Turtle Class Explorer Lab

Overview
In this lab you will continue learning about classes and objects by exploring a part of the class Turtle. You will start learning about the syntax and the meaning of the member function calls. You will further practice working with the graphics window coordinate system by drawing simple line drawings using a LOGO-like Turtle.

You will record your observations and work in a document that will contain snapshots of the program's behavior, your comments, and your calculations. You will also modify the original program by adding the code that will create a drawing that you designed.

Program summary
You will be working with a program TurtleClassExplorer that reports on your work in three different windows. The GUI window is a Turtle Class Explorer that allows you to view the current state of the Turtle class member data and has controls representing all relevant member functions. The graphics window shows the Turtle and its trace as the Turtle object moves and turns. The GUI and graphics are bundled together in one frame. The Console window records all the member function calls made in your program - it shows them as they would appear if they were a portion of the code.

You will have several tasks in this lab. You will first learn how to navigate the Turtle to create interesting drawings. While doing so, you will observe the syntax of member function calls and, the behavior of the Turtle object, and the meaning of its member data. You will then add your creativity by designing a new drawing and navigating the Turtle in creating this drawing. Finally, you will read the code for a portion of the Turtle class and identify the different parts of this class definition.

Brief program description (you do not need to understand this for this lab):
Only one class is included in the project in its source code format. The TurtleExplorer class is responsible for creating the entire GUI. In addition, it creates one Turtle object (called myTurtle) that calls its member functions in response to the selection of appropriate action buttons by the user. These member function calls are also echoed in the Console, and, of course, the Turtle's movement and change of state is reflected both in the BufferedPanel and in the GUI section that shows the contents of the member data.

The Turtle class itself is a part of the Java Power Tools and its compiled code is included in jpt.jar. However, we include both the source code for the entire Turtle class and a compressed version of the Turtle class that contains only the functions that are the focus of this lab. These are given, so you have an easy access to the code you will read.
Snapshot of running application

When the program is running user sees a GUI as shown:

The picture in the Turtle Graphics window has been generated by the following sequence of function calls:

```java
// red
myTurtle.setPaint(255, 0, 0);
myTurtle.step(50.0);
myTurtle.turn(60.0);

// cyan
myTurtle.setPaint(255, 0, 255);
myTurtle.step(50.0);
myTurtle.turn(60.0);

// green
myTurtle.setPaint(0, 255, 0);
myTurtle.step(50.0);
myTurtle.turn(60.0);

// yellow
myTurtle.setPaint(255, 255, 0);
myTurtle.step(50.0);
myTurtle.turn(60.0);

// blue
myTurtle.setPaint(0, 0, 255);
myTurtle.step(50.0);
myTurtle.turn(60.0);

// magenta
myTurtle.setPaint(0, 255, 255);
myTurtle.step(50.0);
myTurtle.turn(60.0);

// black
myTurtle.setPaint(0, 0, 0);
```
Just like in the Picture lab, the action buttons correspond to the appropriate member function calls. The Turtle behaves in a manner similar to the Turtle in LOGO programming language used in elementary schools. Several commands control the movement of an imaginary turtle - here represented as a small rectangle: user can tell the turtle to move forward a given distance and to turn by a given angle. With the pen down, the turtle draws a trail as it moves. User can select the turtle's color, and can choose to hide the turtle, leaving the entire turtle trail exposed.

The draw action in the TurtleExplorer class is designed, so that the programmer can insert code for an arbitrary drawing to be created directly - without the interactive step-by-step user control. Initially this function does not do anything interesting. You will insert here the code for a drawing that you designed.

The clear button erases the graphics window, but leaves the Turtle state unchanged. The reset button rests the Turtle state to the original default state, but does not erase the drawing in the graphics window.

There are four helper buttons beneath the Member Data View. Three of them draw simple shapes and you can study the code to see how these shapes were created. The fourth function will draw a light colored grid in the window, so you can adjust your coordinates during the design time.

Lab Tasks
In this lab you have to perform the following tasks.

1. Learn the functionality of all action buttons and their effect on the Turtle object behavior and create a couple of simple turtle drawings.
2. Create your own turtle drawing, record the sequence of member function calls, and modify the draw function so it displays your picture.
3. Study the code for the Turtle class and highlight the different parts.

Task 1: Understanding the behavior of Turtle class

1. Try the step, turn, and setPaint buttons and describe what they do. Draw a simple drawing (a square, a simple polygon, ...) with more than one color and copy the code from the Console into your report.
2. Experiment with showTurtle, hideTurtle, setPenState(PEN_UP), and setPenState(PEN_DOWN) functions and describe what they do.
3. Try the Triangle, Square, and Hexagon buttons. Study the code that performs the drawings. Now create a regular pentagon and include a snapshot and the code in your report.
4. Create the following drawing and include the code and a full snapshot in your report. You always get the code by copying the transcript in the Console.
   Hint: Plan your work first, write down what you want to do, the order in which you want to do things. Remember to put the pen down or up as needed.
Task 2: Create your own Turtle drawing

5. Now that you understand how the turtle behaves and how some simple drawings can be created, it is time to use your creativity. Start by drawing your design on a graph paper. Record the coordinates of all points where the turtle needs to turn, or change color, or change the pen state. Write down the sequence of the function calls that need to be made to create your drawing. Include your drawing and design in your lab report. These can be sketched by hand.

6. Now, run the application, and perform your actions. The function calls will be recorded in the Console. Copy the console contents into the designated place in the draw function. Run the program again and check that it creates your drawing correctly. Include both the code and a snapshot in your report.

Task 3: Read Turtle class code

The code for TurtleX class is not really a complete class definition. It has been extracted from the real Turtle class, but some constants and member data have been deleted, and most of the static member functions had their code removed.

Static member functions play a special role in this class and will be discussed in lectures. They represent the collective behavior of all objects in this class, and are not called by specific object instances. For example, the program may define several Turtle objects in the same BufferedPanel and the static member data and member functions make it possible to keep track of all of them.

7. Highlight in pink the names of all static member data (not constants!!) and the names of all static member functions. Pick two static member data and two static member functions and write down what your think is their purpose.
The member data needed for each instance of a Turtle object are defined as protected. This is similar to private, but is useful, if we are defining a new class that extends the Turtle class but want to do things a little bit differently.

8. Highlight in yellow the names and types/classes of all non-static member data. Explain the reason for selecting each type of class for each member data item.

The TurtleX class has more member functions than those we used in our exploration. Highlight again in yellow the headers of those (non-static) member functions that were called from the TurtleExplorer GUI.

9. Highlight in another color the headers of all get non-static member function - specify the color in your report.

10. Highlight in a fourth color the headers of the remaining non-static member functions and the constructor header.

You can see that it is possible to understand the role of most of the member functions - even though you really do not know yet how to write them.

You also are learning that the set and get functions (called accessors) play an important role in all class definitions, even though a lot of their work is quite routine.

**Hand in:**

- The lab report (paper version), including the hand-written drawing design and outline of the code for the drawing.
- The highlighted copy of the TurtleX.java class printout.
- Electronic submission of the class TurtleExplorer.java that contains the draw function modified to create your drawing and of the lab report.