

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

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

Antonio Pintado U172771
Amanda Pintado U137702

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();
            }
        }
    }
    if (!this.hasFinished()) {
        currentLine++;
        return instructions.get(currentLine);
    }
    return current_instruction;
}
```

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--;
}
```

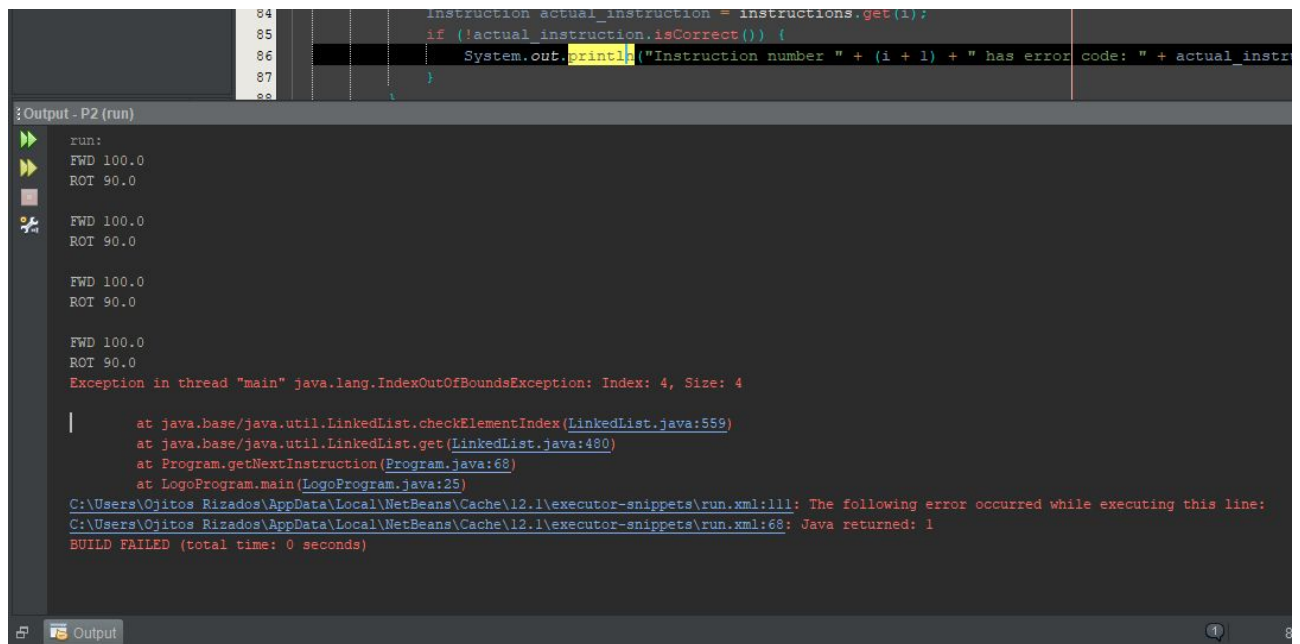
Antonio Pintado U172771
Amanda Pintado U137702

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:



```
84         instruction actual_instruction = instructions.get(i);
85         if (!actual_instruction.isCorrect()) {
86             System.out.println("Instruction number " + (i + 1) + " has error code: " + actual_instr
87         }
88     }

Output - P2 (run)
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
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index: 4, Size: 4
    at java.base/java.util.LinkedList.checkElementIndex(LinkedList.java:559)
    at java.base/java.util.LinkedList.get(LinkedList.java:480)
    at Program.getNextInstruction(Program.java:68)
    at LogoProgram.main(LogoProgram.java:25)
C:\Users\Ojitos_Rizados\AppData\Local\NetBeans\Cache\12.1\executor-snippets\run.xml:111: The following error occurred while executing this line:
C:\Users\Ojitos_Rizados\AppData\Local\NetBeans\Cache\12.1\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 0 seconds)
```