Lab session 2: Implementing a class diagram
24292-Object Oriented Programming

1 Introduction

The aim of this lab session is to implement the design of Seminar 2 consisting of a complete Logo application. The session is mandatory and you have to deliver the source code of the Java project as well as a document that outlines the solution of the problem. To elaborate the document you can use the guidelines from Lab session 1.

2 Adding the classes Program and Instruction

You should start by incorporating the classes Program and Instruction from Lab session 1. To do so you can either copy them to a new directory, or (optionally) create a package that you can import from your new code. Add a new method to Program.java called getCurrentInstruction() that returns the instruction on the current line of the program.

3 Creation of Turtle and Logo classes

A possible solution design of the first seminar, where Turtle and Logo classes are included, is shown in Figure 1.

3.1 The Turtle class

We will create a new class Turtle. Add all attributes, constructor, setters, getters and other methods specified in the Turtle class design from Seminar 2.

As in Lab session 1, implement a class called LogoProgram with a main method, and test the Turtle class by creating an instance and applying methods on it. Compile and run your code.

3.2 The Logo class

Next, create the Logo class. Add again all attributes, constructor and methods from the design from Seminar 2.

Compile and run the test program again to check the correct implementation.

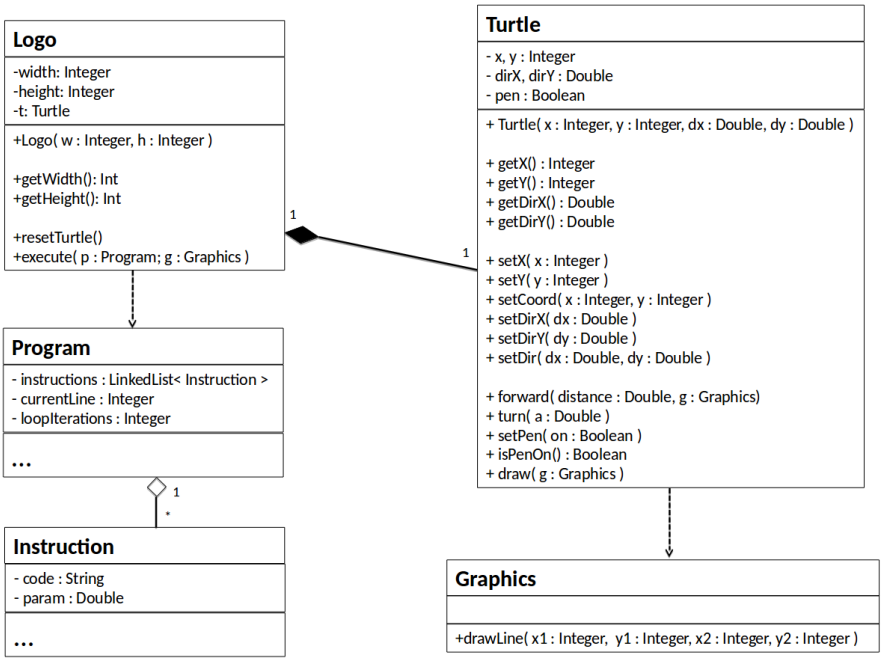

Figure 1: LogoApp class diagram

4 Implementing the Logo and Turtle classes

The next step in the project is to start implementing the different methods of Turtle and Logo classes. Remember to run the application often so that you check everything works.

4.1 Turtle class

4.1.1 Turtle forward method

This method updates the position of the turtle due to a forward instruction. You have to update the position with respect to the current direction and the parameter received as input.

4.1.2 Turtle rotate method

This method updates the direction of the turtle given the input angle in degrees. The formula to rotate vectors is as follows:

$$dx' = cos(\alpha)dx - sin(\alpha)dy$$
$$dy' = sin(\alpha)dx + cos(\alpha)dy$$

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

Since Logo is used to draw figures, it is natural to incorporate the code into a graphical user interface (GUI). As in Lab session 0, we will explain how to do this in Netbeans, though it is possible to use a different editor and write Java code for implementing the GUI.

We will create a Netbeans Java project LogoApp that will be a GUI project.

- Create the project LogoApp and remember from previous lab sessions to **UNSELECT** the create main class option.
- We will then add a JFrame container as in the first session giving it the name LogoWindow. *Reminder*: right-click the LogoApp node and choose New; JFrame Form. When adding a JFrame we are creating a main class LogoWindow that inherits from JFrame.
- We will add finally a JPanel in JFrame as in last session. *Reminder*: click JFrame in the right hand side column. Then put it into the window frame. Extend it to occupy the whole window.

Now switch to the sources view of the LogoWindow pressing the source button. The result after executing should be similar to the one showed in Figure 2.

Run the program and test that everything works well and that the correct main method is executed when clicking on play button.

Search for the LogoWindow constructor that you can find on top of Figure 2 and add the following line after the call to initComponents:

```
setSize( 800, 600 );
```

This sets the window size to the specified by the parameters width and height. The resulting code should be:

```
public LogoWindow(){
    initComponents();
    setSize( 800, 600 );
}
```

Important: Run again the program to see the effect on the window size.

In Netbeans there is no explicit menu to add files to an existing project. Simply copy the files Program.java and Instruction.java to the source folder where the project is stored.

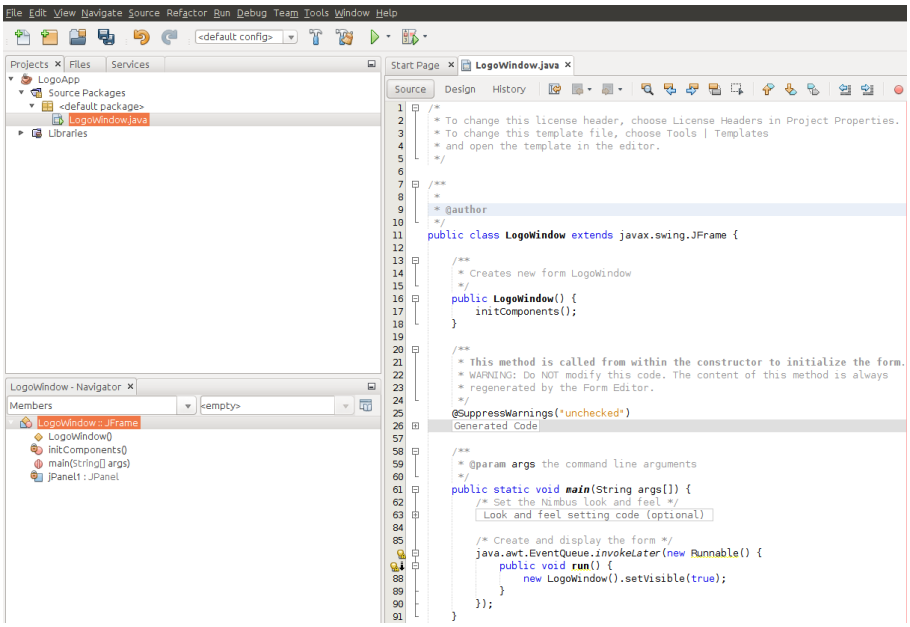

Figure 2: Netbeans interface demo with LogoApp project

6 Changes in the main LogoWindow class

Apply the following comments in the LogoWindow class:

- Add a Logo variable named logo.
- Initialize it in the LogoWindow constructor.
- Change the initial window size setting using the width and height getters from Logo class.
- Declare a Program variable called prog.
- Initialize the program in the LogoWindow constructor.
- Add the necessary instructions to the program for drawing a square.
- Add the function paint as specified in the code below.

```
public class LogoWindow extends javax.swing.JFrame{
    // Attributes declaration
    Logo logo;
    Program prog;

    // Constructor
    public LogoWindow(){
        initComponents();
        // Add here the creation of instances of
        // Logo and Program classes
        // Add to the Program instance the square program
        setSize( logo.getWidth(), logo.getHeight() );
    }

    // Paint method for Graphics
    public void paint( Graphics g ){
        // Call to parent class paint method
        super.paint( g );
        // Add here a call to the execute method
        // of the Logo program
    }
}
```

The method paint is a method that exists in the parent class JFrame and declaring it again means that we are *overwriting* it. We then add a call to the method paint of its parent classes with a keyword `super` that we saw for calling the constructor of the parent class; but now its used to call the method paint of the parent class: super.paint(g).

The method paint that we just added uses the class Graphics which, if it is not found in compilation, you have to make available by adding:

```
import java.awt.Graphics;
```

in the beginning of the file. At this point, run and check again the program works well.

6.0.1 Turtle draw method

The draw method of the turtle class will draw a triangle in the screen at the position of the turtle and facing its direction. For this purpose we will use the method drawPolygon of class Graphics. Remember that when using the Graphics class the corresponding import has to be added. The method drawPolygon takes as parameters arrays of the coordinates of the points of the polygon (in our case a triangle) and finally the number of points. You will only need to initialize the needed variables and the rest of the method is given in the code:

```
public void draw( Graphics g ){
    // Declare and Initialize all the necessary variables
    // We set the x and y coordinates in a triangular shape
    xc[0] = (int)(x + 8 * dirY); yc[0] = (int)(y - 8 * dirX);
    xc[1] = (int)(x - 8 * dirY); yc[1] = (int)(y + 8 * dirX);
    xc[2] = (int)(x + 16 * dirX); yc[2] = (int)(y + 16 * dirY);

    // Graphics method to draw a Polygon where the
    // attributes require the x and y coordinates
    // and the number of points
    g.drawPolygon( xc, yc, nPoints );
}
```

6.1 Logo execute method

You will now add the execute method to the Logo class which will use a program as input and also the Graphics class to draw the result of that program.

```
void execute( Program p, Graphics g ){
}
```

First of all add the code to call the turtle draw method. Then we are going to test that the turtle is shown in the screen by adding a call to execute in the main window class LogoWindow in the method paint. Look again a the code of the paint method shown in Section 5 and add the execute call in the appropriate location.

Important: Run the program and check that the turtle is shown in the screen.

In this method you have to start the program, get instructions and draw them using the graphical primitives provided by the Graphics class. You will only need the g.drawLine(x1, y1, x2, y2) method, which draws a line from one point to another. Draw also the turtle at each instruction to see its movements.

Test that everything works correctly and that a square is drawn in the screen.

6.2 Pen Operations

Now add all the necessary operations to make the pen work making it possible to turn it on and off so that the turtle can go forward with and without drawing.

Test the new operation by making a program that draws two squares in the screen.

7 Submission

You have to submit the Java source code as well as a report describing the implementation. The deadline is before the next Lab session.