

Final Exam, June 3rd 2017
No document except a dictionary, length 3h

Any answer difficult to understand will be considered as false.

You may add any code (method, data, class) not requested each time you think it is necessary.

A class or a method requested may be considered available in the follow-up of the problem, even if you have not treated the question. It is recommended to read all the subject before starting your work.

If you want to use a class (or a given method) from the Java API, and you don't remember its name, give to it an explicit name and explain clearly the goal of the class on your copy.

You may use, each time you think it is comprehensible, an abbreviation instead the complete name of a class or a method, and also use '...' to replace an unmodified part of the code.

The instructions `import` and `package` will be not given.

Problem - Circular List

We consider the following class `CircularList`.

```
public class CircularList {
    private Object[] elements;

    public CircularList(Object... elements) {
        this.elements = elements;
    }

    /** get the element at index i % size(). */
    public Object get(int i) {
        return elements[i % elements.length];
    }

    /** return the first index of the object o in the circular list, -1 if the
     ** object is not present. */
    public int indexOf(Object o) {
        for (int i = 0; i < elements.length; i++) {
            if (elements[i].equals(o))
                return i;
        }
        return -1;
    }

    /** number of elements contained in the circular list. */
    int size() {
        return elements.length;
    }
}
```

Question 1 Modify `CircularList` in such a way that the following instruction

```
System.out.println(c);
```

where `c` is an instance of `CircularList` and n its size, will give the following result :

Circular list of size n

Question 2 The result of the following instructions

```
Object [] a = {1, 2, 3};
CircularList cl = new CircularList(a);
System.out.print(cl.get(0) + " ");
*** hidden instruction ***
System.out.println(cl.get(0));
```

is

1 4

where the hidden instruction does not contain any reference to `cl`.

Give the content of the hidden instruction and correct this lack of encapsulation.

Question 3 Add a method `Object nextElement()` which returns the element following the element previously returned by `nextElement` in the array `elements`. The first call to `nextElement` will return the first element of the circular list, i.e. element with index 0. The element which follows the element of index `size() - 1` is the element of index 0.

For example, the following instructions

```
CircularList cl = new CircularList(1, 2, 3);
for (int i = 0; i < 10; i++)
    System.out.print(cl.nextElement() + " ");
```

will give the following result :

1 2 3 1 2 3 1 2 3 1

Question 4 Modify `CircularList` in such a way that calling the method `indexOf(Object o)` throws an exception `ObjectMissingException` if the object `o` is not present. Give the code of the class `ObjectMissingException`.

Question 5 Modify `CircularList` in such a way that the type of its elements becomes generic.

Question 6 Modify `CircularList` in such a way that two circular lists are equals if and only if they contain the same elements in the same order.

Which other method of `CircularList` (inherited from `Object`) is it necessary to rewrite? Propose an implementation of it.

Question 7 Now, we consider a new class `CircularButton` with the following code :

```
public class CircularButton<E> extends JButton {

    public CircularButton(CircularList<E> cl) {
        if (cl.size() == 0) {
            throw new IllegalArgumentException();
        }
        setText(cl.nextElement().toString());
        addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println(getText());
                setText(cl.nextElement().toString());
            }
        });
    }

    public static void main(String[] args) {
        CircularList<String> cl = new CircularList<>(args);
        CircularButton<String> cb = new CircularButton<>(cl);
        JFrame f = new JFrame();
        f.add(cb);
        f.pack();
        f.setVisible(true);
    }
}
```

Describe in few lines what's happen when we execute the `main` method of this class.

Question 8 Modify the code of `CircularButton` by replacing the anonymous inner class used to implement the `ActionListener` by a lambda expression.

Question 9 Now, we want to be able to use a `CircularList` as an `Iterator`. We recall the interface `Iterator` :

```
public interface Iterator<E> {

    /** Returns true if the iteration has more elements. */
    boolean hasNext();

    /** Returns the next element in the iteration.
     * Throws NoSuchElementException if the iteration has no more elements*/
    E next();

    /** Removes from the underlying collection the last element returned by
     * this iterator (optional operation). This method can be called only once
     * per call to next(). The behavior of an iterator is unspecified if the
     * underlying collection is modified while the iteration is in progress in any
     * way other than by calling this method.
     * The default implementation throws an instance of
     * UnsupportedOperationException and performs no other action. */
    default void remove();
}
```

```

    /** Performs the given action for each remaining element until all
    elements have been processed or the action throws an exception.
    The default implementation behaves as if:
    while (hasNext())
        action.accept(next());
    */
    default void forEachRemaining(Consumer<? super E> action);
}

```

Write a class `CircularListIterator` which extends `CircularList` and implements `Iterator`. The method `hasNext()` will always return `true`, except if the circular list is empty. The methods `remove()` and `forEachRemaining()` will throw an `UnsupportedOperationException` (*Attention, it is not the default behavior of `forEachRemaining()`*).

To which kind of design pattern corresponds the class `CircularListIterator`?

Question 10 Propose an other solution for `CircularListIterator` which uses delegation instead of inheritance.