Skriftlig eksamen i Grundlæggende Programmering

ITU, 20. januar 2000

Alle hjælpemidler tilladt, dog ikke datamat.

Eksamen er skriftlig, fire timer, og bedømmes efter 13-skalaen.

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

Hvis du i besvarelsen benytter klasser og metoder fra bogen og noterne, så behøver du ikke at skrive dem af, men du skal give en præcis henvisning med afsnitsnummer eller sidetal.

Et godt råd: Gennemlæs opgavesættet inden du begynder at besvare de enkelte opgaver.

Opgave 1

Opgave 1.1

Vis hvad dette Java-program udskriver på skærmen når det køres:

```
class Opgave1_1 {
  public static void main(String[] args) {
    int sum = 0;
    for (int i = 0; i < 6; i = i+1) {
      sum = sum + (i * i);
      System.out.println(sum);
    }
  }
}
```

Opgave 1.2

Vis hvad dette Java-program udskriver på skærmen når det køres:

```
class Opgave1_2 {
  public static void main(String[] args) {
    for (int i=0; i<4; i = i+1) {
      for (int j=0; j<4; j = j+1)
        switch ((i + j) % 2) {
        case 0: System.out.print("o"); break;
        case 1: System.out.print("x"); break;
        }
      System.out.println();
    }
  }
}
```

Husk at n `%  2` er resten ved division med 2, dvs. 0 når n er et lige tal og 1 når n er et ulige tal. For eksempel er `3  %  2` lig med 1, og `4  %  2` er lig med 0.

Opgave 1.3

Skriv et komplet Java-program som udskriver disse seks linier på skærmen når det køres:

```
*          *
 *        *
  *      *
   *    *
    *  *
     **
```

Den første linie har altså en stjerne helt til venstre på linien, efterfulgt af ti mellemrum og en stjerne. Den anden linie har ét mellemrum, en stjerne, otte mellemrum og en stjerne. Den tredje linie har to mellemrum, en stjerne, seks mellemrum og stjerne. Og så fremdeles. Den sidste linie har ikke noget mellemrum mellem stjernerne.

Opgave 1.4

Skriv en metode `static void kile(int n)` der kan udskrive en kile-formet figur som vist i opgaven ovenfor. Parameteren `n` skal angive hvor mange linier figuren består af. For eksempel skal kaldet `kile(6)` udskrive præcis den figur der er vist ovenfor.

Opgave 2

Denne opgave handler om et program til simpel databehandling af nogle badevandstemperaturer. Badevandstemperaturerne er aflæst en gang om ugen og lagret i en heltalstabel `basetemp`:

```
class Badevand {
  public static void main(String[] args) {
    int[] badetemp = { 12, 10, 12, 14, 15, 16, 18, 18, 15 };
    System.out.println("Maksimum: " + maks(badetemp));
    System.out.println("Antal fald: " + antalfald(badetemp));
  }

  static int maks(int[] temp) {
    ...
  }

  static int antalfald(int[] temp) {
    ...
  }
}
```

I delopgaverne 2.1 og 2.2 skal du færdiggøre metoderne `maks` og `antalfald` sådan at når man kører ovenstående program med kommandoen

```
java Badevand
```

så giver det denne udskrift:

```
Maksimum: 18
Antal fald: 2
```

Opgave 2.1

Færdiggør metoden `maks` sådan at `maks(temp)` returnerer den største værdi i tabellen `temp`.

Du kan gå ud fra at alle tal i `temp` er større end eller lig med nul. Hvis tabellen `temp` er tom, så skal metoden returnere 0.

Opgave 2.2

Færdiggør metoden `antalfald` sådan at `antalfald(temp)` returnerer antallet af fald i tabellen `temp`, dvs. antallet af gange hvor et mindre tal følger lige efter et større tal i `temp`. (Det er antallet af gange hvor badevandstemperaturen er faldet fra en uge til den næste).

Opgave 2.3

Nu skal du ændre metoden `main` i programmet `Badevand`, sådan at indholdet af tabellen `badetemp` tages fra kommandolinien. Eksempel: Hvis man kører det ændrede program med kommandoen

```
java Badevand 10 12 8 11
```

så skal det give denne udskrift:

```
Maksimum: 12
Antal fald: 1
```

Vis hvordan den ændrede metode `main` skal se ud.

Opgave 3

Denne opgave handler om beregning af ejendomsvurdering (af hensyn til skattefastsættelse). En ejendom består af en grund og en bygning, og er beskrevet som et objekt af klassen Ejendom vist her:

```
class Ejendom {
  Bygning bygning;
  Grund grund;

  Ejendom(Bygning bygning, Grund grund)
  { this.bygning = bygning; this.grund = grund; }

  int vurdering()
  { return bygning.vurdering() + grund.vurdering(); }
}

class Bygning { ... }

class Grund { ... }

class Vurdering {
  public static void main(String[] args) {
    Ejendom k29 =
      new Ejendom(new Bygning(200, 8000), new Grund(800, 500, 400000));
    Ejendom a11 =
      new Ejendom(new Bygning( 70, 4000), new Grund(700,  30,  24000));
    System.out.println(k29.vurdering());
    System.out.println(a11.vurdering());
  }
}
```

Som det ses indeholder et Ejendoms-objekt henvisninger til et Grund-objekt og et Bygnings-objekt. Desuden er der en metode `vurdering` som beregner ejendommens vurdering som summen af grundens vurdering og bygningens vurdering. I delopgaverne 3.1 og 3.2 nedenfor skal du erklære klasserne Bygning og Grund. I delopgave 3.3 skal du definere yderligere en metode (ikke vist eller benyttet ovenfor).

Opgave 3.1

Skriv klassen Bygning. En Bygning er kendetegnet ved sit areal (f.eks. 200 kvadratmeter) og en kvadratmeterpris (f.eks. 8000 kroner pr. kvadratmeter); begge er heltal. Klassen Bygning skal have en konstruktor der kan kaldes som vist i metoden `main` ovenfor. Desuden skal klassen have en metode `int vurdering()` der beregner og returnerer bygningens vurdering (ved at gange arealet med kvadratmeterprisen).

Opgave 3.2

Skriv klassen Grund. En Grund er kendetegnet ved sit areal (f.eks. 800 kvadratmeter) og en kvadratmeterpris (f.eks. 500 kroner pr. kvadratmeter); begge er heltal. Desuden er der en byggeret (f.eks. 400000 kroner); det er et heltal som angiver værdien af retten til at have en bygning på grunden. Klassen Grund skal have en konstruktor der kan kaldes som vist i metoden `main` ovenfor. Desuden skal klassen have en metode `int vurdering()` der beregner og returnerer grundens vurdering (ved at gange arealet med kvadratmeterprisen og lægge byggeretten til).

Opgave 3.3

Skriv en metode `static int vurdering(Ejendom[] ejd)` der som argument tager en tabel af ejendomme, og beregner og returnerer summen af ejendommenes vurderinger.

Eksempel: Metoden skal kunne kaldes som vist i de to linier, der her er føjet til slutningen af den tidligere viste `main`-metode i klassen Vurdering:

```
class Vurdering {
  public static void main(String[] args) {
    Ejendom k29 = new Ejendom( ... );
    Ejendom a11 = new Ejendom( ... );
    ...
    Ejendom[] ejendomme = { k29, a11 };
    System.out.println(vurdering(ejendomme));
  }
}
```

Opgave 3.4

Hvis en grund er forurenet gives der et fradrag i vurderingen (fordi en forurenet grund er sværere at sælge).

Skriv derfor en subklasse ForurenetGrund af Grund:

- Subklassen skal have et nyt felt `fradrag` som er et positivt heltal der angiver fradraget. Feltet skal være privat.
- Subklassen skal have en konstruktor der tager de samme argumenter som i klassen Grund, samt fradraget.
- Subklassen skal overskrive (omdefinere) metoden `vurdering` således at der tages hensyn til fradraget.
- Subklassen skal have en ny metode `void sætFradrag(int fradrag)` så man kan ændre fradraget.

Opgave 4

Denne opgave handler om appletten, hvis programtekst er vist her:

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

public class Opgave2000Applet extends Applet {
  TextField ind = new TextField(20);
  Button lægtil = new Button("Læg til");
  TextArea ud   = new TextArea(5, 30);
  int sum = 0;
  /* A */

  public void init() {
    Panel p = new Panel();
    p.setLayout(new BorderLayout());
    p.add(ind, "West");
    /* B */
    Panel pe = new Panel();
    pe.add(lægtil);
    p.add(pe, "East");
    p.add(ud, "South");
    add(p);
    lægtil.addActionListener(new KnapLytter());
    /* C */
  }

  class KnapLytter implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      int v = Integer.parseInt(ind.getText());
      sum = sum + v;
      ud.append("Summen er nu " + sum + "\n");
    }
  }
  /* D */
}
```

Kommentarerne `/* A */`, `/* B */` osv. har ingen effekt; de er kun indsat for at navngive programpunkter af hensyn til opgave 4.3.

Opgave 4.1

Dette delspørgsmål handler udelukkende om applettens udseende (layout). Vis med en tegning hvordan appletten ser ud umiddelbart efter den er startet. Forklar hvilke komponenter (paneler, tekstfelter, knapper, tekstområder, ...) der sidder hvor på appletten. Komponenternes placering er vigtig, men deres nøjagtige størrelse er ikke vigtig.

Opgave 4.2

Dette delspørgsmål handler udelukkende om applettens virkemåde. Antag at man starter appletten, taster 12 i tekstfeltet `ind`, trykker på knappen `lægtil`, taster 8 i tekstfeltet `ind`, og trykker på knappen `lægtil`. Hvad står der så i tekstområdet `ud`?

Opgave 4.3

I dette delspørgsmål skal du tilføje en ny knap mærket Nulstil til appletten. På appletten skal knappen sidde mellem tekstfeltet `ind` og knappen `lægtil`. Når der trykkes på knappen mærket Nulstil, så skal variablen `sum` sættes til 0, og i tekstområdet `ud` skal der tilføjes en ny linie med teksten `Sum nulstillet`.

Angiv alle de ændringer til den givne Java-kode som er nødvendige for at opnå denne effekt. Der skal angives præcis hvilke nye erklæringer og ordrer som skal tilføjes, og hvor de skal tilføjes. Vink: Det er nyttigt at henvise til programpunkterne mærket `/* A */`, `/* B */`, osv. i programmet vist ovenfor.