Exercises 23

lists (continued)

Exercise 23.1 Work on a list

This exercise uses the class list.
We will work on the list obtained by the following Java code:

```java
ListeIter lal = new ListeIter();
lal.ajouterAuDebut(4);
lal.ajouterAuDebut(3);
lal.ajouterAuDebut(2);
lal.ajouterAuDebut(1);
```

1. **draw** list `lal`
2. **define** a list containing the same values in the same order using a single constructor and inserting elements through `ajouerALaFin`.
3. by what Java code using the list `lal` can we reference the item 1 of the list?
4. by what Java code using the list `lal` can we reference the item 2 of the list?
5. by what Java code using the list `lal` can we reference the item 3 of the list?
6. by what Java code using the list `lal` lal we reference the portion of the list that contains only two elements?
7. by what Java code can we replace the integer 3 by 5 in the list?

Exercise 23.2 - Eement insertion

The purpose of this exercise is to familiarize you with lists and associated graphical notation.
This notation is only an abstract representation of what happens in memory.
It can help us humans to understand what happens in the machine.
This can help us to build correct programs.

Question 23.2.1

Our goal is to write a function that adds an item to a given rank in a list. We have already seen the insertion at the start and at the end of the list.
With the insertion at a given rank, we complete the set of methods to add an item to a list.
The algorithm needs to move through the list to the right place, and then insert the new cell. The previous cell points at itself and to the next element.

Let us see an example:

Suppose the list contains 1, 2, 3, 4 and we want to insert 5 in position 3.
Draw the representation of the initial state.
Should we use an auxiliary variable? If so, how should it be initialized?
Add it to your representation.
Then draw all the intermediate states through which the algorithm must pass to complete the transaction. Make out a separate drawing for each state.
Try to describe the change of state by a Java statement (or a portion of instruction as we did previously). This statement must be an assignment or method call like this:

```
- l1 = l2;
- l = l.getSuivant ();
- l1 = new ElementListe (x, l2);
- l1.setSuivant (l2);
```

If you need more than one instruction to describe the passage from one state to the next , it means that you forgot an intermediate state between the two.
If it is not possible to find a statement that describes the transition from one state to another it means that you try to do the things impossible in Java.

Question 23.2.2
At this point you should have your sequence of drawings and in parallel to the corresponding sequence of instructions. If things still are not clear, repeat the same for other examples of lists, and add entire row.
Write a procedure that works in all simple cases (like the example studied).
In this first version, we process any cases or problems that may occur. Test your procedure.
Basically, we must find different assignments Example organized with control structures (`while`, `if`).
Eventually, it will require new variables to describe the conditions of these control structures.

Question 23.2.3
We will now finish the job by handling errors and special cases.
Special cases:
The procedure, does it work if you want to add the item at first position?
Does it work if you want to add the item at last position?
Modify the procedure so that it works in all these cases.
The problem that can happen when the list does not contain enough information for the required range. For example, it is not possible to insert at tenth position in a list of three integers.
Where can we detect this problem and how?