

**Skriftlig eksamen i  
Programmering og Udvidet Programmering**

(015111E og 015121E)

KVL, 22. december 2000

*Alle hjælpemidler tilladt, dog ikke datamat.*

*Opgavesættet består af tre opgaver der alle ønskes løst. De tre opgaver vægtes lige.*

*For studerende i Programmering gives eksamenskarakteren på grundlag af besvarelsen af disse tre opgaver.*

*For studerende i Udvidet Programmering gives eksamenskarakteren på grundlag af besvarelsen af disse tre opgaver samt besvarelsen af kursets rapportopgave.*

*I besvarelsen må du gerne benytte klasser og metoder fra bogen og noterne uden at skrive dem af, men du skal give en præcis henvisning med afsnitsnummer eller sidetal.*

**Opgave 1**

Denne opgave handler om en applet der er en elektronisk julekalender. Når appletten startes ser den således ud:

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
<input type="text"/>					

Ved tryk på en af knapperne 1–24 fremkommer et (jule-relateret) ord i tekstfeltet for neden. Applettens programtekst har følgende hovedindhold:

```
import java.applet.Applet; import java.awt.*; import java.awt.event.*;

public class JuleApplet extends Applet {
    String tekster[] = {"Juletræ", "Pebernød", "Kræmmerhus", "Julelys",
        ... I alt 24 tekster i denne tabel ...
        "Julestjerne", "Krybbespil", "Julemand", "Klejne"};
    TextField tekstfelt = new TextField(30);

    class KnapLytter implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String label=((Button)e.getSource()).getLabel();
            tekstfelt.setText(tekster[Integer.parseInt(label)-1]);
        }
    }

    public void init() {
        KnapLytter lytter = new KnapLytter();
        /* A */
    }
}
```

**Opgave 1.1**

Vis hvordan JuleApplet kaldes fra et HTML-dokument. Appletten skal være 200 pixels høj og 200 pixels bred.

**Opgave 1.2**

Skriv den programtekst der skal indsættes ved kommentaren `/* A */` for at opnå at appletten får det udseende, der er vist oven for. Der skal bruges et Panel, 24 Button-objekter, et BorderLayout og et GridLayout. Sørg endvidere for at lytter-objektet lytter knyttes til alle knapperne.

**Opgave 1.3**

For et `ActionEvent` objekt returnerer metoden `getSource()` en reference til det `Object`, som er den AWT-komponent der genererede hændelsen. Hvis der trykkes på knappen mærket "3", hvilken værdi får variabelen `label` i metoden `actionPerformed`, og hvilken værdi bruges som indeks i `tekster` i det efterfølgende kald af `setText`?

**Opgave 1.4**

Nu skal appletten ændres således at de 24 tegnstreng i tabellen `tekster` forekommer i forskellig rækkefølge hver gang appletten køres, så man får en julekalender der er forskellig fra gang til gang. Som det allerførste i metoden `init` tilføjes følgende linie:

```
bland(tekster, 50);
```

Skriv metoden `void bland(String [] tabel, int antal)`, der antal gange ombytter to tilfældigt valgte elementer i `tabel`. Det tæller også som en ombytning selv om et element skulle blive ombyttet med sig selv.

## Opgave 2

Denne opgave handler om et program der tæller ord og sætninger i en tekstfil. Givet at tekstfilen indeholder følgende:

```
Denne fil indeholder nogle sætninger. Sætninger afsluttes med punktum.
Det er ikke sikkert, at en sætning er mindre end en linie lang, det
kan sagtens forekomme at den er længere.
```

Skal programmet udskrive følgende:

```
Der var 3 sætninger
Den gennemsnitlige sætningslængde var 10.0 ord
```

Betragt programteksten neden for:

```
import java.io.*;

public class TaelSaetninger {

    public static void main(String[] args)
        throws FileNotFoundException, IOException
    {
        Reader inp = new FileReader("input.txt");
        StreamTokenizer tstream = new StreamTokenizer(inp);
        tstream.wordChars(',',',',',','); tstream.wordChars(' ',' ',' ');
        tstream.wordChars(':',':',':','); tstream.wordChars('.',',','.');
        int saetninger=0, ord_ialt=0;
        /* A */
        System.out.println("Der var " + saetninger + " sætninger");
        System.out.println("Den gennemsnitlige sætningslængde var " +
            (double)ord_ialt/saetninger + " ord");
        /* B */
    }
}
```

Programmet åbner tekstfilen "input.txt" og opretter et StreamTokenizer objekt tstream for at kunne læse filens indhold ord for ord. StreamTokenizer er gennemgået ved forelæsning 8. I programmet ovenfor initialiseres tstream således at kommaer, semikoloner, koloner og punktummer opfattes som almindelige bogstaver og derfor under indlæsningen klistres bag på de ord, de afslutter. For eksempel vil teksten "Alle, der kunne, løste opgaven." blive indlæst som følgende brikker:

Alle,	der	kunne,	løste	opgaven.
-------	-----	--------	-------	----------

Ved kommentaren /\* A \*/ er StreamTokenizer objektet tstream klar til indlæsning og der skal i opgaverne neden for indsættes programtekst således at programmet læser filens indhold og optæller ord og sætninger. Når hele tekstfilen er læst, efter kommentaren /\* A \*/, udskriver programmet antallet af sætninger samt det gennemsnitlige antal ord i filen.

### Opgave 2.1

Angiv hvad den første linie i main metoden (Reader inp = ...) skal erstattes med for at brugeren kan angive et filnavn på kommandolinien. Hvis brugeren angiver et filnavn læses teksten fra denne fil, men hvis brugeren ikke angiver noget filnavn læses fra filen input.txt.

### Opgave 2.2

Skriv den programtekst der skal erstatte kommentaren /\* A \*/ for at programmet læser tekstfilen igennem og optæller antallet af sætninger i variabelen saetninger og antallet af ord i variabelen ord\_ialt. Du kan gå ud fra at tekstfilen indeholder en velformet tekst, således at hver brik fra tstream svarer til netop ét ord, og at en brik, der svarer til det sidste ord i en sætning, afsluttes med et punktum.

**Opgave 2.3**

Nu ønskes programmet udvidet så det også udskriver længden af den længste sætning i tekstfilen. For eksemplet ovenfor skal det som sidste linie udskrives:

```
Den længste sætning indeholdt 21 ord
```

Kommentaren `/* B */` erstattes med programlinien

```
System.out.println("Den længste sætning indeholdt " + laengste + " ord");
```

Skriv hele den programtekst der skal erstatte kommentaren `/* A */`, for at programmet ud over at fungere som i opgave 2.2 også gemmer længden af den længste sætning i variabelen `laengste`. Husk at erklære de nødvendige ekstra variable, inklusiv variabelen `laengste`.



**Opgave 3.3**

Nu tilføjes følgende abstrakte metode til klassen `Brik`:

```
abstract public String toString();
```

Ideen er at når man kalder metoden for en given brik får man en tegnstring der beskriver brikkens farve og type, for eksempel "hvid bonde", "sort bonde" eller "hvidt tårn".

Skriv `toString` metoden for klassen `Taarn`.

**Opgave 3.4**

Antag at i skakprogrammet er skakbrættet defineret som en todimensionel tabel som følger:

```
public static Brik braet[][] = new Brik[8][8];
```

Feltet med række nummer `r` og søjle nummer `s` findes i `braet[r-1][s-1]`. Tomme felter uden en skakbrik indeholder værdien `null`, mens felter, hvor der står en brik, indeholder en reference til et objekt af en af `Brik`'s konkrete subclasser.

Skriv en metode `public static void udskrivStilling()` der med `System.out.println` udskriver en line for hver brik, som findes på brættet. Linien skal indeholde feltets betegnelse efterfulgt af et kolon og beskrivelsen af den brik, der står i feltet. Beskrivelsen af en brik fås fra brikkens `toString` metoden som defineret i opgave 3.3. Et felt betegnes ved søjle og række, hvor søjlenummeret dog oversættes til et bogstav fra A til H, således at søjle 1 svarer til A, søjle 2 til B og så videre. For et hvidt tårn i feltet i søjle 5, række 4 skal der for eksempel udskrives følgende linie:

```
E4: hvidt tårn
```