

INTRODUCTION TO PROGRAMMING - CONCEPTS AND TOOLS
IT UNIVERSITY OF COPENHAGEN

Examiner: Noah Torp-Smith & Jens Chr. Godskesen, IT University of Copenhagen
Exam Form: Written Exam
Date: June 18, 2004
Time: 9:00 – 13:00

SOLUTION TO THE WRITTEN EXAM, JUNE 18, 2004

Please read the following instructions carefully before starting answering the questions.

- The exam is a written exam. You have four (4) hours to answer the questions.
- The exam is graded on the Danish 13 scale.
- The question set is available in English only. You are allowed to answer the questions in either *English* or *Danish*.
- The exam is an open book exam, which for this exam means that in particular the following aids may be used:
 - The text book *Java by Dissection*.
 - Additional course notes, like copies of slides, the notes by Peter Sestoft on searching and sorting, and on software testing.
 - Handwritten notes.
 - A language dictionary.
- No electronic devices are permitted. Electronic dictionaries are not allowed.
- The exam consists of three main questions. The weight of each question is given (in percentage points) next to the question. All three questions should be answered.
- This question set consists of six (6) pages.
- When using classes or methods from the book or from the notes by Peter Sestoft it is not necessary to copy them. However, you have to give the precise location of those (for example, source, edition, section number, page number). It is your responsibility that the location can be identified.

(Weight 25%) QUESTION 1

QUESTION 1.1

ANSWER:

```
s = Aloha
s = Hi, st = Hi
s = Davs!, st = Jalla
s = Hej, st = Guten Tag
s = Hello, st = Bon jour
```

QUESTION 1.2

State the scope of the following identifiers:

1. ANSWER: All of class S1, and classes that inherit from it and do not redefine `s` (so also classes S2 and S3).
2. ANSWER: All of class S2.
3. ANSWER: All of class S3.
4. ANSWER: The body of that constructor.

QUESTION 1.3

ANSWER: No. The variable `this` is immutable. Also `s` has type S1, which is a *superclass* of S2, so it cannot be used here.

QUESTION 1.4

ANSWER: No. The method declaration of `toString` overrides that of S1 (or `object` for that matter). Because of the subtype principle, it is not legal to narrow access to instance methods of subclasses. See the discussion on p. 241 - 242 in "Java by Dissection".

QUESTION 1.5

ANSWER: The output is:

```
s = Goodbye, st = null, str = Farewell
```

When the call to `super(s,st)` is invoked, the variable `st` is set to "Cheers" *in the superclass S2*. That is, the call only changes the variable `super.st`, not the local instance variable `st`. The latter is the one that is referred to by the method `toString`, and since it has not been initialized by our test program, it still has the default value `null`.

(Weight 40%) QUESTION 2

QUESTION 2.1

ANSWER: No. The runtimes given by the O -notation are *asymptotic*, so nothing can be said about an input of a predetermined size.

QUESTION 2.2

ANSWER: When an array is not sorted.

ANSWER:

Iteration	i	arr[i]	found
1	9	27	false
2	14	49	false
3	11	31	false
4	12	42	true

QUESTION 2.3

ANSWER:

137	157	131	149	151	127	<u>163</u>
127	<u>131</u>	157	149	151	137	163
<u>127</u>	131	157	149	151	137	163
127	131	137	<u>149</u>	151	157	163
127	131	137	149	<u>151</u>	157	163

QUESTION 2.4

ANSWER: The algorithm will still sort the array correctly. Since the algorithm just moves the pivot to its right place and partitions it according to the pivot, it does not matter the pivot was placed initially. The time complexity of the algorithm is the same, the analysis in Sestoft's note is not dependent of the initial position of the pivot.

QUESTION 2.5

ANSWER: Here is my implementation:

```
static int[] merge(int[] A, int[] B) {
    int k = A.length + B.length;
    int[] res = new int[k];
    int i = 0; int j = 0;

    for(int l = 0; l < k; l++) {
        if (i < A.length && j < B.length) {
            if(A[i] < B[j]) {
                res[l] = A[i];
                i++;
            }
            else {
                res[l] = B[j];
                j++;
            }
        }
        else {
            if (i < A.length) {
                res[l] = A[i];
                i++;
            }
            else {
                res[l] = B[j];
                j++;
            }
        }
    }

    return res;
}
```

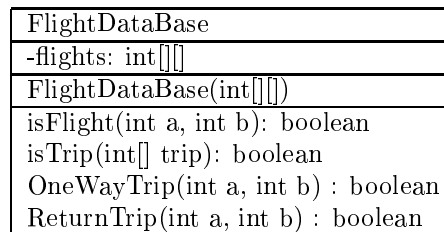
(Weight 35%) **QUESTION 3**

An airline company has flights between airports. To have an overview of the connections the company maintains a small database, i.e. a two dimensional array. For simplicity the destinations are identified by integers starting with zero (0). For example, a particular array with airports 0, . . . ,5 could look as follows:

	0	1	2	3	4	5
0	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>
1	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>
2	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>
3	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>
4	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>
5	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>

where a *true* at row *i* and column *j* denotes a flight from *i* to *j*, dually a *false* at row *i* and column *j* means that there is no flight from *i* to *j*. Clearly, there are no flights from an airport to itself (which explains the values on the main diagonal). It may be that there are only flights in one direction between two destinations.

In this question we will implement a class `FlightDataBase` representing a route database with the following UML class diagram:



The instance variable `flights` contains the two-dimensional array of flights described above.

QUESTION 3.1

```
boolean[] [] db = {{false,true,false},{false,false,true},{false,false,false}};
FlightDB fdb = new FlightDB(db);
```

QUESTION 3.2

```
public boolean isFlight(int a, int b) {return flightDB[a][b];}
```

QUESTION 3.3

```
public void updateFlight(int i, int j, boolean b) {
    flightDB[i][j] = b;
}
```

QUESTION 3.4

```
public boolean isTrip(int[] t) {
    for(int i = 0; i < t.length-1;)
        if (!isFlight(t[i],t[++i])) return false;
    return true;
}
```

QUESTION 3.5

```
public boolean returnTrip(int a, int b) {
    return oneWayTrip(a,b) && oneWayTrip(b,a);
}
```

QUESTION 3.6

```
public boolean oneWayTrip(int a, int b) {
    if (a==b) return true;
    for(int i = 0; i < flightDB.length; i++)
        if (isFlight(a,i) && oneWayTrip(i,b)) return true;
    return false;
}
```