Programming Assignment 1: Turtle Graphics

Due: 11:59pm, Thursday, October 9

Overview

You will use simple turtle graphics to create a three-line drawing consisting of the following text, where XX should be replaced with the two letters **from your own CS11 login id**:

```
CS11FXX
FALL
2014
```

All letters will be drawn in upper-case.

Setup

Open a new Linux terminal window. In your home directory, create/make a new directory called `pa1` (mkdir) and go (change directory - cd) into that directory (in Unix, the tilde ~ expands to your home directory):

```
mkdir ~/pa1
cd ~/pa1
```

Copy the `turtleClasses.jar` file from the public directory that will provide the drawing environment for this assignment. Notice the single dot at the end of this command. The single dot means your current working directory, which is your `pa1` after you executed the `cd` command above. The dot dot means the parent directory (one up).

```
cp ~/../public/turtleClasses.jar .
```

Instructions

You should create a Java file called **`CS11Turtle.java`** under your `~/pa1` directory. This is the only file that needs to be turned in with this assignment. This file will contain all the Java code that will run your program. Use vim or gvim to edit the file. We will not be using any IDEs, like DrJava or Eclipse for this class. (It is your responsibility to make sure all your programs compile on the lab computers.)

To compile your code, use the following command at the Linux command line prompt (-cp is short for -classpath):

```
javac –cp ./turtleClasses.jar:. CS11Turtle.java
```

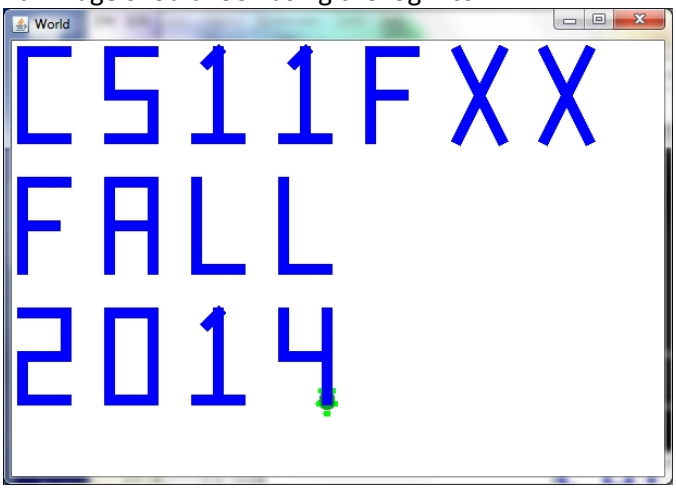
To run your program, use the following:

```
java –cp ./turtleClasses.jar:. CS11Turtle
```

Your code will use classes from `turtleClasses` library including classes `World` and `Turtle`. `World` is the drawing canvas and `Turtle` is the "pen" that will do the drawing. `Turtle` extends `SimpleTurtle`, which is another class in `turtleClasses`. You will only be dealing with methods of `SimpleTurtle` to do all of your drawing. The Javadoc API for this class `SimpleTurtle`, and other classes in `turtleClasses`, can be accessed by the appropriate link in the Useful Links section of the class home page. You should definitely take a look at the API for `SimpleTurtle` as all the needed drawing methods are covered there. Note on the `SimpleTurtle` methods `turnLeft` and `turnRight`: these methods do not make the turtle face in the left or right direction. Rather, they **rotate** the turtle 90 degrees counterclockwise or clockwise, respectively.

Use the code at the end of this write-up as a starting point for `CS11Turtle.java`. The code for drawing the letter 'C' in `CS11FXX` is provided for you. Do not change the size of 600x400 pixels for the World. Each character should be drawn 40 pixels wide and 80 pixels high. There should be 40 pixels of space between each character and between each line of text. The first letter, 'C', is positioned so that its top-left corner is at the position (10, 10). Use constants defined.

Here is an example of how the final image should look using the login cs11fxx:



Remember: **Use your own login id instead of CS11FXX**. As in the template provided, you should write additional methods to draw the other characters in the text and call these methods from `main()`. Note: if a particular character appears multiple times in the text (such as '1' in CS11 and in 2014), do NOT write separate methods for each instance of the character. Use just one method and pass in the appropriate x and y values to the method. These x and y coordinates should correspond to the top-left corner of the bounding box for the drawn character.

Use the code given for 'C' as a guide for drawing the other letters, but feel free to use other methods from `SimpleTurtle` that you may find useful. One method in particular you may find useful to draw line segments at an angle (such as the at the top of '1') is `turn()`, which turns the turtle by a specified number of degrees. There is no specific, required style to draw your letters. As long as it can be clearly understood what each character actually is and the width and height requirements are observed, you are free to get creative.

Extra Credit (Both 1 and 2 below)

1) Draw the text so that it is centered both vertically and horizontally in the 600x400 World, as in the example below. **And**,
2) Add something creative and unique. You can change the colors, draw stars and robots, add highlighting, etc.

Style Requirements

You will be graded for the style of programming on this assignment. A few suggestions/requirements for style are given below. These guidelines for style will have to be followed for all the remaining assignments. Read them carefully. In the template code provided below for this assignment, all of these requirements are met (replace comments appropriately).

- Use reasonable comments to make your code clear and readable.
- Use class header and method header block comments to describe the purpose of your program and methods (see below).
- Use reasonable variable names that are meaningful.
- Use `static final` constants to make your code as general as possible. No hardcoding constant values inline.
- Judicious use of blank spaces around logical chunks of code makes your code much easier to read and debug.
- Keep all lines less than 80 characters. Use 2-3 spaces (not tabs) for each level of indentation. Make sure each level of indentation lines up evenly.
- Every time you open a new block of code (use a '{'), indent farther. Go back to the previous level of indenting when you close the block (use a '}').
- Always recompile and run your program right before turning it in, just in case you commented out some code by mistake.

Hints and Suggestions

- Go to Discussion Section!
- Draw out the character by hand on graph paper before coding it.
- Test your code in small increments. Write code to draw one character at a time, test it, and then move on to the next character.
- Each time you run your program close the Java window with your drawn characters. You should get your command line prompt back in your terminal window.
- Look at the Java API for `SimpleTurtle`
- **Use your own login id**. Remember to replace "XX" with the letters from your own cs11 login id. If you use “XX” or someone else’s login id, you will be docked one grade letter.

Turnin Instructions

Remember the deadline to turn in your assignment is **Thursday, October 9, by 11:59pm**.

Make sure the program works correctly in your cs11f login on the workstations in the labs in the basement of the CSE building when the workstations are booted under Linux.

When you are ready to turn in your program in, type in the following command and answer the prompted questions

```
turnin pa1
```

You can verify your turnin with the following command

```
verify pa1
```

It is your responsibility to make sure you properly turned in your assignment.

NO LATE ASSIGNMENTS ACCEPTED! NO EXTENSIONS! NO EXCUSES!
DO NOT EMAIL US YOUR ASSIGNMENT.
Start Early and Have fun!

CS11Turtle.java

```
/*
 * Name:  Jane-Joe Student  <<< --- Replace with Your Name
 * Login: cs11fXX  <<< --- Use your cs11f course-specific account name
 * Date:  Month Day, Year
 * File:  Name of this file, for example: CS11Turtle.java
 * Sources of Help: ... (for example: names of people/tutors/students, books, websites, etc.)
 *
 * Describe what this program does here.
 */

import turtleClasses.*;
import java.awt.Color;

/*
 * Name:       Class name
 * Purpose:    Briefly describe the purpose of this class
 */

public class CS11Turtle extends Turtle {
    private final static int CHAR_WIDTH = 40;
    private final static int CHAR_HEIGHT = 80;
    private final static int PADDING_BETWEEN_CHARS = 40;
    private final static int PADDING_BETWEEN_LINES = 40;
    private final static int CHAR_SPACING = CHAR_WIDTH + PADDING_BETWEEN_CHARS;
    private final static int LINE_SPACING = CHAR_HEIGHT + PADDING_BETWEEN_LINES;

    private final static int START_X_1 = 10; // starting x offset for line 1
    private final static int START_X_2 = 10; // starting x offset for line 2
    private final static int START_X_3 = 10; // starting x offset for line 3
    private final static int START_Y = 10;   // starting y offset

    private final static int PEN_WIDTH = 10;
    private final static Color PEN_COLOR = Color.BLUE;

    private final static int WORLD_WIDTH = 600;
    private final static int WORLD_HEIGHT = 400;

    /*
     * Delay between turtle actions (turns, moves) in milliseconds.
     * 1000 = 1 sec.  so  200 = 0.2 sec.
     */
    private final static int DELAY = 200;

    /*
     * Name:       Constructor name
     * Purpose:    Briefly describe the purpose of this constructor
     * Parameters: List all parameters and their types and what they represent.
     *             If no parameters, just state None.
     */

    public CS11Turtle(World w, int delay) {
        super(w, delay);  // Invoke superclass's ctor to create this turtle
    }                     // on World w with delay of each turtle's action.

    /*
     * Name:       Method name
     * Purpose:    Briefly describe the purpose of this method
     * Parameters: List all parameters and their types and what they represent.
     *             If no parameters, just state None.
     * Return:     Specify the return type and what it represents.
     *             If no return value, just specify void.
     */

    private void drawC(int x, int y) {
        penUp();
        moveTo(x, y);  // always start in upper left corner of this char block
        turnToFace(getXPos() + 1, getYPos()); // face right
```

```
        penDown();
        forward(CHAR_WIDTH);
        backward(CHAR_WIDTH);
        turnRight();
        forward(CHAR_HEIGHT);
        turnLeft();
        forward(CHAR_WIDTH);
    }

    /*
     * Name:       Method name
     * Purpose:    Briefly describe the purpose of this method
     * Parameters: List all parameters and their types and what they represent.
     *             If no parameters, just state None.
     * Return:     Specify the return type and what it represents.
     *             If no return value, just specify void.
     */

    private void drawS(int x, int y) {
        penUp();
        moveTo(x, y); // always start in upper left corner of this char block
        turnToFace(getXPos() + 1, getYPos()); // face right

        /* TODO: Complete this part to draw the 'S' character */
    }

    /* TODO: Add any remaining methods to draw all the other characters */


    /*
     * Name:       Method name
     * Purpose:    Briefly describe the purpose of this method
     * Parameters: List all parameters and their types and what they represent.
     *             If no parameters, just state None.
     * Return:     Specify the return type and what it represents.
     *             If no return value, just specify void.
     */

    public static void main(String [] args) {
        int startX1 = START_X_1,  // starting x offset for line 1
            startX2 = START_X_2,  // starting x offset for line 2
            startX3 = START_X_3;  // starting x offset for line 3
        int startY  = START_Y;    // starting y offset

        int x, y;

        World w = new World(WORLD_WIDTH, WORLD_HEIGHT);
        CS11Turtle myTurtle = new CS11Turtle(w, DELAY);

        myTurtle.setPenWidth(PEN_WIDTH);
        myTurtle.setPenColor(PEN_COLOR);

        myTurtle.drawC(x = startX1, y = startY);
        myTurtle.drawS(x += CHAR_SPACING, y);

        /* TODO: Complete this main method to draw the remaining text */
    }

}  // End of public class CS11Turtle extends Turtle
```