

**Written exam in
Introductory Programming**

IT University of Copenhagen, June 11, 2001
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 three main questions, all of which should be answered.

Question 1 counts 25 %, question 2 counts 35 %, and question 3 counts 40 %.

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 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 (25 %)

Question 1.1

Show what is printed when this Java program is run:

```
class opg1_1 {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 4; i >= 1; i=i-1) {
            sum = sum + 2*i;
            System.out.println(sum);
        }
    }
}
```

Question 1.2

Show what is printed when this Java program is run:

```
class opg1_2 {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 0; i < 6; i=i+1) {
            for (int j = 2; j < i; j=j+1) {
                sum = sum + j;
            }
            System.out.println(sum);
        }
    }
}
```

Question 1.3

Write a Java method `static void Linie(int m, int s)` which prints a single line containing `m` spaces followed by `s` asterisks, followed by a newline.

For instance, executing the two method calls `Linie(1, 3)` and `Linie(0, 5)` should produce the following printout:

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

Question 1.4

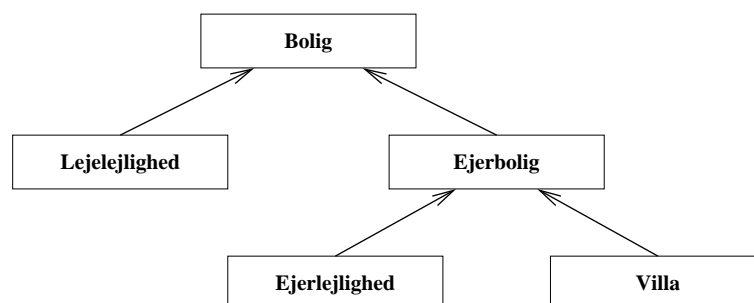
Write a Java method `static void diamant(int n)` which prints a diamond of asterisks. The diamond must consist of $2*n+1$ lines, and the longest line must contain $2*n+1$ asterisks.

For instance, the method call `stjerne(2)` should print a diamond consisting of $2*2+1 = 5$ lines, as follows:

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

Question 2 (35 %)

This question concerns a program for modelling a register of dwellings. There are several different kinds of dwellings, modelled by this class hierarchy:



Thus Lejelejlighed (rented apartment) and Ejerbolig (owned dwelling) are subclasses of the class Bolig (dwelling), and Ejerlejlighed (owned apartment) and Villa (villa) are subclasses of Ejerbolig (owned dwelling).

The class Bolig (dwelling) is defined like this (where 'areal' means area, and 'kommune' means municipality):

```

class Bolig {
    int areal;
    String kommune;

    Bolig(int areal, String kommune) {
        this.areal = areal; this.kommune = kommune;
    }

    public void print_info() {
        System.out.println("areal: " + areal);
        System.out.println("kommune: " + kommune);
    }
}
  
```

Question 2.1

An owned dwelling ('ejerbolig') is a dwelling that has a cash price ('kontantpris') which is an integer, and a monthly net payment ('nettohusleje') which is an integer. Write the class Ejerbolig as a subclass of Bolig. The class must have a constructor that initializes all the fields (area, municipality, cash price, net payment). The method `print_info` from the superclass Bolig must be overridden so that the method prints not only the area and municipality, but also the cash price and net payment.

Question 2.2

An owned apartment ('ejerlejlighed') is an owned dwelling ('ejerbolig') that may have a lift ('elevator'), denoted by the value of a boolean field `elevator`, which is `true` if there is a lift, and `false` otherwise. Write the class Ejerlejlighed as a subclass of Ejerbolig. The class must have a constructor that initializes all the fields (area, municipality, cash price, net payment, elevator). Override the method `print_info` so that all fields get printed.

Question 2.3

A rented apartment ('lejelejlighed') is a dwelling ('bolig') that has a monthly rent ('husleje') which is an integer, and that may have a lift ('elevator') denoted by the value of a boolean field `elevator`. Write the class Lejelejlighed as a subclass of Bolig. The class must have constructor that initializes all the fields (area, municipality, rent, elevator). Override the method `print_info` so that all fields get printed.

Question 2.4

Consider the following skeleton of a program whose main method creates a register of dwellings ('boligregister') with four dwellings:

```
class Boligopgave {
    public static void main(String[] args) {
        Bolig bb = new Bolig(120, "København");
        Villa bv = new Villa(200, "Gentofte", 3000000, 20000, 800);
        Ejerlejlighed be = new Ejerlejlighed(100, "Århus", 2000000, 12000, false);
        Lejelejlighed bl = new Lejelejlighed(65, "Odense", 2500, false);

        Bolig[] boligregister = {bb,bv,be,bl};

        // Print whether dwelling number 3 has a lift                // pp1
        // Print information about all dwellings in boligregister    // pp2
        // Count and print number of owned dwellings in boligregister // pp3
    }
}
```

At program point pp1 above, one wishes to print whether dwelling number 3, that is, `boligregister[3]`, has a lift ('elevator') or not. Note that `boligregister[3]` refers to an object of class `Lejelejlighed`, and that that object has an `elevator` field. Here are two proposals pp1a and pp1b for statements to be used at program point pp1:

```
System.out.println("boligregister[3].elevator: " +                // pp1a
    boligregister[3].elevator);

System.out.println("boligregister[3].elevator: " +                // pp1b
    ((Lejelejlighed)boligregister[3]).elevator);
```

Which one(s) of the above proposals pp1a and pp1b would be accepted by the Java compiler if inserted at program point pp1? Briefly give a reason for your answer.

Question 2.5

Add some program text at program point pp2 to print out information about all dwellings in the register.

Question 2.6

Add some program text at program point pp3 to count and print the number of owned dwellings in the register.

Hint: Use Java's built-in operator `instanceof`. The expression `(e instanceof K)` is true if the object `e` belongs to class `K` or a subclass of `K`, and otherwise the expression is false. Example: when `bv` refers to an object of class `Villa`, then the expressions `(bv instanceof Villa)` and `(bv instanceof Ejerbolig)` and `(bv instanceof Bolig)` are all true, but `(bv instanceof Lejelejlighed)` is false.

Question 2.7

An apartment ('lejlighed') is a dwelling that has an area ('areal') which is an integer, a municipality ('kommune'), which is a String, and possibly a lift ('elevator'), indicated by a boolean.

Write a Java interface `Lejlighed` that describes the methods `get_areal`, `get_kommune`, `get_elevator`, and `print_info`. Modify the classes `Lejelejlighed` and `Ejerlejlighed` so that they implement the interface `Lejlighed`. You need only show the changes relative to the earlier answers, and it suffices to show it for one of the two classes.

Question 2.8

Given the above changes it makes sense to extend the above main methods (after pp3) with a declaration of a variable `lejlighedsregister` (apartment register), which is an array whose elements are of type `Lejlighed`. For instance:

```
Lejlighed[] lejlighedsregister = {be,bl};
```

Show how the main method can be extended to find and print information about the apartment (object) in such an array `lejlighedsregister` that has the largest area.

Question 3 (40 %)

A road distance table may be used to give the distances between some towns on Sjælland. To simplify the programming of distance tables, we assign numbers 0, 1, ... to the towns. Then we may get a distance table like this:

Town	Number
København	0
Køge	1
Hillerød	2
Vordingborg	3
Slagelse	4
Hundested	5
Kalundborg	6

	0	1	2	3	4	5	6
0							
1	42						
2	41	72					
3	97	60	128				
4	92	56	97	64			
5	67	80	34	136	106		
6	103	85	99	100	37	108	

The distance from Slagelse (town number 4) to Køge (town number 1) is found by looking up row number 4 and then column number 1, where we see that the distance is 56 km.

These questions concern a Java class `AfstandsTabel` (distance table), that contains the names of some towns ('byer') and the corresponding distance table.

To begin with, we assume that the class `AfstandsTabel` has a method `int afstand(int by1, int by2)` which returns the distance ('afstand') between the towns `by1` and `by2`. For instance, the call `afstand(4, 1)` returns the distance between Slagelse and Køge. The method `afstand` will also return the distance between Slagelse and Køge when called as `afstand(1, 4)`. That is, the order of the two towns is immaterial.

Moreover, we assume that the class has a method `bynummer` (town number), which given the name `bynavn` of a town returns its number. The method must be declared as `int bynummer(String bynavn)`.

Question 3.1

Write a method `rutelængde` (route length), which takes as argument a route (an array of town numbers), and computes the length of a journey through these towns (in the given order). For instance, it may be called as follows:

```
int[] minRoute = {0, 1, 4, 6}; // København, Køge, Slagelse, Kalundborg
int længde = rutelængde(minRoute);
```

You should write a method with the declaration `int rutelængde(int[] byer)`. If called with a route (an array of town numbers) with 0 or 1 elements, it must return 0 km. The method must use the method `afstand` mentioned above.

Question 3.2

Write a new version of method `rutelængde` that takes as argument an array of town names. It should have this declaration: `int rutelængde(String[] byer)`. You may use the method `bynummer` mentioned above. You may also use the method `rutelængde` from Question 3.1, even if you have not answered that question.

Question 3.3

Assume now that the method `bynummer` throws an exception of class `UkendtBy` (unknown town) if asked for a town that is not in the distance table. Thus the method must be declared as `int bynummer(String bynavn) throws UkendtBy`. This question is *not* about writing the method `bynummer` or declaring the class `UkendtBy`.

Rewrite the method `rutelængde` from Question 3.2 so that it catches the exception and returns -1 (that is, minus one) as the route length if there is an unknown town in the route. If you have not answered Question 3.2, just indicate by a comment where in your answer you would use the answer to Question 3.2.

Question 3.4

Write the declaration `class Afstandstabel { ... }` of class `Afstandstabel` itself, but include only the following aspects of the class:

- The class must have this constructor:

```
AfstandsTabel(String[] byNavne){
    antalByer = byNavne.length;
    afstande = new int[antalByer][];
    for (int i=1; i<antalByer; i=i+1)
        afstande[i] = new int[i];
    byer = byNavne;
}
```

You must give correct declarations of the three fields `antalByer` (number of towns), `afstande` (distances), and `byer` (towns), so that the above constructor can be compiled and works.

- You must write a method `void sætAfstand(int by1, int by2, int afst)`, that inserts the distance `afst` between the two towns `by1` and `by2` in the array `afstande`. Imagine that a series of calls to this method is used to fill in the distance table. The method should not insert anything when `by1` equals `by2` (because in this case the distance is zero, and there is no need to store it in the table).
- Write the method `int afstand(int by1, int by2)` described in the introduction to Question 3.

Question 3.5

Write the method `int bynummer(String bynavn)`, described in the introduction to Question 3, which given a town name `bynavn` returns the town number. The method must belong to class `AfstandsTabel` and must use the array `byer` in the class. It is immaterial what happens when the given town name is not found. (Thus the method need not throw exception `UkendtBy` as assumed in Question 3.3).

Question 3.6

Add a method `findNærmeste` to class `AfstandsTabel` to find the nearest neighbour town for a given town. For instance, Hundested's nearest neighbour is Hillerød, so the call `findNærmeste(5)` should return 2. The method must have this declaration: `int findNærmeste(int by)`, and must return the number of the town closest to town number `by`.