Lab session 1: Implementing a class
24292-Object Oriented Programming

1 Introduction

The aim of this lab session is to implement the design of Seminar 1. The session is mandatory and you have to deliver the source code of the java project and a document describing the implementation. We will implement a minimal version of a Logo program. The code instructions will be strings and we will use for the drawing set : "PEN", "FWD", "ROT". For the loop instruction set we will use: "REP" and "END". The application will need to be able to add instructions to a program and implement the correct operations to start this program and give subsequent operations. For example the following program that draws a square on the screen:

REP 4
FWD 100
ROT 90
END

could be created and executed as indicated in the following code, belonging to the main application class LogoProgram:

```
public class LogoProgram {
   public static void main ( String[] args ) {
      Program p = new Program ( ``Square'' ) ;
      p.addInstruction( ``REP'' , 4 );
      p.addInstruction( ``FWD'' , 100 );
      p.addInstruction( ``ROT'' , 90 );
      p.addInstruction( ``END'' , 1 );
      if( p.isCorrect() ){
         p.restart();
         while ( !p.hasFinished() ) {
            Instruction instr = p.getNextInstruction();
            System.out.println( instr.info() );
         }
      }
   }
}
```

The screen output of the previous program should be: FWD 100, ROT 90, FWD 100, ROT 90, FWD 100, ROT 90, FWD 100, ROT 90; that is, instructions corresponding to drawing primitives without any instruction to manage loops.

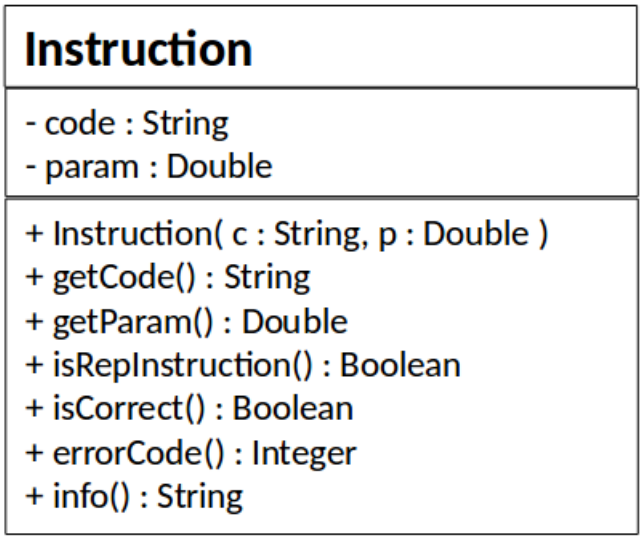
2 Creating the main application class

We will start creating a new Netbeans java project called LogoProgram.

1. Create the project LogoProgram
2. Leave the option create main class selected as this is not a GUI project.

2.1 Creating the Instruction class

Then we will create a new class Instruction. Remember that is done pressing the new file button and with java class selected: call it Instruction. A possible solution of the first seminar is in the picture below:



Add the attributes as specified in the design. Afterwards we will start adding methods: the constructor, setters and getters. Look that "isCorrect" method returns a Boolean instead of an Integer, because we have created an specific method called "errorCode" to know specifically the Integer that identifies the instruction error. And the "isRepInstruction" will return a Boolean that is True if the instruction is "REP" or "END", or False otherwise.

Important: Run the program often to check everything is ok.

Now is the moment to code each Instruction method. As a hint we provide you with an example about how to compare two Strings:

```
String s = new String( ``FWD'' );
if( s.equals( ``ROT'' ) )
   System.out.println( ``Strings are not Equal'' );
```

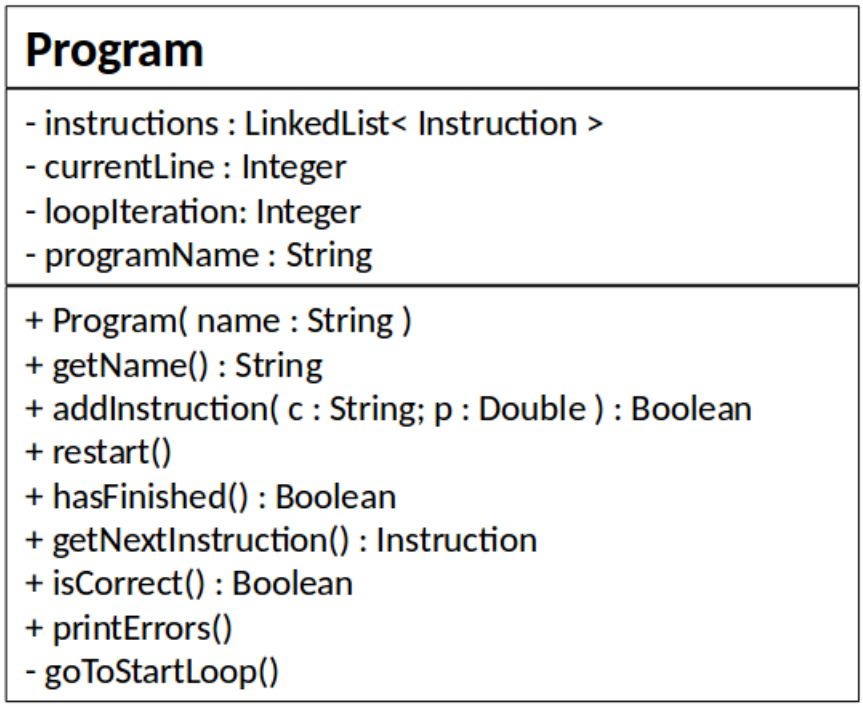
An instruction is considered wrong if:

- The code is not among the valid logo codes
- In "FWD" code, the param is not in the interval (-1000,1000)
- "PEN" code has param different from 0 or 1
- "ROT" code has param greater or equal 360, or less or equal -360
- or "REP" code has param less or equal 0 or greater or equal 1000

Any other case is considered a successful instruction and must return a 0.

2.2 The Program class

Lets create another class: Program, as we did for Instruction.



As we know from last seminar a Program contains a list of instructions. For this list we will use an already existing java object called LinkedList. Elements can be added in a linked list as we wish and they can be acceded in different ways, including direct access by its index. See a complete example below:

```
import java.util.LinkedList;

public class TestLinkedList{
 public static void main( String[] args ) {
   LinkedList< String > list = new LinkedList< String >();
   list.add( ``anElement'' );
   list.add( ``anotherElement'' );
   System.out.println( list.size() );
   String firstElement = list.get( 0 );
   System.out.println( firstElement );
 }
}
```

Now you will need to add all the attributes and methods to implement the desired behavior specified in the introduction taking into account:

- To be able to execute our program we will need a current instruction index currentLine
- To be able to remember at which loop iteration we are we will need an attribute variable loopIteration.
- When we reach an END instruction the method goToStartLoop goes to the first instruction after the previous REP instruction. (**Hint**: This execution logic must be implemented when we move to the next instruction).
- You will also need to implement the methods isCorrect and printErrors so that the correctness of the program is checked and hints of the errors are given.

2.3 The Main Application class

Now you should be able to execute the code presented in the Introduction section. The code draws a square with the turtle and you should see the corresponding output in the screen: FWD 100, ROT 90, FWD 100, ROT 90, FWD 100, ROT 90, FWD 100, ROT 90.

You have to upload the program code of Instruction.java and Program.java to your Git repository, as well as the class containing the main method for testing the classes. The deadline is before the next Lab session.

3 Documentation

Apart from the source code, you should also upload to your Git repository a document that outlines the solution of the problem. To elaborate the document you can use the following guidelines regarding the content:

1. An introduction where the problem is described. For example, what should the program do? Which classes do you have to define? Which methods do you have to implement for these classes?

2. A description of possible alternative solutions that were discussed, and a description of the chosen solution and the reason for choosing this solution rather than others. It is also a good idea to mention the related theoretical concepts of object-oriented programming that were applied as part of the solution.

3. A conclusion that describes how well the solution worked in practice, i.e. did the tests show that the classes were correctly implemented? You can also mention any difficulties during the implementation as well as any doubts you might have had.