Program:

The program class is a set of iterable instructions with its required indices. And with goToStartLoop() it is able to loop by using this function when required, that is when the instruction END is called, to return to the start of the loop, if there has been some error and there is no REP instruction then it does not move.
With the function isCorrect() we are able to check if there is some error on the program, with its instruction equivalent. And if there is some error we are able to print it with printErrors() function, that basically reads all the program and prints everything that has some sort of error, printing the instruction using the info() function to get a String with the instruction code and parameter.
addInstruction(Instruction instruction) function can be used to append a new instruction to the program, first checking if it is correct and then appending it.

Instruction:

We can use isCorrect() to check if the specific instruction is correct or not, this is done by first checking if the instruction is within the list {"REP", "PEN", "ROT", "FWD", "END"} and after that checking if the parameter of the specific instruction is within the required boundaries. If there is an error it is printed on screen as well as returning the number, or ID, of the error, in order to do all we made and use the function errorCode() that checks if the code is one of the previous list and check then the error of the parameter if it is one of those, returning an integer in function of the error, 0 being no errors and from 1 to 5 inclusive indicating some sort of error in the instruction that is later read by the isCorrect() and used to print the error.
With info() function we can get the parameter and code of the instruction in String format if needed. This is done by transforming the parameter through Double.toString(parameter) and since the code is already a String we concatenate them both.
The isRepInstruction() can be used to get true if the current instruction is “REP” or “END” that are the base instructions for the loop or repetition, therefore we check if the instruction is one of those two and return true if it is and false otherwise.

Turtle:

The forward(double distance) can be used to update the position of the turtle by a certain distance, it checks if the pen is on or not, if it is it draws a line until it reaches the new coordinates, otherwise it moves to the same position without drawing.
turn(double a) it updates the angle of the turtle first by getting the angle a, that is on degrees, to radiants and then by using an equation to get the new angles, that is:

$$dirX' = cos(\alpha) * dirX - sin(\alpha) * dirY$$
$$dirY' = sin(\alpha) * dirX + cos(\alpha) * dirY$$

With setPen(boolean on) to set the current state of the pen be the input.
The function draw(Graphics g) we compute the coordinates of the three points of a triangle and get the X on one array and the Y on another, after that we send both and the number of points (n) to g.drawPoligon(Xcoords, Ycoords, n) to draw the triangle.

Logo:

resetTurtle() is used to reset the turtle to its initial configuration, used with clearRect of Graphics to reset the window on the LogoWindow class.
execute(Program p, Graphics g) read the program instruction per instruction and checks which is it and then execute what it is commanded by the instruction. And it uses the draw() function from turtle to draw the turtle on each instruction. If there is some Instruction wrong it is ignored in order to avoid errors on the execution.
togglePen() is used as a function to be used to change the state of the pen regardless of its current state.

LogoWindow:

On the LogoWindow() we initialize the required objects to execute the program properly, for example the Program prog is initialized here. it also checks if there is something wrong on the program before and it prints the errors.
The paint() function is overriding the same function of JFrame. It calls super.paint() to use the overridden function and sets the color to paint, after that it uses the execute(Program program, Graphics g) function of Logo.
There are three buttons initialised btnNewButton, btnNewButton_1 and btnNewButton_2 on code and “Execute”, “Toggle Pen” and “Clear” as how they are shown on window.
The “Execute” button calls directly to the pain(function)
The “Toggle Pen” button calls to togglePen() of Logo.
The “Clear” button uses clearRect(int x0, int y0, int x1, int y1) of Graphics and resetTurtle() of Logo.

We struggled the most with the use of Graphics, we knew that we required a Graphics instance but we did not know how to get one. Finally we discovered that the JFrame class from which we inherited there was a function that let us get a Graphics instance from the window, JFrame.getGraphics(), and we could finally use Graphics and test the graphics dependant functions.

We also enjoyed the Lab as we could use Graphical User Interface (GUI) to get inputs from the user and as well return the outputs through it. Though the errors and Strings were returned through the console. And, even though it gave us some headache, the Graphics class was also enjoyable as we were able to see and make code to Show on the window various shapes.