This lab uses the Picture class on the class website to draw pictures with loops. It is a preparation for the next lab wherein you will build and draw a simple type of fractal called Lindenmayer Systems. The line drawing above is an L-system created with the same commands used in this lab together with the auto string building techniques you will implement in the next lab.

1) Create a new, project with empty src and bin directories.
2) Download the Picture class from the class website and save it in the src directory of your new project.
3) Right-click on the project name and select "refresh".
4) Right-click on the src directory of the new project and select: new→class. This class must contain the main method and is where you will add your own code.
Example class that draws the simple image below:

```java
1) //Listing 1: Draw an X
2) import java.awt.Color;
3) import java.awt.Graphics;
4) public class FunWithDrawing
5) {
6)   public static void main(String[] args)
7)   {
8)       Picture myPic = new Picture(100,50);
9)       Graphics canvas = myPic.getOffScreenGraphics();
10)      canvas.setColor(Color.RED);
11)      canvas.drawLine(0, 0, 99, 49);
12)      canvas.setColor(Color.BLUE);
13)      canvas.drawLine(99, 0, 0, 49);
14)      myPic.repaint();
15)   }
16) }
```

More information on how to use the `Graphics` class, can be found in section 1.4, section 4.3, and section 5.4 of the textbook.

Additional information of `java.awt.Graphics` can be found on the web:

Full JavaDoc of `java.awt.Graphics`:

`http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html`

Getting Started with Graphics Tutorial:

`http://docs.oracle.com/javase/tutorial/2d/basic2d/index.html`

In this lab, we mix the graphics from section 1.4, with the String class form chapter 2, `if` statements from chapter 3 and `while` loops from chapter 4.
Listing 2 draws a set of vertical lines. Listing 3, uses a while loop to draw more vertical lines with less code.

```
1) //Listing 2: Vertical Lines
2) import java.awt.Color;
3) import java.awt.Graphics;
4) public class FunWithDrawing
5) {
6)   public static void main(String[] args)
7)   {
8)       Picture myPic = new Picture(300,200);
9)       Graphics canvas = myPic.getOffScreenGraphics();
10)      canvas.setColor(Color.WHITE);
11)      canvas.fillRect(0, 0, 300, 150);
12)      canvas.setColor(Color.BLUE);
13)      canvas.drawLine(0, 0, 0, 199);
14)      canvas.drawLine(10, 0, 10, 199);
15)      canvas.drawLine(20, 0, 20, 199);
16)      canvas.drawLine(30, 0, 30, 199);
17)      canvas.drawLine(40, 0, 40, 199);
18)      canvas.drawLine(50, 0, 50, 199);
19)      canvas.drawLine(60, 0, 60, 199);
20)      canvas.drawLine(70, 0, 70, 199);
21)      myPic.repaint();
22)   }
23) }
```

Why are parts of the screen captures above black?

In each listing, what change in the code would make all of the background white?
How could you change the code above to draw horizontal lines?

How could diagonal lines be drawn?

What would be drawn if line 17 of listing 3 is replaced with:

```java
canvas.drawLine(0, x, x, 199);
```

It is a small change, but it makes a very different image. Try to draw out the first few lines on paper. Then guess the pattern. Once you think you have it, make the change and run.
Your task is to write a program that displays JOptionPane.showInputDialog asking the user for a string command, checks that string for errors and if none are found, draws the image encoded in the string within a 600x600 pixel Picture.

Your program must implement a "turtle" by keeping track of 4 turtle variables:

<table>
<thead>
<tr>
<th>int x</th>
<th>Location of the turtle in the x-direction in the image. Default is 300.</th>
</tr>
</thead>
<tbody>
<tr>
<td>int y</td>
<td>Location of the turtle in the x-direction in the image. Default is 300.</td>
</tr>
<tr>
<td>int heading</td>
<td>Direction the turtle is currently facing (0 = up, 1 = right, 2 = down, 3 = left). Default is 0</td>
</tr>
<tr>
<td>Color penColor</td>
<td>Color the turtle uses to draw. Default is BLACK</td>
</tr>
</tbody>
</table>

Your program must recognize the following commands:

| h | draw a straight line segment 10 pixels long in the current heading. |
| f | draw a straight line segment 10 pixels long in the current heading. |
| g | move (without drawing) 10 pixels in the current heading. |
| + | Turn the handing clockwise 90° |
| - | Turn the handing counter-clockwise 90° |
| K | Change the turtle color to BLACK |
| R | Change the turtle color to RED |
| G | Change the turtle color to GREEN |
| B | Change the turtle color to BLUE |
| C | Change the turtle color to CYAN |
| O | Change the turtle color to ORANGE |
Example 1: input string = "h+Rhh+Ghhh+Bhhhh+Chhhhh"

Example 2: input string = "Bffff+ffff+ffff-ffff"

The next project will have the user not inputting a string of commands, but a set of rules for automatically building very large strings of commands that form cool fractal patterns. Also, in the next project, turns will not always be 90 degrees.

Grading Rubric [20 points total]:

1) [1 point]: Attached one file in Blackboard Learn with the file name: DrawCommands_yourName.java.

2) [4 points]: When given the two examples above, your program produces the output shown above (only yours will have a lot more empty space around the figure). Two points for each test case.

2) [10 points]: When given an unknown test case, your program draws the correct image. There will be 10 unknown cases: one point each.

3) [1 point]: When the user clicks "cancel" on an input dialog, your program exits.

4) [4 points]: When the user enters invalid input, your program must display an error dialog. Then, when the user closes the error dialog, your program must loop back to the original input dialog. Invalid input is any string containing a
command not in the command list. Note: the command list is case sensitive. Thus, 'r' is an error, yet 'R' is a command.

Note: Requirements verses Implementation
The examples show if statements and while loops. The solution I post and go over in class will use if and while. This, however, is an implementation suggestion, not a requirement. You, for example, may use case statements and for loops. Be careful of "fall through" if you decide to try case statements.

Similarly, variable names used above for the turtle state, (x, y, heading and penColor), are implementation details, not requirements. You may name your variables as you see fit - so long as you are in keeping the naming conventions. For example, variable names must start with a lowercase letter.

Penalties:
[-5 points]: Code does not adhere to the hallowed CS-259 coding standard. This includes indenting, class, method, and in-line comments, removing all warnings, etc.