

Skriftlig eksamen i Grundlæggende Programmering

IT-C, 16. juni 2000
Dansk udgave

Alle hjælpemidler er 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 besvarelsen benytter klasser og metoder fra lærebogen, forelæsningsnoterne, eller forelæsningsplanterne, så behøver du ikke at skrive dem af, men du skal give en præcis henvisning med udgave (af bogen), afsnitsnummer, side-tal, osv.

The exam questions are available also in English (separately). Eksamensspørgsmålene findes også på engelsk (på separate ark).

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=6; i>=2; i=i-1) {
            sum = sum + i;
            System.out.println(sum);
        }
        System.out.println("Gennemsnittet er: " + sum / 5);
    }
}
```

Opgave 1.2

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

```
  \
 - \
-- \
--- \
---- \
----- \
```

Den første linie har en baglæns skråstreg helt til venstre på linien. Den anden linie har en bindestreg, efterfulgt af en baglæns skråstreg. Den tredje linie har to bindestreger efterfulgt af en baglæns skråstreg. Og så fremdeles.

Opgave 1.3

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

Opgave 1.4

Skriv en metode `static void parallel(int n)` som kan udskrive figurer som denne:

```
  \
 - \
 \- \
 - \- \
 \- \- \
 - \- \- \
```

Parameteren `n` skal angive hvor mange linier figuren består af. For eksempel skal kaldet `parallel(6)` udskrive præcis den figur der er vist ovenfor. Vink: metoden `parallel` kan laves ved at modificere metoden `figur` fra opgave 1.3.

Opgave 2

Denne opgave omhandler simpel databehandling af husnumre. Husnumrene er lagret i en heltalstabel `nr_tabel`, som er sorteret stigende:

```
class Opgave2 {
    static int antal_lige(int[] t) { ... }
    static int antal_ulige(int[] t) { .. }
    static int st_mlm(int[] t) { ... }

    public static void main(String[] args) {
        int[] nr_tabel = {12, 13, 14, 17, 19, 21, 24, 26, 27};

        System.out.println("Antal lige numre: " + antal_lige(nr_tabel));
        System.out.println("Antal ulige numre: " + antal_ulige(nr_tabel));
        System.out.println("Største mellemrum: " + st_mlm(nr_tabel));
    }
}
```

I delspørgsmål 2.1 og 2.2 skal du færdiggøre metoderne `antal_lige`, `antal_ulige` og `st_mlm`. Når du kører programmet med kommandoen

```
java Opgave2
```

skal det give følgende udskrift:

```
Antal lige numre: 4
Antal ulige numre: 5
Største mellemrum: 3
```

Opgave 2.1

Færdiggør metoden `antal_lige`, således at `antal_lige(t)` returnerer antallet af lige numre i tabellen `t`. Hvis tabellen `t` er tom skal metoden returnere 0. Vink: Du kan benytte modulus-operatoren (også kaldet rest-operatoren), idet `x%2` er 0 hvis `x` er lige, og 1 hvis `x` er ulige.

Færdiggør metoden `antal_ulige`, således at `antal_ulige(t)` returnerer antallet af ulige numre i tabellen `t`. Hvis tabellen `t` er tom skal metoden returnere 0.

Opgave 2.2

Færdiggør metoden `st_mlm`, således at `st_mlm(t)` returnerer den største forskel (spring) mellem to på hinanden følgende tal in tabellen `t`. Eksempel: I tabellen `{1, 4, 5}` er forskellene henholdsvis 3 og 1, dvs. den største forskel er 3. Metoden skal returnere 0 for tabeller med 0 eller 1 element. Du kan antage at tabellen er sorteret stigende.

Opgave 2.3

To lige tal i en tabel kaldes *lige naboer* hvis de står ved siden af hinanden eller kun er adskilt af ulige tal i tabellen.

Skriv en metode `static int st_mlm_lige(int[] t)` som returnerer den største forskel mellem to lige naboer i tabellen `t`.

Eksempel: I den ovenfor viste tabel `nr_tabel` gælder der at 12 og 14 er lige naboer; at 14 og 24 er lige naboer; og at 24 og 26 er lige naboer. Altså skal `st_mlm_lige(nr_tabel)` give 10, nemlig forskellen mellem 14 og 24. Vink: det kan betale sig at lade metoden opbygge en ny tabel der kun indeholder de lige numre (stadig sorteret stigende).

Opgave 3

Denne opgave omhandler en applet med grafisk brugergrænseflade til omregning mellem danske og amerikanske eksamenskarakterer. Applettens programtekst er givet nedenfor.

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

public class Opgave3 extends Applet {
    TextArea ud = new TextArea(5,20);
    TextField ind = new TextField(10);
    Button us = new Button("US");
    /* a */

    class USlytter implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            int karakterDK = Integer.parseInt(ind.getText());
            String karakterUS = "?";
            switch (karakterDK) {
                case 13: case 11: case 10: karakterUS="A"; break;
                case 9: case 8: case 7: karakterUS="B"; break;
                case 6: karakterUS="C"; break;
                case 5: karakterUS="D"; break;
                case 3: case 0: karakterUS="F"; break;
            }
            ud.append(karakterDK + " svarer til " + karakterUS + ".\n");
        }
    }

    public void init() {
        Panel p = new Panel();
        p.setLayout(new BorderLayout());
        Panel pNorth = new Panel();
        pNorth.add(ind);
        us.addActionListener(new USlytter());
        pNorth.add(us);
        /* b */
        p.add(pNorth, "North");
        /* c */
        p.add(ud, "Center");
        /* e */
        add(p);
    }
}
```

Kommentarerne `/* a */`, `/* b */` osv. har ingen effekt; de er kun indsat for at navngive programpunkter af hensyn til delspørgsmål 3.3 og 3.4.

Opgave 3.1

Dette delspørgsmål omhandler udelukkende applettens udseende (layout). Lav en tegning som viser hvorledes appletten ser ud umiddelbart efter den er startet. Af tegningen skal fremgå hvilke komponenter (paneler, tekstfelter, tekstområder, knapper, ...) der er anvendt. Tegningen skal vise komponenternes placering i forhold til hinanden, men komponenternes nøjagtige størrelse er ikke vigtig.

Opgave 3.2

Dette delspørgsmål omhandler udelukkende applettens virkemåde. Antag, at man starter appletten, taster 9 i tekstfeltet `ind`, trykker på knappen `us`, taster 12 i tekstfeltet `ind`, og trykker på knappen `us`. Hvad står der så i tekstområdet `ud`?

Opgave 3.3

I dette delspørgsmål skal du tilføje en knap mærket Nulstil. Knappen skal sidde nederst på appletten, dvs. nedenunder tekstområdet. Når knappen klikkes, skal indholdet af tekstområdet ud og tekstfeltet ind slettes.

Du skal angive 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 ovenfor.

Opgave 3.4

I dette delspørgsmål skal du tilføje en knap mærket DK. Knappen skal sidde til højre for knappen us. Når knappen trykkes skal indholdet af tekstfeltet ind, der antages at være en amerikansk karakter, omregnes til en dansk karakter. Du skal anvende følgende omregningstabel:

Amerikansk karakter	Dansk karakter
A	11
B	9
C	7
D	5
F	03

Hvis der indtastes andet end A, B, C, D eller F, så skal der udskrives en passende meddelelse i tekstområdet; f.eks. "A+ svarer til ?", hvor brugeren har indtastet "A+" som ikke er omfattet af omregningstabellen ovenfor.

Du skal angive præcis hvilke nye erklæringer og ordrer som skal tilføjes, og hvor de skal tilføjes. Vink: Man kan sammenligne tegnstrengene `s1` og `s2` ved at anvende `s1.equals(s2)`.

Opgave 4

I denne opgave skal du lave et program der modellerer andelslejligheder, hvad angår antal rum, areal (i kvadratmeter) og andelsværdi per kvadratmeter. En et-værelses andelslejlighed består af et køkken og et enkelt rum, og er beskrevet af klassen `EtVærelses` vist her:

```
class EtVærelses {
    Rum rum;
    Køkken køkken;
    int andelsværdi;

    EtVærelses(Rum rum, Køkken køkken, int andelsværdi) {
        this.rum = rum;
        this.køkken = køkken;
        this.andelsværdi = andelsværdi;
    }

    public int getAreal()
    { return rum.getAreal() + køkken.getAreal(); }

    public int getAndelsværdi() { ... }
}

class Rum { ... }
class Køkken { ... }

class Opgave4 {
    public static void main(String[] args)
    { EtVærelses v1 = new EtVærelses(new Rum("Stue", 12), new Køkken(4), 2002);
      System.out.println("Areal: " + v1.getAreal());
      System.out.println("Andelsværdi: " + v1.getAndelsværdi()); }
}
```

Som det fremgår ovenfor indeholder et `EtVærelses`-objekt henvisninger til et `Køkken`-objekt og et `Rum`-objekt samt et heltal der angiver andelsværdien per kvadratmeter. Metoden `getAreal` returnerer det samlede areal for lejligheden som summen af rummets areal og køkkenets areal. Metoden `getAndelsværdi` beregner og returnerer andelsværdien for lejligheden; det er lejlighedens samlede areal ganget med andelsværdien per kvadratmeter.

I delspørgsmål 4.3 skal du definere en klasse `NVærelses` som ikke er vist ovenfor.

Opgave 4.1

Skriv klassen `Rum`. Et `Rum` er kendetegnet ved et navn (f.eks. "Stue"), som er en tegnstreng, og et areal (f.eks. 12), som er et heltal. Klassen `Rum` skal have en konstruktor der kan kaldes som vist i metoden `main` ovenfor. Derudover skal klassen have en metode `int getAreal()`, der returnerer rummets areal.

Opgave 4.2

Skriv klassen `Køkken`. Et `Køkken` er kendetegnet ved et navn som altid er tegnstrengen "Køkken" og et areal (f.eks. 4) som er et heltal. Klassen `Køkken` skal have en konstruktor, der kan kaldes som vist i metoden `main` ovenfor. Desuden skal klassen have en metode `int getAreal()`, der returnerer køkkenets areal. Vink: Det er muligt at lave klassen `Køkken` som en subklasse af klassen `Rum`, men det ikke nødvendigt.

Opgave 4.3

Skriv en klasse `NVærelses`, som svarer til klassen `EtVærelses`, bortset fra, at feltet `rum` nu har type `Rum[]` og ikke `Rum`; klassen `NVærelses` er altså kendetegnet ved et vilkårligt antal værelser udover køkkenet. Metoderne `int getAreal()` og `int getAndelsværdi()` skal tilpasses.

Eksempel: Klassen `NVærelses` skal kunne anvendes som vist i metoden `main` nedenfor:

```
class Opgave4 {
    public static void main(String[] args)
    { Rum[] r = {new Rum("Stue", 10), new Rum("Spisestue", 15),
                new Rum("Soveværelse", 12)};
      NVærelses v3 = new NVærelses(r, new Køkken(9), 2342);
      System.out.println("Areal: " + v3.getAreal());
      System.out.println("Andelsværdi: " + v3.getAndelsværdi()); }
}
```

Opgave 4.4

Det skal være muligt at lave en nedskrivning af andelsværdien på en lejlighed, hvis den er i meget dårlig stand. Skriv derfor klassen `RingeEtVærelses` som en subclasses af klassen `EtVærelses`, således at:

- Subklassen har et nyt felt `nedskriv` som er et positivt heltal. Feltet skal være privat.
- Subklassen har en konstruktør med de samme argumenter som konstruktoren i klassen `EtVærelses`, samt nedskrivningen.
- Subklassen overskriver (omdefinerer) metoden `getAndelsværdi` så den tager hensyn til nedskrivningen.

Eksempel: Klassen `RingeEtVærelses` skal kunne anvendes som vist i metoden `main` nedenfor:

```
class Opgave4 {
    public static void main(String[] args)
    { RingeEtVærelses v4 = new RingeEtVærelses(new Rum("Stue", 12),
                                              new Køkken(4), 2002, 10000);
      System.out.println("Areal: " + v4.getAreal());
      System.out.println("Andelsværdi: " + v4.getAndelsværdi()); }
}
```