Programming Assignment 3 (100 Points)

Due: 11:59pm Thursday, October 19

START EARLY!

This programming assignment has two programs:
1) a Java application (HourGlass) that displays an hourglass shape on the console outputting individual hourglass and filler surrounding characters one character at a time using nested loops, and
2) extending the Mickey program from PA2 to be more Object-Oriented and add some additional functionality.

START EARLY!

README (10 points)
You are required to provide a text file named README, NOT Readme.txt, README.pdf, or README.docx, etc. 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 (3 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 (7 points): Answer the following questions:
Vim related 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? (Not arrow keys)
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? (Not arrow keys)

Unix/Linux related Questions:
5. How can you remove all .class files in a Unix directory with a single command?
6. How do you remove a Unix directory? Define the commands you would run on the terminal when the directory is empty and when it is not empty. How do these command(s) differ?
7. What is the command to clear a Unix terminal screen?

STYLE (20 points)
Please see PA2 write-up for details on Coding Style.
CORRECTNESS (70 points; 35 points each program)
All of your code/files for this assignment need to be in a directory named pa3 in your cs11f home directory. Please see previous programming assignments to setup your pa3 directory, compile, and run your programs.

START EARLY!

Program 1 - HourGlass:
Write a Java application (HourGlass.java) that displays the following hourglass pattern using nested loops. Everything can be in main(), but you can write additional methods within class HourGlass if you like. The only constraint is each character must be output one character at a time in a loop.

- Read command line arguments for the size of the hourglass with optional hourglass and filler chars
- No hardcoding or magic numbers in your code.
- Each loop should be based on the current row number/iteration.
- Each character is output individually one character at a time. Do not use a String with the pattern. You must calculate how many times to loop printing a single hourglass char at a time and how many times to loop printing a single filler char at a time.
- 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.

To compile:
$ javac HourGlass.java

Example Executions (user input in BOLD):

Some examples showing some error messages. Note the extra blank line after the error message before the next prompt.

Example showing the usage message:
$ java HourGlass
java HourGlass size [hourglass_char] [filler_char]
    size must be an odd integer >= 3 and <= 79
$

Use Integer.parseInt() to convert size command line arg catching appropriate exception. Command line arguments are passed to main as Strings:

$ java HourGlass 123abc
123abc is not a valid integer.

$ java HourGlass 12345678901234567890
12345678901234567890 is not a valid integer.
$

Check the size for valid range of values before checking whether it is odd:

$ java HourGlass 2
Hourglass size must be >= 3 and <= 79

$ java HourGlass 99
Hourglass size must be >= 3 and <= 79

$ java HourGlass 4
Hourglass size must be an odd integer
$

$
Example showing valid hourglass size (number of rows) using default hourglass and filler chars. Note the blank line above and below the hourglass design:

```
$ java HourGlass 11

@@@@@@@@@@@
''@@@@@@@@@
''''@@@@@@@@
''''''@@@@@@
''''''''@@@@@
'''''''''@@@ 
''''''''''@@ 
'''''''''@@@ 
''''''''''@@@@
'''''''''@@@@@
''''''''''@@@@@@

$
```

Example showing valid hourglass size specifying hourglass char using default filler char.

```
$ java HourGlass 11 '*'

***********
''*******''
''*****''
''***''
''*' 
''***''
''*****''
''*******''
''***********

$
```

Example showing valid hourglass size specifying hourglass char and filler char (space char in this example).

```
$ java HourGlass 11 '@' ' ' 

@@@@@@@@@@@
@@@@@@@@@
@@@@@@
@@@@@
@@@
@@
@@
@@@@@
@@@@@@@
@@@@@@@@@
@@@@@@@@@@@

$
```

Example showing more than one char specified for hourglass char and filler char. Check hourglass char first:

```
$ java HourGlass 11 '***' 55
Hourglass char argument must be a single character

$ java HourGlass 11 '*' 55
Filler char argument must be a single character
```

$
The Unix command line shell interprets several characters as meta characters that have a special meaning to the shell. So it is usually best to use single quotes around the hourglass and filler characters on the command line.

The hourglass and filler characters can be the same, but that output is not very interesting. :)

Also note to specify a single quote character in a single quote, use the escape sequence `\'` (see the DEFAULT_FILLER_CHAR constant below).

Here are some constants and predefined error messages you should use. Add additional constants as needed.

```java
private static final int MIN_ROWS = 3;
private static final int MAX_ROWS = 79;

private static final String USAGE_MSG = 
"java HourGlass size \{hourglass_char\} \{filler_char\}
 size must be an odd integer >= \" + MIN_ROWS + " and <= \" + MAX_ROWS + \n";

private static final String MIN_MAX_ERR_MSG = 
"Hourglass size must be >= \" + MIN_ROWS + " and <= \" + MAX_ROWS + \n";

private static final String EVEN_ERR_MSG = 
"Hourglass size must be an odd integer\n";

private static final String EXCEPTION_ERR_MSG = 
" is not a valid integer.\n";

private static final String HOURGLASS_CHAR_ERR_MSG = 
"Hourglass char argument must be a single character\n";

private static final String FILLER_CHAR_ERR_MSG = 
"Filler char argument must be a single character\n";

private static final char DEFAULT_HOURGLASS_CHAR = '@';
private static final char DEFAULT_FILLER_CHAR = '\"';
```
Program 2 – FlippingMickey:

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 left if the mouse is pressed on his left ear, flip right if the mouse is pressed on his right ear, and flip upside-down if the mouse is pressed on his face—all mouse presses must last for longer than 500 milliseconds (1/2 second) during a mouse click. Once the Mickey is flipped upside-down, if the mouse presses the face in the Mickey for longer than 500 milliseconds during a mouse click, the Mickey should flip right-side up again. Even when Mickey is upside-down, it could still be flipped either left or right by clicking on its ears. For flipping the Mickey either left or right, the flip should be centered on the ear that was clicked on. A long mouse press during a mouse click not in 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.

To summarize, this program will have all the features of PA2, plus the following features:
- If the mouse is clicked for > 0.5 seconds on the left ear -- Mickey will be flipped left
- If the mouse is clicked for > 0.5 seconds on the right ear -- Mickey will be flipped right
- If the mouse is clicked for > 0.5 seconds on the face -- Mickey will be flipped upside down (or right side up if the Mickey was already upside down)

Examples:

Display instructions at startup:

Initial mouse click:
After holding down the mouse button for more than ½ second on mouse click on the left (red) ear. Flip left centered on red ear. Left ear now green and right ear is now red:

After holding down the mouse button for more than ½ second on mouse click on the left (green) ear. Flip left centered on green ear. Left ear now red and right ear is now green:

After long mouse click on face. Flip upside down centered on blue face:

After long mouse click on left (red) ear. Flip left centered on red ear. Left ear now green and right ear red:
After long mouse click on face. Flip back right side up centered on blue face:

After long mouse click on right (red) ear. Flip right centered on red ear. Left ear now red and right ear green:

After long click on blue face. Flip upside down and drag:

Move mouse out of canvas and back in:
Notes: You should be able to drag mickey around however the figure was flipped. Also, even if part of the Mickey is off the canvas, you can still flip Mickey based on its parts that are still left on the canvas.

Drag or create Mickey close to an edge, long mouse click on right (green) ear, long mouse click on same ear. You should not have to do anything special for this.

You will need three files for this program:

```
FlippingMickey.java   Mickey.java   Timer.java
```

**FlippingMickey.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, ask the Mickey figure to check if the current mouse Location is contained in the Mickey figure, tell the Mickey figure to move or flip, tell the Mickey figure to remove itself from the canvas, etc.).

**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 as part of the FlippingMickey 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 from Figure 6.5: Timer class in the textbook for the class Timer. You can also copy it from ~/../public/Timer.java into your pa3 directory.

```
$ cp ~/../public/Timer.java ~/pa3/
```

To compile all three files, you only need to specify the source file that is the main driver that references these other types. For example,

```
$ javac -cp ./Acme.jar:./objectdraw.jar:. FlippingMickey.java
```

Since FlippingMickey.java uses the types Mickey and Timer, the compiler looks for Mickey.java/Mickey.class and Timer.java/Timer.class in the classpath specified and you have . in the classpath. The compiler will automatically compile Mickey.java and Timer.java if needed.
It is important to note the difference between the FlippingMickey class and the Mickey class. The Mickey class should have all the relevant variables and methods specific to each Mickey created on the canvas (in this case we only create one at a time). The GUI controller class, FlippingMickey, handles all the activities that occur on the canvas – this includes the mouse interactions/events with the canvas and Mickey object on the canvas.

Since both classes interact with each other, the FlippingMickey (GUI) needs to have a reference to a Mickey object and the Mickey object needs a reference to the GUI canvas in the FlippingMickey so it knows where to draw itself. This is done by passing the canvas from the GUI as an actual argument to the Mickey constructor when the code in onMouseClick() in FlippingMickey creates a Mickey object. Examples in Chapter 6 of the textbook show how to do this (sections 6.2 and 6.3 will be especially useful).

The program should still have same functionality as was specified in PA2 (able to create a Mickey on the canvas and drag it around).

Use calls to the Timer class methods in class FlippingMickey to determine how long the mouse button was pressed (remember - anything dealing with event handlers are done in the GUI controller - FlippingMickey in this case):
- Create a Timer object in begin()
- When the mouse button is pressed, the timer should be reset.
- When the mouse button is clicked, get the elapsed time in milliseconds. If the mouse click was in the Mickey figure and the elapsed time was longer than ½ second (500 milliseconds), then the Mickey figure should be flipped based on whether the long mouse click was on the left ear, right ear, or face.

A mouse click will fire these events: onMousePress(), onMouseRelease(), onMouseClick()
A mouse drag will fire these events: onMousePress(), onMouseDrag(), onMouseRelease()

A skeleton example of FlippingMickey.java:

```java
import Acme.*;
import objectdraw.*;

public class FlippingMickey extends WindowController {
    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;

    private static final String INSTR_LINE_1 = "Click to display a Mickey silhouette centered at the mouse click."
    private static final String INSTR_LINE_2 = "Mouse press in any part of the image and drag to move image around."
    private static final String INSTR_LINE_3 = "Mouse click for more than 0.5 seconds in left ear to flip left, right ear to flip right, face to flip up/down.

    private Mickey mickey;
    private Timer timer;

    // additional variables you might need and ...

    public void begin() {}
```
public void onMouseClick( Location point ) {} 

public void onMousePress( Location point ) {} 

public void onMouseDrag( Location point ) {} 

public void onMouseRelease( Location point ) {} 

public void onMouseExit( Location point ) {} 

public void onMouseEnter( Location point ) {} 

public static void main( String[] args ) { 
    new AcmeMainFrame(new FlippingMickey(), args, CANVAS_WIDTH, CANVAS_HEIGHT); 
} 

A skeleton example of Mickey.java:

```java
import objectdraw.*;
import java.awt.Color;

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 ) {} // constructor

    public boolean contains( Location point ) {} 

    public boolean inFace( Location point ) {} 

    public boolean inLeftEar( Location point ) {} 

    public boolean inRightEar( Location point ) {} 

    public void move( double dx, double dy ) {} 

    public void removeFromCanvas() {} 

    public void flipLeft() {} 

    public void flipRight() {} 

    public void flipUpDown() {}
}
```

To run your program:

```bash
$ java -cp ./Acme.jar:.objectdraw.jar:. FlippingMickey
```
EXTRA CREDIT (5 points)

Create another main GUI controller class that instead of flipping Mickey, rotates Mickey and repositions Mickey. Before you start the extra credit, make sure that FlippingMickey.java is running perfectly. From there, type the following commands:

```
$ cd ~/pa3
$ cp FlippingMickey.java EC_RotatingMickey.java
$ cp Mickey.java EC_Mickey.java
```

By using the previous Unix commands, you are creating a copy of FlippingMickey.java and naming that copy EC_RotatingMickey.java and copied Mickey.java to EC_Mickey.java. BOTH FILES ARE NEEDED IF YOU ATTEMPT THE EXTRA CREDIT.

After this, make sure to change the class definition in EC_RotatingMickey.java to the new name. ie:

```java
public class EC_RotatingMickey extends WindowController {
    new Acme.MainFrame(new EC_RotatingMickey(), args, CANVAS_WIDTH, CANVAS_HEIGHT);
}
```

and change all instances of Mickey to EC_Mickey

```java
private EC_Mickey mickey;
    mickey = new EC_Mickey(point, canvas);
```

and likewise in EC_Mickey.java:

```java
public class EC_Mickey {
    public EC_Mickey(Location point, DrawingCanvas canvas) { // constructor
    }
}
```

The following features must be implemented:

1) If the Mickey is long mouse clicked on his left ear, Mickey will rotate counter-clockwise by 90 degrees around the center of the face.

2) If the Mickey is long mouse clicked on his right ear, Mickey will rotate clockwise by 90 degrees around the center of the face.

3) If the Mickey is long mouse clicked on his face, Mickey will be repositioned to the middle of the canvas based on the current dimensions of the canvas facing upright (original orientation).

Just like for the FlippingMickey.java implementation, these mouse clicks must last longer than than ½ second (500 milliseconds) during the mouse click.

To compile:

```
$ javac -cp ./Acme.jar:/objectdraw.jar:. EC_RotatingMickey.java
```

To run:

```
$ java -cp ./Acme.jar:/objectdraw.jar:. EC_RotatingMickey
```
Example runs:

Start program with instructions

Initial click

After long click on left (red) ear (rotate left)

After long click on left (red) ear (rotate left)
After long click on face
(center image on canvas in upright orientation)

After long click on right (green ear)
(rotate right)

Dragging the Mickey figure around keeps its current orientation

Just to be explicit, no matter what orientation the Mickey is currently in, clicking in the face for over a half second will center the figure upright in the center of the canvas.
Note: You will NOT receive extra credit for repositioning Mickey in the middle if you hardcode Mickey to be moved to the center of a window with a specific size; we will be checking this particular feature with arbitrary window sizes.

This time, the starting text/instructions of your program should be adjusted to look like the following (and should disappear when a Mickey is placed). First two lines are the same as in FlippingMickey:

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 for more than 0.5 seconds in left ear to rotate left, right ear to rotate right.
Mouse click for more than 0.5 seconds in face to center image on canvas upright.

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

$ cse11turnin 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,

$ cse11verify 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:

- HourGlass.java
- Mickey.java
- FlippingMickey.java
- Timer.java
- objectdraw.jar
- Acme.jar
- README

Additional files to be collected for Extra Credit:

- EC_RotatingMickey.java
- EC_Mickey.java

The files that you turn in (in your pa3 dir) must be EXACTLY the same name as those above, and no other .java source files. Any .java source file in your pa3 must compile without error for turnin.

**NO LATE ASSIGNMENTS ACCEPTED.**

**DO NOT EMAIL US YOUR ASSIGNMENT!**

**Start Early and Often!**