

# Written exam in Introductory Programming

IT-C, June 16, 2000  
English version

*All materials are permitted during the exam, except computers.*

*The exam questions must be answered in writing within four hours. The answers will be graded using the Danish '13-grade' scale.*

*There are four main questions, all of which should be answered. The four main questions have equal weight.*

*Your answers may use classes or methods from the textbook, the lecture notes, and the lecture slides. You do not need to copy them to your answers, but you must give a precise reference, stating edition (of the textbook), section, page number etc.*

*Eksamensspørgsmålene findes også på dansk (på separate ark). The exam questions are available also in Danish (separately).*

*You are advised to read all the questions before you start answering any of them.*

## Question 1

### Question 1.1

Show what this Java-program will output when executed:

```
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);
    }
}
```

### Question 1.2

Write a complete Java program that will output these six lines when executed:

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

The first line contains a backslash at the left end of the line. The second line contains a hyphen followed by a backslash. The third line contains two hyphens followed by a backslash. And so on.

### Question 1.3

Write a method `static void figur(int n)` that can output a figure like that shown above. The parameter `n` specifies the number of lines making up the figure. For example, the call `figur(6)` must output precisely the figure shown above.

### Question 1.4

Write a method `static void parallel(int n)` that can output figures such as this one:

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

The parameter `n` specifies the number of lines making up the figure. For example, the call `parallel(6)` must output precisely the figure shown above. Hint: the method `parallel` can be created by modifying the method `figur` from question 1.3.

## Question 2

The subject of this question is house numbers, as used in street addresses. The house numbers are stored in an integer array `nr_tabel`, which is sorted in ascending order:

```
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));
    }
}
```

In the questions 2.1 and 2.2 you must complete the methods `antal_lige`, `antal_ulige`, and `st_mlm`. When you run the above program using the command

```
java Opgave2
```

then it must produce the following output.

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

Note that ‘lige’ means ‘even’, and ‘ulige’ means ‘odd’.

### Question 2.1

Complete the method `antal_lige`, so that `antal_lige(t)` returns the number of even house numbers in the array `t`. If the array `t` is empty, then the method call must return 0. Hint: You may use the modulo-operator (also called the remainder-operator), since  $x\%2$  is 0 if  $x$  is even, and 1 if  $x$  is odd.

Complete the method `antal_ulige`, so that `antal_ulige(t)` returns the number of odd house numbers in the array `t`. If the array `t` is empty, then the method must return 0.

### Question 2.2

Complete the method `st_mlm`, so that `st_mlm(t)` returns the largest difference (jump) between two adjacent numbers in the array `t`. Example: In the array  $\{1, 4, 5\}$ , the differences are 3 and 1, respectively. That is, the largest difference is 3. The method must return 0 for arrays with 0 or 1 element. You may assume that the array is sorted in ascending order.

### Question 2.3

Two even numbers in an array are called *even neighbours* if they are adjacent or separated only by odd numbers.

Write a method `static int st_mlm_lige(int[] t)` that returns the largest difference between two even neighbours in the array `t`.

Example: In the array `nr_tabel` shown above, we see that 12 and 14 are even neighbours; that 14 and 24 are even neighbours; and that 24 and 26 are even neighbours. Thus `st_mlm_lige(nr_tabel)` must return 10, namely, the difference between 14 and 24. Hint: it is useful to first build a new array that contains only the even house numbers (still sorted).

## Question 3

The subject of this question is an applet (with a graphical user interface) for conversion between Danish and American exam grades. The program text of the applet is shown below.

```
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");
        /* d */
        add(p);
    }
}
```

The program comments `/* a */`, `/* b */` etc. have no effect; they serve only to name program points for reference in questions 3.3 and 3.4.

### Question 3.1

This question is concerned only with the appearance (lay-out) of the applet.

Make a drawing that shows how the applet looks just after it has been started. The drawing must show which components (panels, text fields, text areas, buttons, ...) have been used. The drawing must show the placement of the components relative to each other, but the precise size of the components is immaterial.

### Question 3.2

This question is concerned only with the functionality of the applet. Assume that one starts the applet, types 9 in the text field `ind`, presses the button `us`, types 12 in the textfield `ind`, and presses the button `us`. Then what does the text area `ud` contain afterwards?

### Question 3.3

In this question you must add a button labelled Nulstil (meaning 'reset'). The button must be at the bottom of the applet, below the text area. When the button is pressed, the contents of the text area `ud` and the text field `ind` must be erased.

Specify precisely what new declarations and statements must be added, and where. Hint: It is useful to refer to the program points labelled `/* a */`, `/* b */`, etc. in the program above.

### Question 3.4

In this question you must add a button labelled DK. The button must be to the right of the `us` button. When the new button is pressed, then the contents of the text fields `ind`, which is supposed to be an American grade, must be converted to the corresponding Danish grade. You must use this table for conversion:

American grade	Danish grade
A	11
B	9
C	7
D	5
F	03

If anything else than A, B, C, D, or F is entered, then a suitable message must be shown in the text area, e.g. "A+ corresponds to ?", if the user had entered "A+", which is not in the table shown above.

You must specify precisely what new declarations and statements must be added, and where. Hint: You can compare the text strings `s1` and `s2` by using `s1.equals(s2)`.

## Question 4

The subject of this question is the modelling of cooperative apartments ('andelslejligheder'), as concerns the number of rooms, area ('areal', in square meters), and value ('andelsværdi') per square meter. A one-room apartment consists of a kitchen ('køkken') and a single room ('rum'), and is described by the class `EtVærelses` ('one-room') shown below:

```
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()); }
}
```

As shown above, an `EtVærelses` (one-room) object contains references to a `Køkken` (kitchen) object and a `Rum` (room) object, and an integer that specifies the value ('andelsværdi') per square meter. The method `getAreal` returns the total area of the apartment as the sum of the room's area and the kitchen's area. The method `getAndelsværdi` computes and returns the total value of the apartment; that is, the total area times the value per square meter.

In question 4.3 below you shall define a class `NVærelses` ('N rooms'), not shown above.

### Question 4.1

Declare the class `Rum`. A `Rum` (room) object is characterized by a name (e.g., "Stue"), which is a string, and an area in square meters (e.g., 12), which is an integer. The class `Rum` must have a constructor that can be called as shown in the method `main` above. Moreover, the class must have a method `int getAreal()` which returns the area of the room.

### Question 4.2

Declare the class `Køkken`. A `Køkken` (kitchen) object is characterized by a name which is always the string "Køkken", and an area (e.g., 4) which is an integer. The class `Køkken` must have a constructor that can be called as shown in the method `main` above. Moreover, the class must have a method `int getAreal()` which returns the area of the kitchen. Hint: It is possible, but not necessary, to declare `Køkken` as a subclass of class `Rum`.

### Question 4.3

Declare a new class `NVærelses` ('N rooms') corresponding to the class `EtVærelses`, except that the field `rum` now has type `Rum[]`, not `Rum`. Thus an object of class `NVærelses` is characterized by having an arbitrary number of rooms in addition to the kitchen. The methods `int getAreal()` and `int getAndelsværdi()` must be changed accordingly.

Example: It must be possible to use the class `NVærelses` as in the method `main` shown below:

```
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()); }
}
```

### Question 4.4

If an apartment is in poor shape, it should be possible to reduce its value by a fixed amount. Thus you must write a class `RingeEtVærelses` ('poor one-room') as a subclass of class `EtVærelses`, so that:

- The subclass has a new field `nedskriv` ('depreciation') which is a positive integer. The field must be private.
- The subclass has a constructor with the same arguments as the constructor in class `EtVærelses`, as well as the depreciation.
- The subclass overrides (redefines) the method `getAndelsværdi` so that it reduces the computed value by specified depreciation.

Example: It must be possible to use the class `RingeEtVærelses` as shown in the method `main` below:

```
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()); }
}
```