

**Skriftlig eksamen i  
Programmering og Udvidet Programmering**

KVL, 22. december 1999

*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 kurssets 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 reservation af sæder i et fly. En flyvning beskrives som et objekt af klassen Fly, vist nedenfor. Et Fly-objekt har to felter, sæde og m. Tabellen sæde beskriver hvilke sæder der er optaget. Husk at elementerne i en boolean-tabel er false fra begyndelsen. Rækker og sæder er nummereret fra 1 og opefter. Der er altid mindst 1 række, og mindst 1 sæde i hver række. Ethvert fly har en midtergang, mellem sæde m og sæde m+1.

```
class Fly {
    boolean[][] sæde;      // sæde[r-1][s-1]==true hvis række r sæde s er optaget
    int m;                 // midtergangen er mellem sæde m og m+1

    static String sædenavn = "ABCDEFGHGIJK";

    public Fly(int antalrækker, int antalsæder, int m) {
        sæde = new boolean[antalrækker][antalsæder];
        this.m = m;
    }

    boolean ledig(int r, int s) // Undersøg om række r sæde s er optaget
    { return sæde[r-1][s-1]; }

    void reserver(int r, int s) // Reservér række r sæde s
    { sæde[r-1][s-1] = true; }

    void print() {
        for (int s=0; s<sæde[0].length; s=s+1) {
            if (s==m)
                System.out.print(" ");
            System.out.print(sædenavn.charAt(s) + " ");
        }
        System.out.println();
        for (int r=0; r<sæde.length; r=r+1) {
            for (int s=0; s<sæde[r].length; s=s+1) {
                if (s==m)
                    System.out.print(" ");
                if (sæde[r][s])
                    System.out.print("x ");
                else
                    System.out.print(". ");
            }
            System.out.println(" " + Integer.toString(r+1));
        }
    }
}
```

Her er et eksempel på et program der benytter ovenstående Fly-klasse:

```
public class Opgave991201 {
    public static void main(String[] args) {
        Fly SK117 = new Fly(7, 4, 2);
        SK117.reserver(1, 1);
        SK117.reserver(2, 1);
        SK117.reserver(2, 2);
        SK117.reserver(3, 4);
        SK117.reserver(4, 4);
        SK117.print();
    }
}
```

Når eksempelprogrammet køres udskriver det følgende:

```
A B C D
x . . . 1
x x . . 2
. . . x 3
. . . x 4
. . . . 5
. . . . 6
. . . . 7
```

### Opgave 1.1

Her er et andet lille program der benytter ovenstående Fly-klasse:

```
class Opgave991202 {
    public static void main(String[] args) {
        Fly SK4242 = new Fly(4, 3, 1);
        SK4242.reserver(2, 1); SK4242.reserver(2, 2); SK4242.reserver(2, 3);
        SK4242.print();
    }
}
```

Hvad udskriver dette program når det køres?

### Opgave 1.2

Tilføj en metode

```
void tjekOgReserver(int r, int s) { ... }
```

til klassen Fly. Når *f* er en variabel af type Fly skal kaldet *f.tjekOgReserver(r, s)* reservere række *r* sæde *s* hvis denne plads er ledig, og ellers rejse undtagelsen ('exception') *Exception*.

### Opgave 1.3

Tilføj en metode

```
int antalledige() { ... }
```

til klassen Fly. Når *f* er en variabel af type Fly skal kaldet *f.antalledige()* returnere antallet af ledige pladser i flyet *f*.

## Opgave 2

Denne opgave handler om tabeller af heltal. I delopgaverne 2.1 og 2.2 er tallene i tabellerne eksamenskarakterer efter 13-skalaen, dvs. kun tallene 0, 3, 5, 6, 7, 8, 9, 10, 11, 13 kan forekomme. I delopgave 2.3 kan vilkårlige heltal forekomme i tabellen.

Lad eksempeltabellerne `eks1`, `eks2` og `eks3` være erklæret og initialiseret således:

```
int[] eks1 = { 5, 6, 6, 7, 8, 8, 9, 9, 10, 13 };
int[] eks2 = { 10, 10, 10, 10 };
int[] eks3 = { };
```

### Opgave 2.1

Skriv en metode `static boolean alleEns(int[] a)` som returnerer `true` hvis alle elementer i `a` er ens, og returnerer `false` hvis der er to forskellige elementer i `a`.

For eksempeltabellerne vist ovenfor skal der gælde `alleEns(eks1)` er `false`, mens `alleEns(eks2)` og `alleEns(eks3)` er `true`.

### Opgave 2.2

Skriv en metode `static int[] forekomster(int[] a)` der som argument tager en tabel `a`, der ikke nødvendigvis er sorteret. Metoden skal returnere en heltalstabel, sådan at hvis `int[] antal = forekomster(a)`, så er `antal[k]` antallet af forekomster af karakteren `k` i `a`. Bemærk at eftersom 1, 2, 4 og 12 ikke er gyldige karakterer, så vil `antal[1]`, `antal[2]`, `antal[4]` og `antal[12]` altid være nul.

For eksempeltabellerne vist ovenfor skal der for eksempel gælde at

- `forekomster(eks1)` returnerer `{ 0, 0, 0, 0, 0, 1, 2, 1, 2, 2, 1, 0, 0, 1 }`
- `forekomster(eks2)` returnerer `{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0 }`
- `forekomster(eks3)` returnerer `{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }`
- hvis `int[] eks4 = { 7, 5, 3, 5 }` så gælder at `forekomster(eks4)` returnerer `{ 0, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0 }`.

### Opgave 2.3

Skriv en metode `static int antalforskellige(int[] a)` der som argument tager en sorteret tabel `a` og returnerer antallet af forskellige elementer i `a`. *I denne delopgave kan du ikke længere regne med at tallene i `a` er karakterer. Vilkårlige heltal kan altså forekomme i tabellen `a`. Til gengæld kan du udnytte at tabellen er sorteret.*

For eksempeltabellerne vist ovenfor skal der gælde at `antalforskellige(eks1)` returnerer 7, mens `antalforskellige(eks2)` returnerer 1, og `antalforskellige(eks3)` returnerer 0.

### Opgave 3

Denne opgave handler om en grafisk applet til at optælle karakterhyppigheder, dvs. antal forekomster af karaktererne. Appletten er vist i figuren nedenfor.

1	
00	0
03	0
5	2
6	1
7	5
8	7
9	4
10	5
11	1
13	2

Appletten har øverst et tekstfelt hvor man kan indtaste en karakter (et af tallene 0, 3, 5, 6, 7, 8, 9, 10, 11, 13), og under tekstfeltet vises hyppighederne af hver af de ti karakterer, i hver sit tekstfelt. Billedet af appletten er altså taget på et tidspunkt hvor karaktererne 00 og 03 endnu ikke er forekommet, karakteren 5 er forekommet 2 gange, karakteren 6 er forekommet 1 gang, karakteren 7 er forekommet 5 gange, osv.

I delopgave 3.1 og 3.2 skal du udfylde de dele der mangler ved punkt A og B i nedenstående skelet til en applet:

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

public class KarakterApplet extends Applet {
    ... punkt A: erklæringer af tekstfelter og lyttere osv ...
    public void init() {
        ... punkt B: erklæringer og ordrer til opsætning ...
    }
    void optael(int k) { ... denne del skal du ikke lave ... }
}
```

#### Opgave 3.1

Dette delspørgsmål handler udelukkende om applettens layout. Applettens layout er opnået med en kombination af BorderLayout og GridLayout.

Vis på en tegning hvilke dele layoutet består af.

Skriv derefter erklæringer (ved punkt A i applet-klassen) samt erklæringer og ordrer (ved punkt B) i metoden `public void init()` som kan frembringe dette layout.

#### Opgave 3.2

Dette delspørgsmål handler udelukkende om applettens virkemåde. Appletten skal virke sådan at når man har indtastet en karakter (f.eks. 7) i det øverste tekstfelt, og derefter trykker retur, så opdateres det af de nedre tekstfelter der svarer til den indtastede karakter (i dette tilfælde tekstfeltet ud for etiketten 7).

Skriv de erklæringer og ordrer (ved punkt A og B) i appletten der er nødvendige for at opnå denne virkemåde.

Du kan antage at der allerede i appletten er givet en metode `void optael(int k)` som kan kaldes med en karakter `k` og som vil lægge én til det af de nedre tekstfelter der svarer til `k`. Du behøver ikke selv definere `optael` men kan uden videre bruge metoden i din løsning.