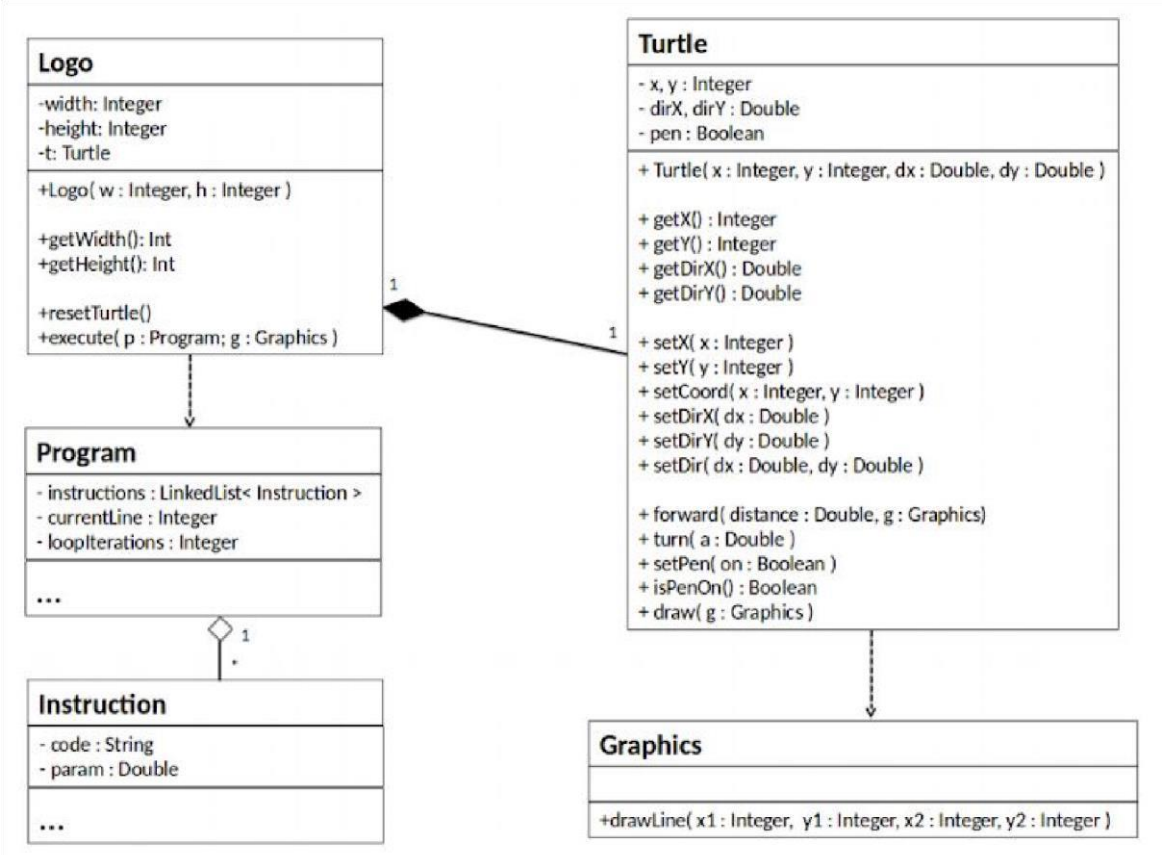
Antonio Pintado U172771

# Report Lab 2

En esta sesión de Prácticas, deberemos implementar el esquema en UML que se nos presentó en la clase de Seminario 2, implementando las clases Logo y Turtle junto a las clases ya vistas en la anterior práctica.

## Introducción

El objetivo es mejorar el programa simple creado en la Práctica 1, añadiendo clases nuevas y una interfaz gráfica (GUI). Usaremos las instrucciones primarias que vimos anteriormente y el esquema propuesto:




## Implementación

Siguiendo el esquema creamos los miembros para cada clase. Los métodos que nos resultan más laboriosos fueron:

Antonio Pintado U172771

1. Método “forward” de Turtle

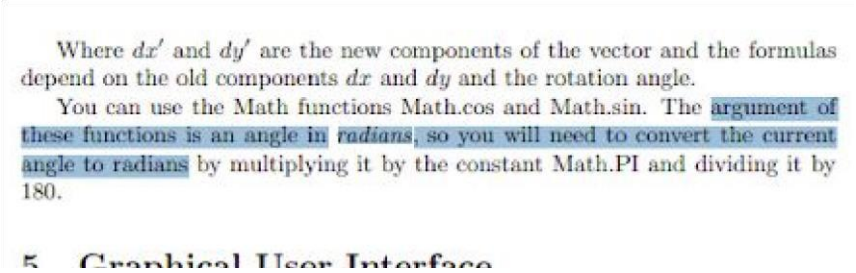
```
64      public void forward(double distance, Graphics g){
65          int xOld = x;
66          int yOld = y;
67          //Las coordenadas de la direccion seran parte de un circulo
68          //unitario, así que usaremos sus componentes multiplicadas
69          //por la distancia para re calcular la posicion
70          this.x = x + (int) (dirX*distance);
71          this.y = y + (int) (dirY*distance);
72          if (isPenOn() == true){
73              g.drawLine(xOld, yOld, x, y);
74          }
75      }
```

En el que debíamos controlar que nuestro turtle se moviera siguiendo su direccion. En concreto calcula la nueva posición en base a la multiplicación de la direccion con la distancia que se quiere mover la tortuga.

2. Método “rotate” de Turtle

```
76      public void turn(double a){
77          //Transformamos el angulo (en el enunciado de la practica hay una errata, dice
78          //que cambiemos de radianes a radianes, y que el argumento de esta funcion esta en radianes
79          //pero la instruccion "ROT" indica la rotacion en grados)
80          double angle= a*Math.PI/180;
81          double dirXold = dirX;
82          this.dirX = Math.cos(angle)*dirXold - Math.sin(angle)*dirY;
83          this.dirY = Math.sin(angle)*dirXold + Math.cos(angle)*dirY;
84      }
```

Al principio nos confundió un poco, ya que en el enunciado de la práctica decía que el parámetro estaba en radianes, pero cuando mirábamos la instrucción que correspondía, ROT, insertaba los ángulos en grados.



Where $dx'$ and $dy'$ are the new components of the vector and the formulas depend on the old components $dx$ and $dy$ and the rotation angle.

You can use the Math functions Math.cos and Math.sin. The argument of these functions is an angle in *radians*, so you will need to convert the current angle to radians by multiplying it by the constant Math.PI and dividing it by 180.

5 Graphical User Interface

En esta parte tuvimos algún percance, por un problema a la hora de representar las funciones, ya que habíamos confundido el coseno con el seno en una de ellas, y el método rotate no hacia su correcto funcionamiento. Pero fue rápidamente solucionado.

3. Método “execute” de Logo

```
36      public void execute(Program p, Graphics g){
37          if ( p.isCorrect ( )){
38              p.restart( ) ;
39              while (true) {
40                  Instruction instruction = p.getCurrentInstruction();
41                  //Consideraremos las instrucciones básicas listadas en el seminario 1
42                  switch (instruction.getCode()) {
43                      case "PEN":
44                          if(instruction.getParam() == 1){
45                              t.setPen(true);
46                          } else {
47                              t.setPen(false);
48                          }   break;
49                      case "FWD":
50                          t.forward(instruction.getParam(), g);
51                          break;
52                      case "ROT":
53                          t.turn(instruction.getParam());
54                          break;
55                      default:
56                          break;
57                  }
58                  if(!p.hasFinished()){
59                      p.getNextInstruction();
60                  }else {
61                      break;
62                  }
63              }
64          }
65          t.draw(g);
66      }
```

Aquí tuvimos que indicar que se diferenciaran las instrucciones para así llamar a los métodos implementados ya en Turtle, siguiendo la lista de instrucciones que ya vimos anteriormente.

Cabe aclarar que uno de los métodos que fue añadido en esta practica fue el de getCurrentInstruction, este método nos fue de gran ayuda ya que, en la primera practica, vimos que faltaba un método para que nuestro programa fuera correcto, ya que solo con el método getNextInstruction el programa quedaba incompleto, esto tenia un problema el cual no llegaba a interpretar del todo la primera instrucción. Sin embargo para esta practica ha sido añadido y este problema queda solventado.

Antonio Pintado U172771

**Conclusiones**

En general, la documentación de la práctica nos ha sido de ayuda, excepto quizás en los puntos 5 y 6, cuando tuvimos que aplicar las clases que ya habíamos implementado en un proyecto en un nuevo proyecto (tuvimos que crear entonces dos proyectos diferentes, fue un poco confuso pasarlo todo luego y que también tuviéramos que usar un LogoProgram main para testear).

Creo que esta parte podría haberse explicado al principio de la práctica, y hay algunos puntos que quedan un poco confusos, como a la hora de inicializar nuestras clases “anteriores” en el LogoWindow, y cómo funciona exactamente un JPanel y donde y como se inicializan exactamente las clases.