Programming Assignment 3 (100 Points)

Due: 11:59pm Thursday, October 31 **START EARLY!**

This programming assignment has two programs: 1) a Java application (Triangles) that displays alternating triangle patterns side-by-side on the console outputting individual '*' and ' ' (space) characters using nested while loops, and 2) a Java applet extending the Mickey program from PA2 to be more Object-Oriented and add some functionality. **START EARLY!**

README (20 points)

You are required to provide a text file named **README,** NOT Readme.txt, README.pdf, or README.doc, with your assignment in your pa3 directory. There should be no file extension after the file name "**README**". Your README should include the following sections:

Program Description (6 points) : Explain how the user can run and interact with each program. What sort of inputs or events does it take and what are the expected outputs or results? How did you test your program? How well do you think your program was tested?

Write your README as if it was intended for a 5 year old or your grandmother. Do not assume your reader is a computer science major. **The more detailed the explanation, the more points you will receive.**

Short Response (14 points) : Answer the following questions:

1. How do you jump to a certain line number in your code with a single command in vim? For example, how do you jump directly to line 20?
2. What command sets auto-indentation in vim? What command turns off (unsets) auto-indentation in vim?
3. What is the command to undo changes in vim?
4. In vim, in command mode, how do you move forward one word in a line of code? Move back one word?
5. How can you remove all .class files in a Unix directory with a single command?
6. How do you remove a Unix directory?
7. What is the command to clear a Unix terminal screen?

STYLE (20 points)

Please see previous programming assignment for details on Coding Style.

CORRECTNESS (60 points, 30 points each program)

All of your code/files for this assignment need to be in a directory named **pa3** in your cs11w home directory. Please see previous programming assignments to setup your **pa3** directory, compile, and run your programs. Remember we run/execute applications differently than applets. **START EARLY!**

Program 1 - Triangles: Write a Java application (**Triangles.java**) that displays the following triangle patterns side-by-side using nested **while** loops. Everything can be in main().

- Use class **Scanner** to read input.
- Allow the user to input the number of rows defining the size of the side-by-side triangles. No hardcoding or magic numbers in your code.
- Make sure your prompt and output messages match the sample prompt and messages word-for-word.
- Each loop must be based on the current row number/iteration of the outer-most loop.
- Hint: Treat each of the four triangle patterns as its own set of nested loops. Each row of each triangle pattern is made up of some number of '*' and ' ' (space) chars.
- Note there is a space between adjacent triangles.
  To get a better understanding, if size of the triangle is 5, here's the spacing (represented in a grid, for a clearer picture of the individual '*' and ' ' characters).

*						*	*	*	*	*		*	*	*	*	*						*
*	*					*	*	*	*				*	*	*	*					*	*
*	*	*				*	*	*						*	*	*				*	*	*
*	*	*	*			*	*								*	*			*	*	*	*
*	*	*	*	*		*										*		*	*	*	*	*

- All asterisks ('*') and spaces (' ') must be printed a single character at a time using **System.out.print('*')** or **System.out.print(' ')** controlled by the nested while loops (No Strings!). You are not allowed to use a String or StringBuilder or StringBuffer or any data structure to build each line to be output.
- Formatting is important for this program. If your output does not match the required output character-for-character, **no credit** will be awarded. Note there is no space at the end of each line.

- Only valid integers will be entered, but you must check for integers less than 2 and report the **exact error message** as shown in the example below.

Example Executions (user input in **BOLD**):

```
Example 1:
java Triangles
Enter the size of the triangles to display: 12

*            ************ ************            *
**           ***********   ***********           **
***          **********     **********          ***
****         *********       *********         ****
*****        ********         ********        *****
******       *******           *******       ******
*******      ******             ******      *******
********     *****               *****     ********
*********    ****                 ****    *********
**********   ***                   ***   **********
***********  **                     **  ***********
************ *                       * ************
```

```
Example 2:
java Triangles
Enter the size of the triangles to display: -1
Triangle size must be > 1; Try again.

Enter the size of the triangles to display: 0
Triangle size must be > 1; Try again.

Enter the size of the triangles to display: 5

*     ***** *****     *
**    ****   ****    **
***   ***     ***   ***
****  **       **  ****
***** *         * *****
```

Program 2 - Mickey:

This program will build off of your last Mickey program, so you should make sure that program is working correctly first. Being more Object-Oriented, we will separate the GUI controller and the Mickey object.

In this program you will be adding a feature that allows you to flip the Mickey upside-down if the mouse is pressed on the Mickey for longer than 500 milliseconds (1/2 second) on a mouse click. Once the Mickey is upside-down, if the mouse press on a mouse click on the Mickey is longer than 500 milliseconds, the Mickey should flip right-side up again. A long mouse press on a mouse click **not** on Mickey will not flip Mickey. Recall: a mouse click event is fired if there is a mouse press event followed by a mouse release event with no mouse movement between the press and release.

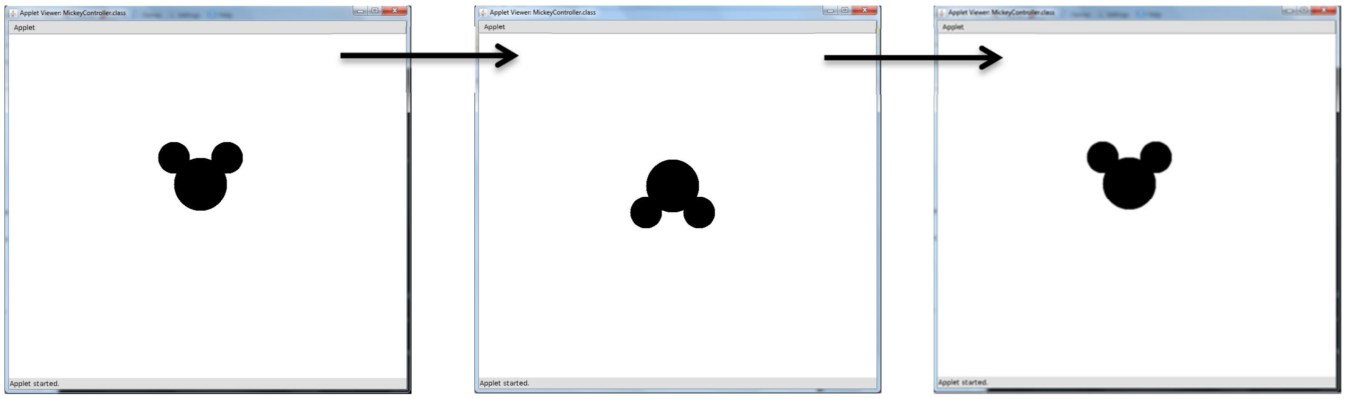
A mouse press followed by moving the mouse one pixel and then a mouse release does not fire a mouse click event.

For example:

How the Mickey looks when you place it with initial mouse click:

How the Mickey looks after holding the mouse button down more than ½ second on a mouse click while the mouse is on Mickey:

Click on Mickey again holding the mouse button down more than ½ second to return it upright:



You will need three files for this program:

- Mickey.html
- Mickey.java
- MickeyController.java
- Timer.java

Mickey.html - change the applet tag's code attribute to be MickeyController.class

MickeyController.java - this is the main GUI controller class that handles all mouse events and controls what we see on the canvas. This class needs to extend WindowController. Methods defined in this class include: begin(), onMouseClick(), onMouseDrag(), onMousePress( ), onMouseRelease(), onMouseExit( ), onMouseEnter( ). Where appropriate, code in this class will create an instance of a Mickey and send messages to this Mickey object (tell the Mickey figure to flip, check if the mouse Location is contained in the Mickey figure, tell the Mickey figure to move, tell the Mickey figure to remove itself from the canvas).

Mickey.java - this class defines what a Mickey figure is (the 3 filled ovals) and what a Mickey figure can do (what messages it can respond to). This class will define a constructor to initialize a new Mickey object placing it on the canvas back in the MickeyController along with the methods to determine if the mouse pointer is contained in the Mickey figure, move the Mickey figure some delta, remove the Mickey figure from the canvas, and flip the Mickey figure.

Timer.java - this class calculates timing between events. Just copy the code on page 168 of the textbook for the class Timer.

It is important to note the difference between MickeyController and Mickey. Mickey class should have all the relevant variables and methods specific to each Mickey created on canvas (in this case we only create one). The GUI controller class, MickeyController, handles all the activities that occur on the canvas – this includes the mouse interactions/events with the canvas and Mickey object.

Since both classes interact with each other, the MickeyController (GUI) needs to have a reference to a Mickey and the Mickey needs a reference to the GUI canvas in the MickeyController. This is done by passing the canvas as an actual argument to the Mickey constructor when the MickeyController creates a Mickey object. Examples in Chapter 6 of the textbook show how to do this.

Specifications:

- The program should still have same functionality as was specified in PA2 (able to create a Mickey on the canvas and drag it around).
- A skeleton example of MickeyController.java:

```
import objectdraw.*;

public class MickeyController extends WindowController
{
  private Text instr1, instr2, instr3;
  private static final int INSTR1_X = 50;
  private static final int INSTR1_Y = 50;
  private static final int INSTR2_X = INSTR1_X;
  private static final int INSTR2_Y = INSTR1_Y + 20;
  private static final int INSTR3_X = INSTR1_X;
  private static final int INSTR3_Y = INSTR2_Y + 20;

  private static final int FLIP_PRESS_THRESHOLD = 500; // Half a second
  private Timer timer;

   …

  // additional variables you might need and begin() and the event handling methods
}
```

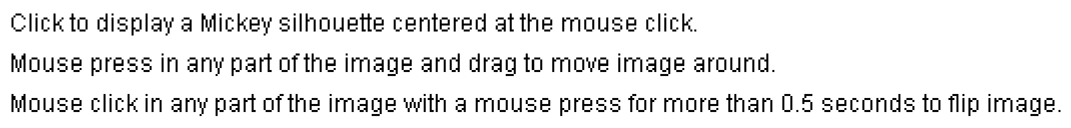
- A skeleton example of Mickey.java

```
import objectdraw.*;

public class Mickey
{
  private FilledOval leftEar, rightEar, face;
  private static final int FACE_RADIUS = 50;
  private static final int EAR_RADIUS = 30;
  private static final int EAR_OFFSET = 50;  // Center of each ear is this
                                             // offset up and over (x and y)
                                             // from center of face.
  private DrawingCanvas canvas;

  …

  // additional variables you might need and …

public Mickey( Location point, DrawingCanvas canvas ) // ctor

public boolean contains( Location point )

public void move( double dx, double dy )

public void removeFromCanvas()

public void flip()

}
```

- Use the **Timer** class in MickeyController to determine how long the mouse button was pressed:
  - Create a Timer object in begin()
  - When the mouse is **pressed**, the timer should **reset** (start the timer).
  - When the mouse is **released**, the timer should get the elapsed time in milliseconds and if the mouse was on the Mickey (grabbed) and the elapsed time was longer than ½ second (500 milliseconds) then flag that the Mickey figure should be flipped if this was indeed a mouse click (vs. a drag).
  - If mouse click event handler send the flip message to the Mickey object if it should be flipped. Note the mouse click event will be fired after a mouse press event and mouse release event with no mouse motion between the press and release.
  - The Mickey should **not flip** if the mouse is **dragged** or if the mouse click is **not on the Mickey**.
- When the mouse is **clicked**, if the mouse was pressed for enough time, the Mickey should **flip** upside-down. Otherwise the Mickey does not change.
- You should use the following constant to compare with the elapsed time Mickey was pressed:

```
private static final int FLIP_PRESS_THRESHOLD = 500; // Half a second
```

- The starting text of your program should be adjusted look like the following (it should disappear when a Mickey is placed, just like in your last assignment):



Click to display a Mickey silhouette centered at the mouse click.
Mouse press in any part of the image and drag to move image around.
Mouse click in any part of the image with a mouse press for more than 0.5 seconds to flip image.

EXTRA CREDIT (5 points)

Every time the Mickey flips, change the color of your Mickey. It should rotate through the following colors: BLACK, RED, ORANGE, YELLOW, BLUE, GREEN, MAGENTA, BLACK. Remember, your Mickey should always start off as BLACK. Then, every time a flip is triggered, you will rotate through each color in the order given and return back to BLACK, and so on.

If you do the extra credit correctly, your Mickey should always change to a new color each time it is flipped either upside down or back right side up.

Do not use any data structures like an array or ArrayList. Keep track of which color to use with a variable that should range between 0 (for BLACK) and 6 (for MAGENTA). Use a **switch** statement to determine which color to use. Increment this variable with each flip and mod the incremented value by MAX_COLORS (7) to keep it in the range of 0-6.

Turnin

To turnin your code, navigate to your home directory and run the following command:

```
> turnin pa3
```

You may turn in your programming assignment as many times as you like. The last submission you turn in before the deadline is the one that we will collect.

Verify

To verify a previously turned in assignment,

```
> verify pa3
```

If you are unsure your program has been turned in, use the verify command. **We will not take any late files you forgot to turn in.** Verify will help you check which files you have successfully submitted. It is your responsibility to make sure you properly turned in your assignment.

Files to be collected:

- Triangles.java
- Mickey.java
- MickeyController.java
- Mickey.html
- Timer.java
- objectdraw.jar
- README

NO LATE ASSIGNMENTS ACCEPTED.

DO NOT EMAIL US YOUR ASSIGNMENT!

Start Early and Often!