université de BORDEAUX

J1INAW12 – Approche Objet
Master Informatique
Pôle Universitaire Français

Final Exam, November 6th 2016
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.*

Constant sets

We want to write a class which implements unmodifiable sets.

Here is a first proposition :

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

    public ConstantSet(Object[] elements) {
        this.elements = elements;
    }

    public boolean contains(Object o) {
        // code omitted
    }

    public int size() {
        // code omitted
    }

    public Iterator<Object> elements() {
        return Arrays.asList(elements).iterator();
    }
}
```

Question 1 Complete the methods `contains(Object o)` and `size()`

Question 2 Modify the constructor in such a way that it becomes possible to replace the instructions :

```
Object [] ts = {"a", "b", "c"};
ConstantSet cs = new ConstantSet(ts);
```

by the instruction :

```
ConstantSet cs = new ConstantSet("a", "b", "c");
```

Question 3 We consider the following sequence of instructions :

```
Object[] ts = { "a", "b", "c" };
ConstantSet cs = new ConstantSet(ts);
System.out.println("contains \"a\" ? " + cs.contains("a"));
?????? // hidden instruction
System.out.println("contains \"a\" ? " + cs.contains("a"));
```

and the following result :

```
contains "a" ? true
contains "a" ? false
```

Replace the question marks (hidden instruction) by an appropriate instruction.

Because we want unmodifiable sets, this result is not suitable. Modify `ConstantSet` in such a way that instances of `ConstantSet` become really constant.

Question 4 Modify `ConstantSet` in such a way that the type of elements becomes generic.

Question 5 Modify the constructor in such a way that, if the array given as argument contains an given element at least twice, an exception `DuplicatedElementException` will be thrown.

Give the code of the class `DuplicatedElementException`. In particular, it must be possible to know, using the exception, which element is contained in the array at least two times.

Question 6 Override in the class `ConstantSet` the method `equals(Object o)` in such a way that an instance `cs` of `ConstantSet` is equal to another object `obj` if and only if

- `obj` is also an instance of `ConstantSet` ;
- `obj` has the same number of elements of `cs` ;
- `obj` contains all the elements from `cs`.

When you override the method `equals`, it is necessary to override another method inherited from `java.lang.Object`. Which one ? *Only the name is requested, not the code.*

Question 7 We consider now a class `OrderedConstantSet` having the same methods and the same behavior than `ConstantSet`, except for the method `equals(Object)` which is defined as follow : an instance `ocs` of `OrderedConstantSet` is equal to another object `obj` if and only if

- `obj` is also an instance of `OrderedConstantSet` ;
- `obj` has the same number of elements than `ocs` ;
- the iterators `obj.elements()` and `ocs.elements()` return the same elements in the same order.

Explain why using inheritance to reuse the code of `ConstantSet` in `OrderedConstantSet` is a bad solution.

What is the good solution to reuse the code of `ConstantSet` in `OrderedConstantSet` ?

Give the implementation of `OrderedConstantSet`, including the code of the method `equals(Object o)`.