Computer Science Exam
2nd year - March 2018

INSA

Total length : 1h
Authorised documents : One double-sided A4 sheet of notes.
The three questions are independant.

Exercise 1 : "for" and "for each" (5 pts.)

Let us consider the following code (that is supposedly integrated in a main method of a class) :

```
LinkedList<Integer> myListe = new LinkedList<Integer>();
int nbElt = 1000;
for(int i=0; i<nbElt; i++)
      myListe.add(i);

int nRep = 10000;
int s=0;

long timeBefore = System.currentTimeMillis();
for (int j=0; j<nRep; j++){   // Loop 1
      for(int v : myListe){
          s += v;
      }
}
long timeAfter = System.currentTimeMillis();
System.out.println("Duration of traversal 1: " + (timeAfter-timeBefore) + " ms");

s=0;
timeBefore = System.currentTimeMillis();
for (int j=0; j<nRep; j++){ // Loop 2
      for(int i=0; i<myListe.size(); i++){
          s += myListe.get(i);
      }
}
timeAfter = System.currentTimeMillis();
System.out.println("Duration of traversal 2: " + (timeAfter-timeBefore) + " ms");
```

The execution of the program will show two execution times, one for loop 1 and one for loop 2.

(Q1-1 – 1 pt) For which loop the execution time is lower?

(Q1-2 – 2 pts) Briefly justify your response by explaining the operations that are performed in the methods in each loop.

(Q1-3 – 2 pts) If `myListe` was an `ArrayList`, would loop 2 have taken more time, or less? Justify your response.

Exercise 2 : Modelling the library (7 pts.)

We will create a data model of books managed by a library. In this simplified model, we will store the title of a book and the number of times it has been lent. The `Book` class is thus defined as follows:

```
public class Book {
      private String title;   // title of the Book
      private int nLends;     // number of times the Book has been lent/borrowed

      public Book(String aTitle, int lends){ // Constructor
            title = aTitle;
            nLends = lends;
      }

      public String toString(){ // Display
            return "[" + title + " / " + nLends +"]";
      }

      public String getTitle(){ // Returns the title
            return title;
      }

      public int getLends(){ // Returns the number of lends
            return nLends;
      }

      public void setLends(int lends){ // Sets the number of lends
            nLends = lends;
      }
}
```

A library has a name and the complete collection of books, where **only one exemplar for each book can be kept**. The `Library` class is thus defined as follows:

```
public class Library {
      private String name;
      private TreeSet<Book> books;

      public Library(String nameLib){ // Standard Constructor
            name = nameLib;
            books = new TreeSet<Book>();
      }


      public Library(String nameLib, String fileName){
      /* This constructor creates the library and fills it with books.
      We assume that the readFromFile method of the Utils class returns a
      TreeSet from the file with the given file name. */

            name = nameLib;
            books = Utils.readFromFile(fileName);
      }


      public void addBook(Book book){ // Adds a book to the books list
            books.add(book);
      }
```

```
    public String toString(){ // Display
        String res= "[";
        for(Book l : books){
            res+=" " + l;
        }
        return res+" ]";
    }

    public String getName(){ // Returns the name
        return name;
    }

    public TreeSet<Book> getBooks(){ // Returns the books
        return books;
    }
}
```

(Q2.1 - 2 pts) What property of `TreeSet` satisfies the constraint of **`Library`** specified in the text above (in bold face)?

In order to be able to use `TreeSet`, the **`Book`** class requires two things: a `compareTo` method and a standard Java interface.

(Q2.2 - 2 pts) Which interface is this? Modify the code of the **`Book`** class accordingly.

(Q2.3 - 3 pts) Propose an implementation of the `compareTo` method of the **`Book`** class ordering the Books in alphabetical order (without considering the number of lends).

Remember that the alphabetical comparison of two Strings A and B by the call `A.compareTo(B)`:

- returns a negative integer if A comes alphabetically before B (A<B),
- returns 0 if A and B are identical (A=B),
- returns a positive integer if A comes alphabetically after B (A>B).

Exercise 3 : Library fusion (8 pts.)

We would like to consistently merge the book collections of two libraries as modelled in exercise 2. To this end, we will consider 2 methods : `bibDiff` and `bibInter`. Their headers are:

```
public TreeSet<Book> bibDiff( Library anotherLib )
public TreeSet<Book> bibInter( Library anotherLib )
```

We assume that the `compareTo` method is implemented in the Book class (as asked in Q2.3) and that `L1.compareTo(L2)` returns a negative integer if the title of L1 comes alphabetically before the title of L2, 0 if it is the same title (equality), and a positive integer otherwise.

(Q3.1 - 3 pts) The `bibDiff` method returns the set of Books that are present in the library and that are not present in the 2nd library `anotherLib`, passed as a parameter.

Write the `bibDiff` method.

(Q3.2 - 3 pts) The `bibInter` method returns the set of Books that are present in the library and that are also present in the 2nd `library` `anotherLib`, passed as a parameter. For each pair of identical Books that is found, it choses the Book with the lower number of lends.

Write the `bibInter` method.

(Q3.3 - 2 pts) The main program performs the fusion of `bibA` and `bibB` and displays it.

Using the preceding methods, complete the main program given below:

```
public static void main(String[] args){
      Library libA = new Library("libA", "refbib.data");
      Library libB = new Library("libB", "testbib.data");
      Library libFusion;
                              … TO COMPLETE (Q3.3) …
      System.out.println("Library fusion : " + libFusion);
}
```