

## Report Lab 1

El objetivo de la práctica era implementar el diseño de clases creado en la pasada sesión de Seminario 1.

### Introducción

Hemos creado un programa minimalista llamado Logo Program, cuyas líneas de código serán instrucciones con un nombre y un valor. Dentro de este código primario, también deberemos implementar bucles, que serán denotados con REP y END.

Las clases que conforman el funcionamiento del programa serán:

Instruction	Program
<ul style="list-style-type: none"><li>- code : String</li><li>- param : Double</li></ul>	<ul style="list-style-type: none"><li>- instructions : LinkedList&lt; Instruction &gt;</li><li>- currentLine : Integer</li><li>- loopIteration: Integer</li><li>- programName : String</li></ul>
<ul style="list-style-type: none"><li>+ Instruction( c : String, p : Double )</li><li>+ getCode() : String</li><li>+ getParam() : Double</li><li>+ isRepInstruction() : Boolean</li><li>+ isCorrect() : Boolean</li><li>+ errorCode() : Integer</li><li>+ info() : String</li></ul>	<ul style="list-style-type: none"><li>+ Program( name : String )</li><li>+ getName() : String</li><li>+ addInstruction( c : String; p : Double ) : Boolean</li><li>+ restart()</li><li>+ hasFinished() : Boolean</li><li>+ getNextInstruction() : Instruction</li><li>+ isCorrect() : Boolean</li><li>+ printErrors()</li><li>- goToStartLoop()</li></ul>

Estas clases interactúan con nuestra main class LogoProgram para ejecutar

```
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() );
            }
        }
    }
}
```

### Implementación

El principal objetivo de nuestro programa era implementar correctamente los bucles.

La implementación de este comportamiento ha recaído dentro de nuestra clase Program, que es la clase que tiene acceso a todas las instrucciones en cualquier momento.

Con el método getNextInstruction interactuamos con las instrucciones del programa. Una posible implementación sería:

```
public Instruction getNextInstruction() {
    //Comprobaremos que tipo de instruccion tenemos
    Instruction current_instruction = instructions.get(currentLine);
    if (current_instruction.isRepInstruction()) {
        //Si nuestra siguiente instruccion es un Rep, deberemos iniciar un
        //loop, por lo que asignamos el valor de Rep a nuestro loopIteration
        if (current_instruction.getCode().equals("REP")) {
            loopIteration = (int) Math.round(current_instruction.getParam());
        } //Si nuestra siguiente instruccion es un End, primero habra que
        //comprobar si estamos terminando con una seccion de repeticiones
        else {
            if (loopIteration != 1) {
                this.goToStartLoop();
            }
        }
    }
    currentLine++;
    return instructions.get(currentLine);
}
```

Antes de avanzar, ejecutaremos la instrucción en la que estamos. Decidimos esto al fijarnos en el código que la main class usaría para ejecutar el programa.

Para terminar en crear el comportamiento de un bucle, también implementamos el método goToStartLoop de la siguiente forma:

```
private void goToStartLoop() {
    //Vamos a usar un loop para cuando nos encontramos en una instruccion END
    //y queremos encontrar la instruccion REP que le precede
    //Estamos suponiendo que no habra loops anidados

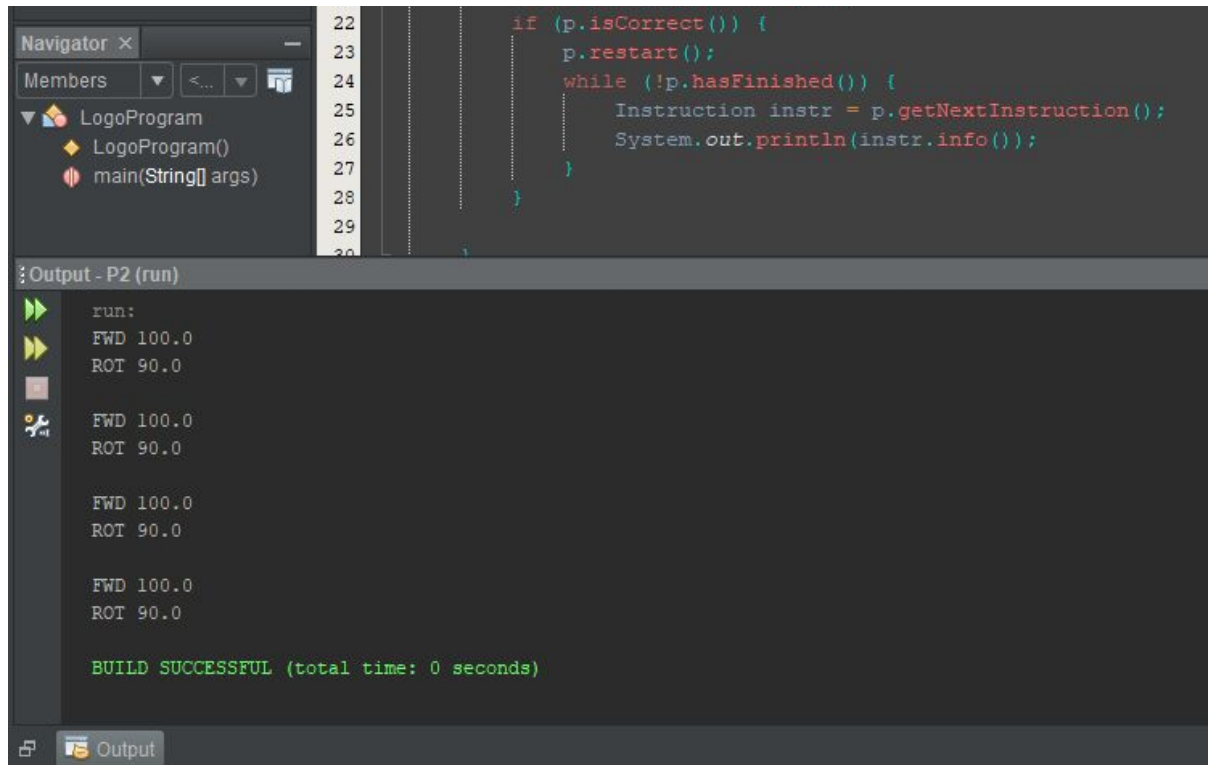
    for (int i = currentLine; i >= 0; i--) {
        Instruction current_instruction = instructions.get(i);
        if (current_instruction.getCode().equals("REP")) {
            currentLine = i;
        }
    }
    loopIteration--;
}
```

El resto de la implementación, hemos seguido el diseño UML de las clases.

## Resultado

Hemos tenido problemas entendiendo donde era implementado el comportamiento de los bucles, a lo mejor los métodos que requieren interactuar con otras clases deberían tener más información, pero ha sido un problema menor al tratarse de la primera práctica, que es más sencilla.

Nuestro output con nuestra implementación ha sido el siguiente:



The screenshot shows an IDE interface. On the left, the 'Navigator' pane shows a project named 'LogoProgram' with two members: 'LogoProgram()' and 'main(String[] args)'. The main editor displays a code snippet with line numbers 22 to 29. The code is as follows:

```
22     if (p.isCorrect()) {  
23         p.restart();  
24         while (!p.hasFinished()) {  
25             Instruction instr = p.getNextInstruction();  
26             System.out.println(instr.info());  
27         }  
28     }  
29
```

Below the editor, the 'Output - P2 (run)' pane shows the execution results. It starts with 'run:' followed by four iterations of 'FWD 100.0' and 'ROT 90.0'. The output ends with 'BUILD SUCCESSFUL (total time: 0 seconds)'.

```
run:  
FWD 100.0  
ROT 90.0  
  
FWD 100.0  
ROT 90.0  
  
FWD 100.0  
ROT 90.0  
  
FWD 100.0  
ROT 90.0  
  
BUILD SUCCESSFUL (total time: 0 seconds)
```