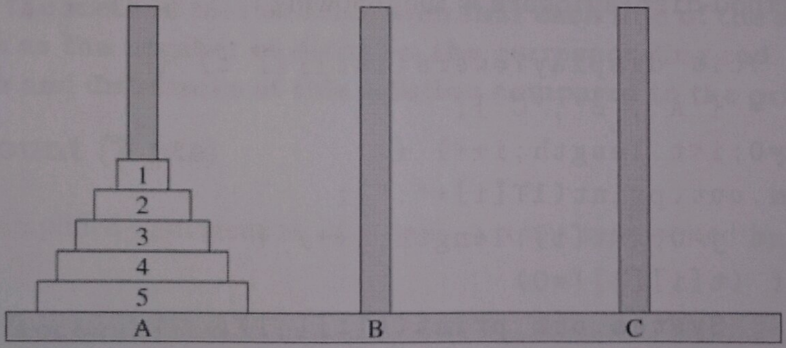
Interrogation d'informatique
PCC-ASINSA-SCAN 1st year - March 2017

Total duration : *1h30*
Allowed materials : *None.*

— The graduation can still change.
— The assignment is on 4 pages.
— All the questions are independant. If it is necessary for A to be solved to solve B, then you can do as if you had solved A (and clearly indicate your assumption) to solve B.

1 Tower of Hanoi (13 pts)

Game description : We have three rods A, B and C and disks of different sizes that can slide onto the rods. A the beginning, the rod A holds n disks of diameter 1 to n, stacked **in decreasing order** of the diameters : the disk of diameter n is at the bottom, then the disk of diameter $n-1$ and so on with, at the top, the disk of diameter 1. The rods B and C are empty. This configuration is presented on the following figure for $n = 5$:

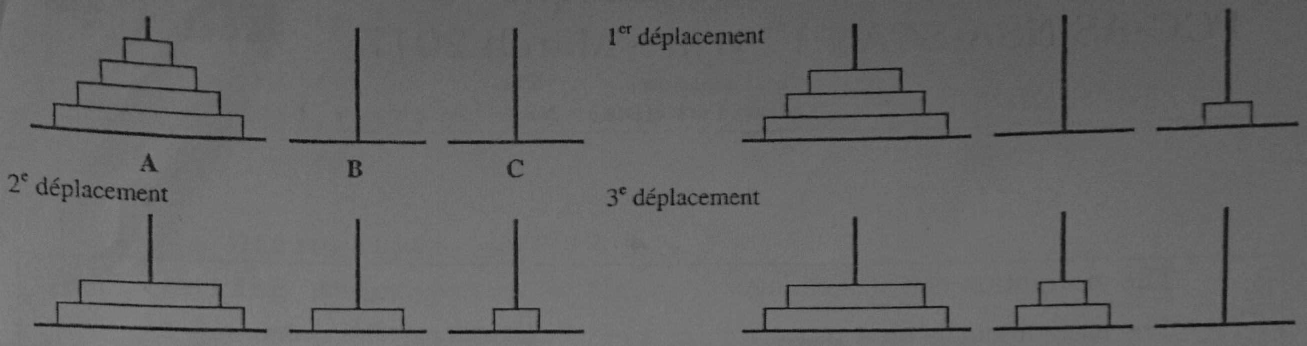


The goal of this game is to rebuild the stack of disks initially on A on another rod (B or C) by moving the disk one by one and according to the following rule :

a disk must never be stacked on a larger disk or must be placed alone on a rod.

One can move a disk of diameter k from a rod to another only if the second rod contains no disk or if its top disk is of diameter strictly greater than k.

The following figure give an illustration of the moves (i) A $\rightarrow$ C then (ii) A $\rightarrow$ B and finally (iii) C $\rightarrow$ B from the initial configuration and for $n = 4$.



Modelisation and display : We choose to represent the state of the game using a 2D array for which each *line* corresponds to a rod. The call to the method `displayTowers` (whose code is provided below) produces the following display :

```
A:54321
B:-----
C:-----
```

for a game in its initial configuration and for $n = 5$. After the three moves previously described, the display becomes :

```
A:543--
B:21---
C:-----
```

54321
00000
00000

The code of the method `displayTowers` is the following :

```
public static void displayTowers(int[][] t) {
    char[] lT = {'A','B','C'};
    for(int i=0;i<t.length;i++) {
        System.out.print(lT[i]+":");
        for(int j=0;j<t[i].length;j++) {
            if (t[i][j]!=0)
                System.out.print(t[i][j]);
            else
                System.out.print('-');
        }
        System.out.println();
    }
}
```

Additional comments : To simplify the code, we give a number to each rod :

$A \leftrightarrow 0 \qquad B \leftrightarrow 1 \qquad C \leftrightarrow 2$

Furthermore, for each method to write, we assume that the state of the game respects the rule when the method is called. Finally, we remind that the efficiency of the proposed algorithm will be taken into account in the evaluation.

(Q1.1) Write the method `createTowers` **that returns the game in its initial configuration, for a number of disk n given as parameter. (2pts)**

(Q1.2) Write the method `isOver` **returning** `true` **if and only if the game is finished, that is to say that the stack has been rebuilt on the rod B or C. (2pts)**

(Q1.3) Write the method `isEmpty` **returning** `true` **if and only if the rod, whose number is given as parameter, is empty (warning, this does not means that this methods takes only one parameter). (1pt)**

We define the height of a rod as the number of disks that it contains. In the initial configuration, the height of the first rod (index 0) is thus n while the height of the other rods (empty) is 0.

(Q1.4) Write the method `rodHeight` **returning the height of the rod whose number is given as parameter (again, this does not means that this method has only one parameter). (2pts)**

We say that a move is *valid* if it follows the rules *and* if it changes the state of the game.

(Q1.5) Write the method `moveDisk` **that (i) removes one disk from the rod numbered** `nTD` **to put it on the rod numbered** `nTA` ***if*** **the move is valid and (ii) returns** `true` **if and only if the move happened.**
Be careful to the case where the originating rod is empty and for which the move is not valid. (4pts)

The chosen model is such that each line of the array contains a fixed number of columns whereas the corresponding rod can contain a variable number of disks.

(Q1.6) Rewrite the method `createTowers` **such that each line of the array has as many columns as the number of disks on the corresponding rod. What are the advantages and drawbacks of this solution compared to the original one ? (2pts)**

2 Bank account (7 pts)

We consider the simplified management of a bank account represented by the following Java class :

```
public class Account{
    String name;
    int number;
    int balance;

    public Account(String myName, int myNumber, int deposit){
        /* ... */
    }
    public void deposit(int amount){
        /* ... */
    }
    public void withdraw(int amount){
        /* ... */
    }
    public void sendTransfer(int amount, Account account){
        /* ... */
    }
    public void receiveTransfer(int amount, Account account){
        /* ... */
    }
    public String toString() {
        String description = "name :"+this.name+" number:"+this.
           number+" balance:" +this.balance;
        return description;
    }
}
```

This class allows to :

- create an account identified by a number for a person with an initial credit
- perform a deposit (method `deposit`)
- withdraw money (method `withdraw`)
- send an amount to another account (method `sendTransfer`)
- receive a amount from another account (method `receiveTransfer`)

The cents are not taken into account : the amounts transfered are in euros.

The signatures of the methods are already written in the class. We now have to write the body of those methods.

(Q2.1) Write the body of the constructor (0,5pt)

(Q2.2) Write the body of the methods `deposit` **and** `withdraw`**. This bank does not give credit : one cannot withdraw more than the current balance. (1 pt)**

(Q2.3) Write the body of the method `sendTransfer` **(1,5pt)**

(Q2.4) Write the body of the method `receiveTransfer` **(1,5pt)**

(Q2.5) Write the `main` **of a class named** `TestAccount` **allowing to generate the following sequence of actions : (2,5pts)**

- Create an account for Durand, number 111, with a initial deposit of 1000 euros;
- Create an account for Dupond, number 222, with a initial deposit of 500 euros;
- Durand witdraw 150 euros;
- Dupont deposit 250 euros;
- Dupont sends a transfer of 450 euros to Durand
- Display the information of the accounts of Durand and Dupont;