Exam in Computer Science
SCAN 2nd year - March 2021

INSA

Total duration : *1h*
Authorized documents : *One double sided A4 sheet*
Attention : cell phones are not allowed.

— The point distribution is not final and the subject is 5 pages long.
— The exercices are independent.
— A **badly indented** or **badly commented** program, or **bad choice of variable names** will be **sanctioned** *(up to -1 point)*.

Last important details to be read carefully :

— the parts are independent. You can deal with them in the order you want.
— for each question, you should use (as much as possible) the methods presented in the previous questions, even if you have not coded them ;
— the efficiency of the proposed algorithms will be the subject of particular attention ;
— the java code given in your answers should be as concise as possible and therefore use inheritance structures as much as possible ;
— the cleanliness of your copy will be taken into account in the scoring.

1 Bubble sort (4 pts)

Below you find java code of a bubble sort algorithm, which allows to sort data passed in the argument (the Collection c). The choice of this type of collection is open and can, for instance, be a LinkedList, or ArrayList etc..

The asymptotic complexity of the bubble sort algorithm is known to be $O(N^2)$, where N is the size of the collection, for a judicious choice of the data structure. Give the asymptotic complexities for the two choices `ArrayList` and `LinkedList` in the table below :

```
1  static void BubbleSort (Collection c)
2  {
3      int n=c.size();
4      for (int i=n; i>=1; i--) {
5        for (int j=2; j<=i; j++) {
6
7          if (c.get(j-1)> c.get(j)) {
8
9            Object temp = c.get(j-1);
10
11           // The methode set() puts an object (arg 2) into a cell (arg 1)
12           c.set(j-1, c.get(j));
13           c.set(j, temp);
14         }
15       }
16     }
17 }
```

Structure	Assumptotic complexity
`ArrayList`	
`LinkedList`	

2 Hangman (16 pts)

"Hangman" is a game traditionally played between two players using a sheet of paper and a pencil. We will implement an electronic version, where a human user has to guess a word chosen by the opponent played by the computer. On each turn, the human user must choose a letter. If this letter is part of the word, it will be displayed (NB : if a correctly guessed letter appears several times in the word, it will be displayed in all the places where it appears in the word). Otherwise, it will be added to a list of erroneous letters, and an extra stroke will be drawn for a "hanged man" figure. The user will have won when he or she has guessed the word. He or she will have lost when a certain number of wrong letters have been used, i.e. when the man is completely drawn (=hanged). Figure 2 shows the main window during two different phases of the game.

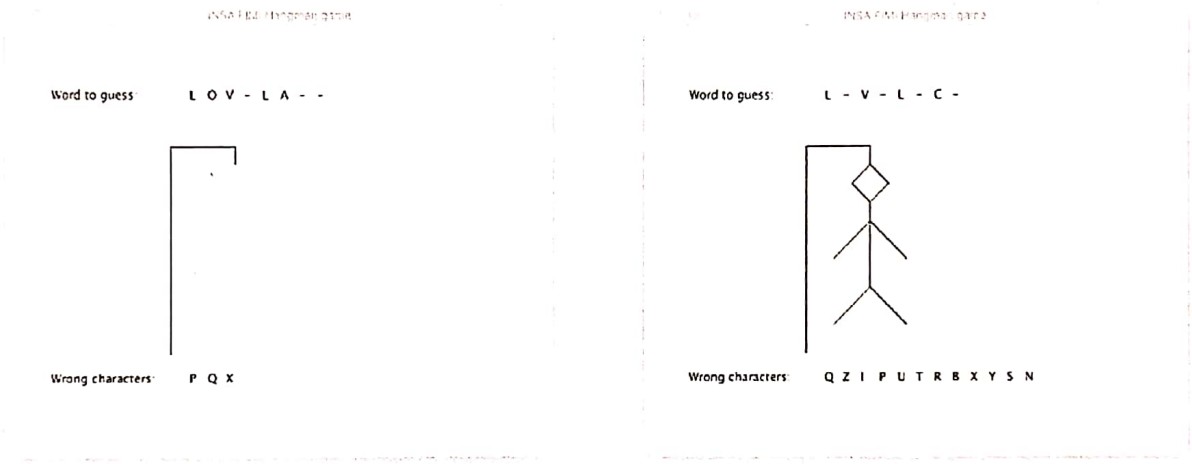

FIGURE 1 – Screenshots of two different games : in the middle of a game (left) and after a defeat (right).

We will not work on the user interface and painting.

Let us recall some existing methods available for java collections :

`contains(...)`	Tests the existence of an object in a collection.
`add(...)`	Adds an element to a collection.
`size()`	Returns the number of elements in a collection.

The program will use the following data structures :

`target`	a list of the letters of the word to guess (may contain duplicates, for example {L,O,V,E,L,A,C,E}).
`guessedRight`	a list of correctly guessed letters (no duplicates).
`guessedWrong`	a list of wrongly guessed letters (no duplicates).

A draft of the program is given below. The methods to be implemented are indicated in the following comments and questions.

```
1  import ...
2
3  public class Hangman extends JFrame implements KeyListener {
4
5      // Declaration of the data structures
6      LinkedList<Character> target, guessedRight, guessedWrong;
7
8      public Hangman (String word) {
9          super ("INSA FIMI Hangman game");
10         this.setSize(600, 480);
11         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12
13         //Initialization of the data structures
14         this.target = new LinkedList<Character>();
15         this.guessedRight = new LinkedList<Character>();
16         this.guessedWrong = new LinkedList<Character>();
17
18         // Method which configures the word to be guessed by the human
19         // Method setUpTarget must be implemented (Q2.1.1)
20         this.setUpTarget(word);
21
22         (...) // drawing and handling of keyboard events - nothing to be done
23         this.setVisible(true);
24     }
25
26     // Method called when a keyboard key is pressed
27     public void entreeLettre(Character c) {
28         // Method addCharacterToGame must be implemented (Q2.1.2)
29         this.addCharacterToGame(c);
30
31         // Method winOrLose must be implemented (Q2.1.3)
32         this.winOrLose();
33     }
34
35     public static void main (String [] args) {
36         new Hangman ("LOVELACE");
37     }
```

(Q2.1) Handling the state of the game

(Q2.1).1 *Give the header and the implementation of the method `setUpTarget(String word)`. This method takes a character string as argument and iterates over it, adding characters to `target`. For that, you can use `word.length()`, which gives you the length of the string `word` and `new Character(word.charAt(i))`, which creates a new object `Character` from a character at position i in the string `word`.*

(Q2.1).2 *Give the header and the implementation of the method `addCharacterToGame()`. This method takes as argument a letter (a `Character`) and it checks if the letter is part of the word to guess. Then it updates the data structures by inserting the letter in the appropriately chosen list. If the user chooses a letter already chosen previously, a new window will be displayed using the instruction `new ResultWindow("Already guessed!");` (don't deal with the result window right now, we will come back to it in a later question).*

(Q2.1).3 *Give the header and the implementation of the method `winOrLose()`. This method checks if the user has won the game (all the letters of the target word have been guessed), or lost (the number of erroneous letters is equal to 12, i.e. to the number of dashes of the hanged man) — Attention : the target word can contain a letter several times, in our example the letter "L".*

According to the case (victory or defeat), a new window will be displayed with the instructions `new ResultWindow("You won");` or `new ResultWindow("You lost!");`, respectively.

(Q2.2) The Result Window

The previous two methods instantiated the ResultWindow class, which is now the subject of this part. Below is a first version of the code for `ResultWindow`.

```
1  import ...
2  public class ResultWindow extends JFrame{
3
4     public ResultWindow(String s){
5        this.setSize(300,130);
6        this.setLocation(700,200);
7        this.add(new JLabel(s));
8        this.setVisible(true);
9     }
10 }
```

We now want to make sure that when the player loses or wins, the window displays, in addition to the message "You won!" or "You lost!", the percentage of the number of errors among the number of letters played and the percentage of the number of letters guessed. The display should like in the figure below.

1. Lady Ada Lovelace (1815-1852) was a computer pioneer best known for making the first real computer program, while working on an ancestor of the computer.

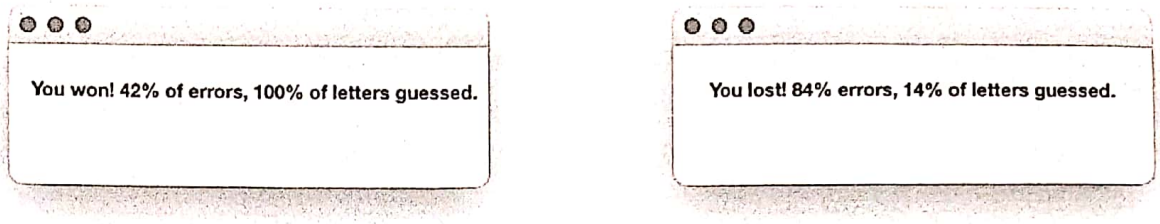

FIGURE 2 – Example screenshots for results windows in the case of victory (left) or defeat (right).

(Q2.2).1 *Implement a method `countUniqueLetters` in class `Hangman`, which takes as parameter a list of characters with potential duplicates and which returns the number of unique characters in this list. If the list contains the letters L,O,V,E,L,A,C,E, it must therefore return 6.*

(Q2.2).2 *Overload the `ResultWindows` constructor, so that it takes the necessary information as a parameter, and displays the requested percentages. Also tell how to modify the `winOrLose` method.*

(Q2.3) Complexity

Suppose that the variables `target`, `guessedRight` and `guessedWrong`, which were structures of the type `LinkedList`, are now replaced by collections of the type `MyEfficientCollection`. The structures `MyEfficientCollection` are collections like `LinkedList` or `ArrayList`, but with the following characteristics :

— The methods add and `contains` have a complexity of $O(\log n)$ where n is the size of the collection.
— There is no direct access to the elements with get(i), they can only be iterated over with `for each`.

Specify for the methods below how this modification will impact the complexity of the calculation by putting a cross in the correct box. Wrong answers can lead to negative points, do not answer if you are not sure what your choice is.

Variable	Complexity increased	Complexity. unchanged	Complexity decreased
`setUpTarget()`			
`addCharacterToGame()`			
`winOrLose()`			
`countUniqueLetters()`			