nedenfor). Mellomrommet i parentesene til `inChar` angir at den skal betrakte mellomrom som et *skilletegn* før og etter regnearten. Dette er nødvendig å si fra til `inChar`, fordi den leser inn ett tegn av gangen. Lagre tallene og operator i passende variabler, og bruk **if-else**-setninger til å velge hvilken regneart skal utføres.
   Hint *for å løse det uten EasyIO: Bruk scan.nextInt() i stedet for tast.inInt(), og scan.next().charAt(0) i stedet for tast.inChar(" ");.*

   (b) Hvordan kan programmet endres for å bruke en **switch**-setning i stedet for if-else? (*Hint:* Se eksemplet på side 80 i læreboka, og husk å ta med `break;`-setningene.)

   (c) Hva slags divisjon får vi utført?, og hvordan kan vi endre programmet for å få utført den andre typen divisjon uten å endre deklarasjonen av de to innleste heltall?

2. **Løkker: Hva blir skrevet ut?**
   Avgjør uten å bruke datamaskin hva som blir skrevet ut når følgende programsetninger utføres.

```
//(a)
        int a = 10;
        while (a < 20) {
            a += 4;
        }
        System.out.println("a = " + a);

//(b)
        int sum = 0;
        for (int b = 1; b < 6; b += 2) {
            sum += b;
        }
        System.out.println("sum = " + sum);

//(c)
        int produkt = 1;
        for (int c = 1; c < 4; c++) {
            produkt = produkt * c;
            System.out.println(produkt);
        }

//(d)
        for (int d = 3; d >= 1; d--) {
            for (int e = 1; e <= 3; e++) {
                System.out.println(d + e);
            }
        }

//(e)
        int teller = 0;
        for (int ytre = 0; ytre < 3; ytre++) {
            teller++;
            for (int indre = 0; indre < 3; indre++) {
                teller++;
            }
        }
        System.out.println(teller);
```

3. **Utskrift og sum av oddetalls-array:** *kap. 5, oppg. 1-2 (side 99)*
   (a) Skriv et program som inneholder en heltalls-array med følgende elementer: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19. Programmet skal inneholde en løkke som skriver ut indeksen og verdien for alle elementene i arrayen.

   (b) Sum av elementene i en array: Vi bruker her samme array som i forrige oppgave: Beregn summen av elementene og skriv ut resultatet.

4. **Søke etter tall i array:**
   (a) Utvid programmet fra forrige oppgave slik at det ber bruker taste inn et tall, og deretter skriver ut en beskjed om det inntastede tallet finnes eller ikke finnes i arrayen.

   (b) Legg også til en løkke som skriver ut alle verdiene i arrayen som er mindre enn det inntastede tallet.

5. **Flerdimensjonal array:**
   To-dimensjonale arrayer brukes på samme måte som en-dimensjonale, men har to sett med klammer, og gjennomgås best ved hjelp av to nestede for-løkker, som vist i følgende program:

```
import easyIO.*;
class Array2D {
   public static void main(String[] args) {

        int[][] b = new int[3][4];

        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 4; j++) {

                b[i][j] = i * j;

                System.out.println(<Hva mangler her?>);
            }
        }
   }
}
```

Arrayen kan illustreres slik:

	0	1	2	3
0			_	
1			_	
2			_	

```
KJØREEKSEMPEL:
> java Array2D
b[0][0] = 0
b[0][1] = 0
b[0][2] = 0
b[0][3] = 0
b[1][0] = 0
b[1][1] = 1
b[1][2] = 2
b[1][3] = 3
b[2][0] = 0
b[2][1] = 2
b[2][2] = 4
b[2][3] = 6
```

   (a) Hvor mange elementer er det plass til i arrayen `b` vist over?
   (b) Hva er array-indeksene til array-elementene i tredje kolonne, og hvilke verdier får disse elementene i programmet over?
   (c) Fullfør `println`-setningen i programmet ovenfor slik at det gir utskriften vist under "Kjøreeksempel".

6. **Blokker og skop:**
   Hvilke av disse programmene er lovlige?

```
class Prog1 {
    public static void main(String[] args) {
        int k = 0;
        if (k >= 0) {
            int n = k + 1;
        }
        System.out.println(n);
    }
}

class Prog2 {
    public static void main(String[] args) {
        int k = 0;
        if (k >= 0) {
            int n = k + 1;
        }
        if (k < 0) {
            System.out.println(n);
        }
    }
}

class Prog3 {
    public static void main(String[] args) {
        int k = 0;
        if (k >= 0) {
            k++;
            System.out.println(k);
        }
    }
}
```

7. **inLine(), inChar(), og ordreløkke:**
Lag et program som spør bruker etter et navn. Programmet leser navnet inn ved hjelp av `inLine()` (hvis du bruker EasyIO) eller med nextLine (hvis du bruker Scanner); og "fyller" så skjermen med navnet ved å skrive det ut 100 ganger. Videre skal programmet spørre bruker om hun vil gi et nytt navn `(j/n)`. Svaret leses nå med `.inChar("\n\r")` hvis du bruker EasyIO (for tips uten EasyIO se "Hint" nedenfor), og hvis det er `'j'` gjentas hele prosessen; hvis svaret ikke er `'j'` avsluttes programmet. Kjøreeksempel:

```
Skriv et navn: Ola Nordmann
Ola Nordmann Ola Nordmann Ola Nordmann Ola Nordmann Ola Nordmann O
la Nordmann Ola Nordmann Ola Nordmann Ola Nordmann Ola Nordmann Ol
a Nordmann Ola Nordmann Ola Nordmann Ola Nordmann Ola  ...osv...
Gi nytt navn? (j/n): j
Skriv et navn:
```

Hint: Legg merke til det som står i parentesene til `.inChar("\n\r")` ovenfor (i varianten med EasyIO). Det som man skriver i disse parentesene kalles for "skilletegn" og angir hvilke tegn som ikke skal regnes som svaret (men i stedet regnes bare som skilletegn som adskiller evt. forskjellige svar). I dette tilfellet skal programmet hoppe over evt. linjeskift (`\n`) eller vognretur (`\r`) som bruker har tastet inn, og i stedet lese inn en vanlig bokstav fra tastatur, f.eks. `'j'`. Det samme kan man også få til **uten EasyIO**, tips kommer her snart for spesielt interesserte (i løpet av 14. sep).

8. **Metoder:**
(a) Endre strukturen til programmet ditt fra oppgave nr. 7 over slik at det følger den typen programstruktur som vi skal bruke i Oblig 2 (vist i skissen nedenfor), dvs. med en liten kontrollklasse øverst, etterfulgt av en egen klasse for metodene. Hele programmet med begge klassene lagres i én fil, kalt `Navn100.java` (dvs. navnet til klassen med metoden `main()`). Lag bare én metode i hjelpeklassen, kalt `ordreløkke()`, som gjør alt som står i nr. 7 ovenfor.

```
import easyIO.*;

class Navn100 {
    public static void main(String[] args) {
        Hjelpeklasse hj = new Hjelpeklasse();
        hj.ordreløkke(); // Kjører metoden ordreløkke() i hjelpeklassen
    }
}

class Hjelpeklasse {
    // Klargjøring for innlesing/utskrift, gjelder for hele klassen:
    In tast = new In();
    Out skjerm = new Out();

    String navn;

    void ordreløkke() {
        char giNyttNavn = 'j'; // Startverdi

        while (giNyttNavn != 'n') {
            // - Be bruker taste et navn og les det inn med .inLine();
            // - Utskrift av navn 100 ganger.
            // - Spør om bruker vil "Gi nytt navn? (j/n):", og .inChar("\n\r"):


        }
    }
}
```

Mer info: Grunnen til at vi skriver programmet på denne måten med to klasser vil bli klarere når vi kommer til kapittel 8, men har sammenheng med at vi ønsker å lage gode "objektorienterte" program der vi jobber med "objekter". I denne skissen er det fem pekere til objekter: args[], hj, tast, skjerm, og navn.

(b) Flere metoder: Endre programmet slik at koden som skriver ut navnet 100 ganger flyttes til en egen metode kalt `utskrift()`. Husk å legge inn et kall på metoden på det stedet i programmet du flyttet koden fra.

(c) Inn-parameter: Endre programmet slik at det også spør bruker hvor mange ganger navnet skal skrives ut. Verdien som bruker taster skal overføres som parameter til metoden `utskrift()` som du lagde i del (b). Endre også metoden `utskrift(..)` slik at den tar imot argumentet ved hjelp av parameteren `int antall`, slik: `void utskrift(int antall) {`

9.  **Ukens nøtt 1:** *(vanskelig!)*
   Lag et program som ber om 5 tall fra bruker, og deretter finner og skriver ut hvilke tall som er gjentatt blant disse. F.eks. hvis bruker tastet inn 6 6 3 6 3, så skal programmet gi meldingen:
   `Tall som er gjentatt: 6 3` *Send løsningen din til josek* [at] *ifi.uio.no, så legger jeg den ut i løsningsforslagene!*

10.  **Ukens nøtt 2:** *(vanskelig!)*
    Lag en metode som skriver ut alle anagrammer av et ord på 4 bokstaver som ligger i en `char`-array. Anagrammene skal ha de samme 4 bokstavene, i alle mulige rekkefølger og uten å gjenta noen bokstav. For eksempel, hvis ordet er deklarert som følger, er 4 av anagrammene som vist under, og totalt 24. *Send løsningen din til josek* [at] *ifi.uio.no, så legger jeg den ut i løsningsforslagene!*

```
char[] ord = { 'A', 'R', 'N', 'E' };

 Kjøreeksempel:
ARNE
AREN
ANRE
ANER
...20 ord til...
```

    Tips: En måte å løse dette på er med nestede løkker som i utgangspunktet kan gå innom alle mulige kombinasjoner, inkludert `AAAA`, `AAAE`, osv. men slik at det bare blir utskrift av de med 4 forskjellige bokstaver. Bruk den tomme strengen `""` og `+` i utskriftssetningen for å konvertere `char`-ene til tekst: `System.out.println("" + ord[3] + ord[2] ···`

11. **Tips til Emacs:** *(for spesielt interesserte)*
    - **Emacs-forkortelser:** Legg til følgende fem linjer i din `~/.emacs`-fil:

```
(define-abbrev-table 'java-mode-abbrev-table '(
    ("psv" "public static void main(String[] args) {" nil 0)
    ("sop" "System.out.println" nil 0)
))
(abbrev-mode 1)
```

      Deretter starter du Emacs på nytt. Når du skriver `psv` i et Java-program (etterfulgt av mellomrom eller linjeskift) så vil det nå bli utvidet til: `public static void main(String[] args) {`, og tilsvarende for `sop` etterfulgt av `(`. Legg gjerne til flere forkortelser. Mer info om filen `~/.emacs` kan du finne i Ukeoppgaver 2.
    - **Undo:** For å angre siste redigering trykk *C-_* (dvs. *Ctrl-understrek*), eller klikk på ikonet med bilde av en «bøyd pil» øverst i Emacs-vinduet.
    - **Copy/paste:** For å kopiere tekst fra et hvilket som helst sted på skjermen til Emacs, start med å markere teksten ved hjelp av musa. Deretter flytter du mus-pekeren til det stedet i Emacs-vinduet der du vil lime inn teksten, og trykker musens midt-knapp (dvs. hjul-tasten) rett ned. Ferdig! Du trenger altså ikke trykke *Ctrl-c* eller *høyreklikk > Copy* for å velge teksten i Emacs, det er nok å markere det.
    - **Cut/paste:** Hvis du vil klippe bort tekst og flytte det til et annet sted i Emacs-vinduet: markér teksten; trykk *Delete*-tasten eller *C-w* for å klippe det bort; flytt tekst-markøren til ønsket sted; og trykk *Insert*-tasten eller *C-y* for å lime inn.
    - **Splittet vindu:** For å kunne se to filer samtidig kan du dele Emacs-vinduet i to ved å trykke *C-x 2* ("*C-*" står for *Ctrl*-tasten). For å gå tilbake til å vise én fil klikk med musa på ønsket del av splitt-vinduet og trykk *C-x 1*.
    - **Flere vinduer:** For å åpne et ekstra-Emacs-vindu slik at du kan se to filer samtidig enda lettere trykk *C-x 52*. Husk *C-x C-f* for å åpne en fil i det nye vinduet.
    - **Innrykk:** Du kan la Emacs sette riktig innrykk i hele programmet ved å trykke *C-x h* og deretter velge i menyene øverst: *Java > Indent Line or Region*.
    - Flere tips til Emacs kan du finne i Emacs-oppgavene fra Forkurs i informatikk

---

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