Programming Assignment 3 (100 Points)

Due: 11:59pm Thursday, October 13
<u>START EARLY!</u>

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

<u>START EARLY!</u>

<u>README (10 points)</u>
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**" (1 point). Your README should include the following sections:

Program Description (2 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:
<u>Vim related Questions:</u>
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)

<u>Unix/Linux related Questions:</u>
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?

<u>STYLE (20 points)</u>
Please see PA2 for details on Coding Style.

CORRECTNESS (70 points)

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 - Parallelograms (35 points) :

Write a Java application (**Parallelograms.java**) that displays parallelogram patterns side-by-side using nested loops. The size of the displayed patterns is user-defined (read from console). Everything can be in `main()`.

Input:

- Use the class **java.util.Scanner** to read user input.
- Allow the user to input the number of rows defining the size of the side-by-side triangles that make up the two parallelograms. **No hardcoding or magic numbers in your code.**
- Only valid integers will be entered, but you must check for invalid sizes (integers < 2 and > 20), and report the **exact error message** as shown in the example below.
- Your program should exit successfully upon entering **Ctrl-D** (see example 3 in the sample output).

Output:

- Print out the patterns one row at a time. That is, the outermost loop should iterate number-of-rows times. Each of the four triangle patterns should have its own inner loop for each row.
- HINT: Each row of each triangle pattern is made up of some number of individual `'*'` and `' '` (space) chars.
- 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 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.

Format:

- Formatting is important for this program. **If your output does not match the required output character-for-character, NO credit will be awarded.**
- Make sure your prompt and output messages match the sample prompt and messages **word-for-word**.
- NOTE: there is a space character between adjacent triangles, there is no space at the end of each line, and there is no new line below the parallelograms.

To get a better understanding, if the size of the parallelogram is 5, the spacing should look like the following (represented in a grid, for a clearer picture of the individual `'*'` and `' '` characters. The stars have been emphasized by shading the box grey--this is just for illustration purposes)

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

Sample Output (user input in **BOLD**) (we will use more/other test cases for grading):

Example 1:
```
[cs11fxyz@ieng6-203]:pa3$ java Parallelograms
Enter the size of the parallelograms to display: 12

************ *                       * ************
 *********** **                     ** ***********
  ********** ***                   *** **********
   ********* ****                 **** *********
    ******** *****               ***** ********
     ******* ******             ****** *******
      ****** *******           ******* ******
       ***** ********         ******** *****
        **** *********       ********* ****
         *** **********     ********** ***
          ** ***********   *********** **
           * ************ ************ *
[cs11fxyz@ieng6-203]:pa3$
```

Example 2:
```
[cs11fxyz@ieng6-203]:pa3$ java Parallelograms
Enter the size of the parallelograms to display: -1
Parallelogram size must be >= 2 and <= 20; Try again.

Enter the size of the parallelograms to display: 0
Parallelogram size must be >= 2 and <= 20; Try again.

Enter the size of the parallelograms to display: 5

***** *         * *****
 **** **       ** ****
  *** ***     *** ***
   ** ****   **** **
    * ***** ***** *
[cs11fxyz@ieng6-203]:pa3$
```

Example 3:
```
[cs11fxyz@ieng6-203]:pa3$ java Parallelograms
Enter the size of the parallelograms to display: ^D
[cs11fxyz@ieng6-203]:pa3$
```

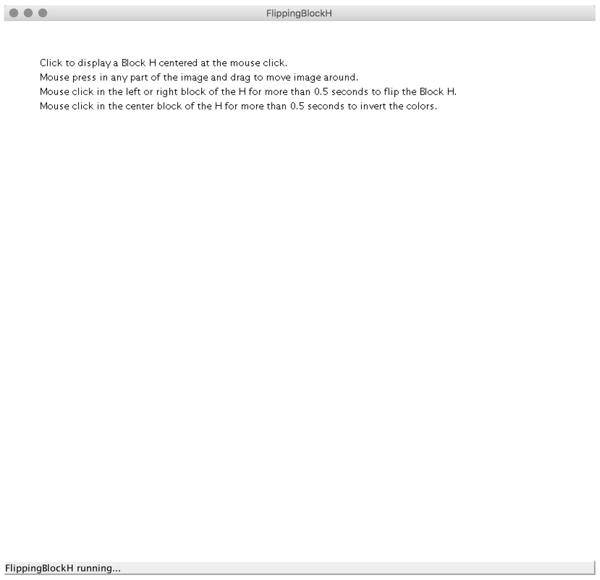
Program 2 – BlockH (35 points) :

This program will build off of your last BlockH program, so you should make sure that your PA2 is working correctly first. In order to make this program more Object-Oriented, we will separate the current functionality inside DraggingBlockH into two different classes: the GUI controller and an improved BlockH object.

This program will have all the features of PA2, plus the following features:

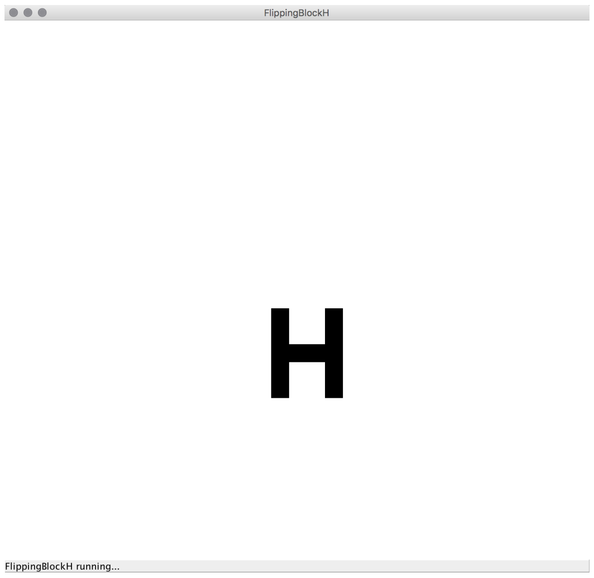
- If the mouse is clicked for > 0.5 seconds **on the left bar** — BlockH will be **flipped left**.
- If the mouse is clicked for > 0.5 seconds **on the right bar** — BlockH will be **flipped right**.
- If the mouse is clicked for > 0.5 seconds **on the center bar** — the colors of BlockH and the canvas become inverted (meaning the black BlockH becomes white, and the white canvas becomes black).

Sample Output:

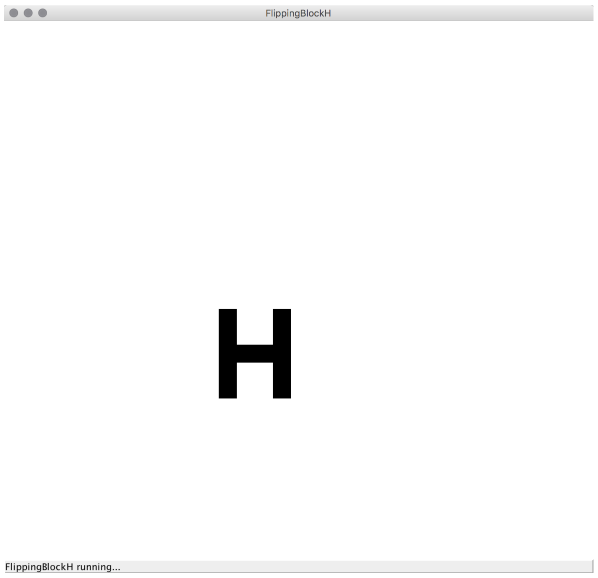


```
Click to display a Block H centered at the mouse click.
Mouse press in any part of the image and drag to move image around.
Mouse click in the left or right block of the H for more than 0.5 seconds to flip the Block H.
Mouse click in the center block of the H for more than 0.5 seconds to invert the colors.
```

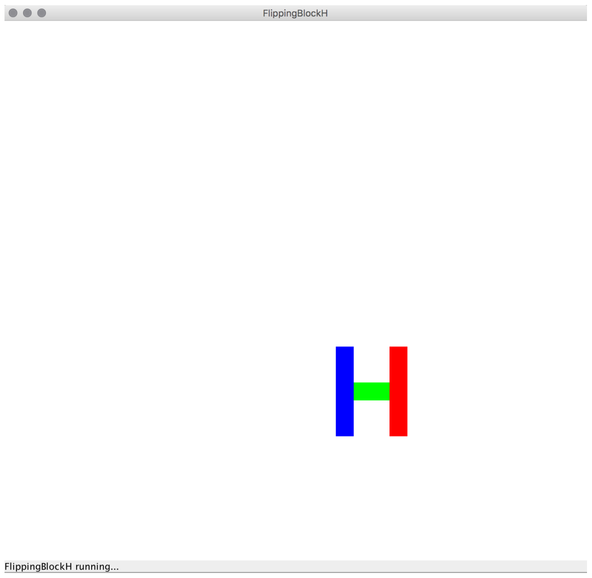
Display instructions at start up.



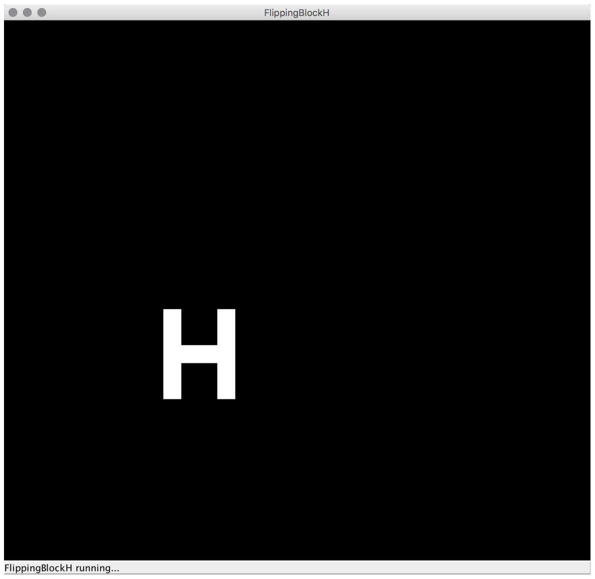
On mouse click, instructions are hidden and BlockH is drawn centered at the point of the mouse click.



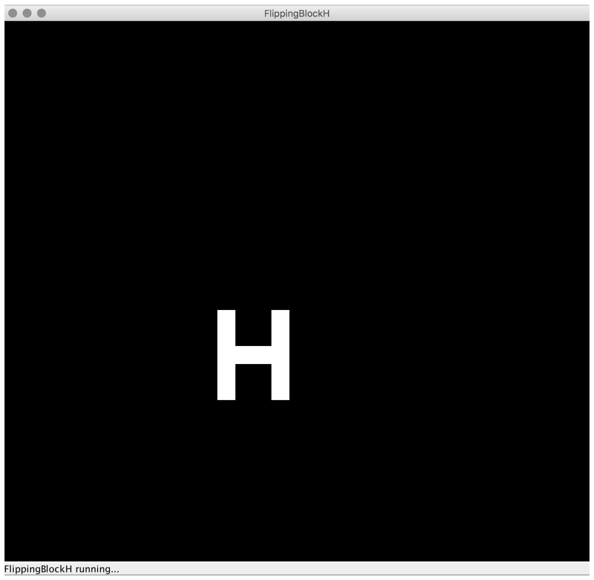
Then after clicking the mouse for more than 0.5s on the left bar of BlockH, BlockH is flipped left.



Dragging BlockH after the previous flip (note how the red and blue are swapped because of the flip).

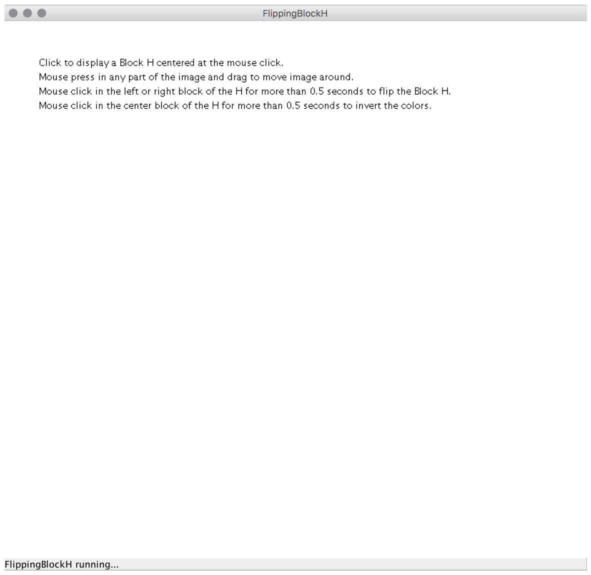


After a >0.5s mouse click on the center bar of the BlockH, the colors of BlockH and the canvas are inverted.



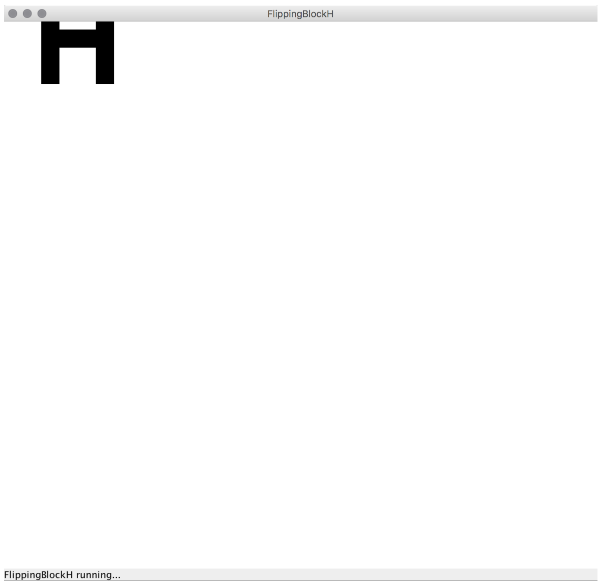
After a >0.5s mouse click on the right bar of BlockH, H is flipped right. And the inverted colors remain unchanged.

Once the mouse exits the canvas, the BlockH is removed and everything is reset to the original state.

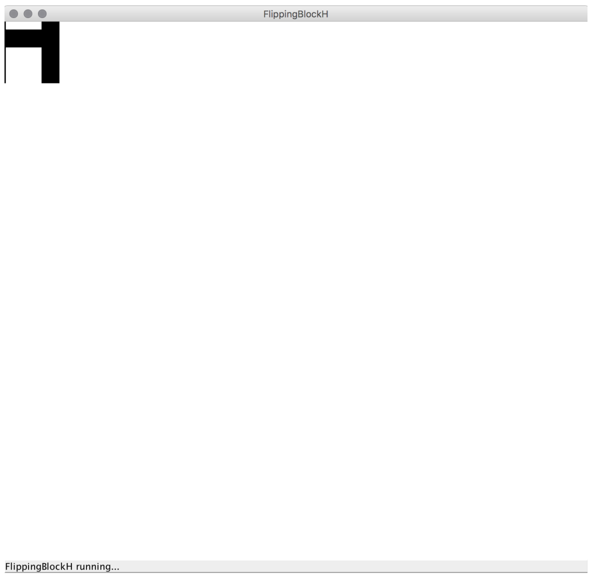


Click to display a Block H centered at the mouse click.
Mouse press in any part of the image and drag to move image around.
Mouse click in the left or right block of the H for more than 0.5 seconds to flip the Block H.
Mouse click in the center block of the H for more than 0.5 seconds to invert the colors.

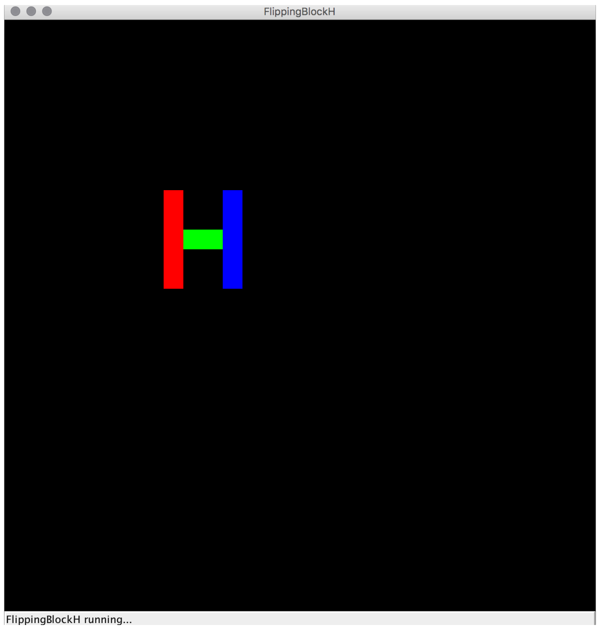
When the mouse re-enters the canvas, the instructions reappear.



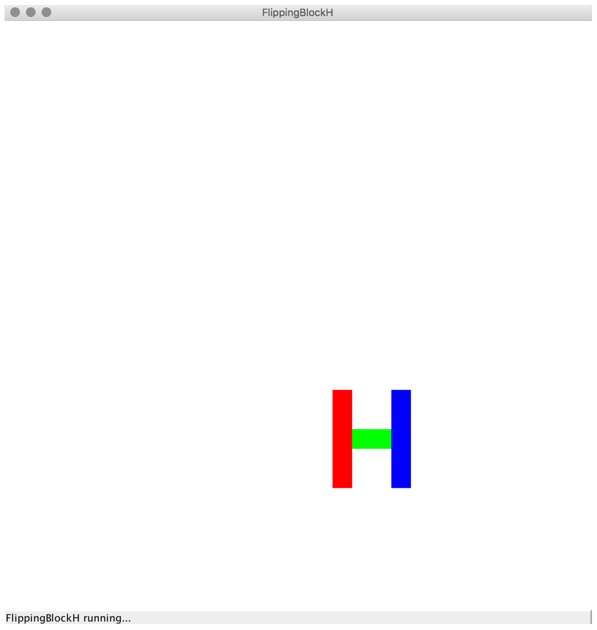
New BlockH is made where part of the BlockH is off the canvas.



Then, after holding on the left bar, you can still flip BlockH based on its parts that are still on the canvas.



Dragging BlockH while colors are inverted.



Dragging BlockH while colors are not inverted.

You will need three files for this program:

`Timer.java` `FlippingBlockH.java` `BlockH.java`

Timer.java

This class calculates timing between events. The code is provided below, as well as on page 168 of the textbook.

FlippingBlockH.java
This is the main GUI controller class that handles all mouse events and controls what we see on the canvas. FlippingBlockH will create an instance of a BlockH and send messages to the BlockH, which includes telling the BlockH to flip, move, remove itself from the canvas, invert its color (meaning black to white or white to black), and asking BlockH to check if the current mouse location is within the BlockH figure. Feel free to use helper methods to encapsulate certain tasks, like inverting the colors.

Note: Use the following line of code to set the color of the canvas (you'll learn more about this later).

```
    ((JDrawingCanvas)canvas).setBackground( Color.BLACK );
```

BlockH.java
This class defines what a BlockH figure is (the 3 filled rectangles) and what a BlockH figure can do (what messages it can respond to). This class will need:

- A constructor that initializes a new BlockH object, which will be added to the canvas as part of the FlippingBlockH
- Methods to determine if the mouse pointer is contained in the BlockH figure, and where it is on the BlockH
- Methods that define different behaviors of the BlockH, such as to move the BlockH figure some distance from its current location, remove the BlockH figure from the canvas, flip the BlockH figure left and right, and any other methods you may need for setting the colors.

Specifications:

It is important to note the differences between the FlippingBlockH class and the BlockH class. The BlockH class should have all the relevant variables and methods that define the behavior of a BlockH object. FlippingBlockH, which is the GUI controller class, handles all the activities that occur on the canvas – this includes the mouse interactions/events with the canvas and creating a BlockH object on the canvas.

In order to create a BlockH object, it will need to know what canvas to draw itself on. This is done by passing the canvas in as an argument when we make a call to the BlockH constructor. This allows the rectangles to be drawn on the canvas inside the BlockH constructor. **Hint:** See examples in Chapter 6 of the textbook (sections 6.2 and 6.3 will be especially useful).

Use calls to the **Timer** class methods in FlippingBlockH to determine how long the mouse button was pressed:

- Create a Timer object in begin()
- When the mouse 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 BlockH figure and the elapsed time was longer than ½ second (500 milliseconds), then flag that the BlockH figure should be flipped.

Details:

Functionality

- The program should still have the same functionality as was specified in PA2 (able to create a BlockH on the canvas and drag it around, changing the colors when the mouse is pressed on the BlockH).

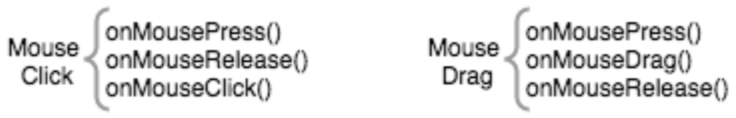
Starting Text

- The starting text/instructions of your program should be adjusted to look like the following (and should disappear when a BlockH is placed, just like in your last assignment):

```
Click to display a Block H centered at the mouse click.
Mouse press in any part of the image and drag to move image around.
Mouse click in the left or right block of the H for more than 0.5 seconds to flip the Block H.
Mouse click in the center block of the H for more than 0.5 seconds to invert the colors.
```

Press, Release, Drag, Click

- If the mouse is pressed outside of the BlockH, nothing should happen.
- When the mouse is *pressed* on BlockH, the H should still turn RED, GREEN, and BLUE as in PA2.
- You should be able to *drag* BlockH around despite the orientation (flipped or not).
- The mouse *click* event is defined to be a mouse *press* event following a mouse *release* event with no mouse movements (no dragging) between the press and release.
- The length of a clicking time is measured by the elapsed time from the point the BlockH was *pressed* to the point the mouse was *released*. In this case, a long click means the elapsed time is at least 0.5 seconds (500 milliseconds). Use the following constant to compare with the elapsed time the BlockH was pressed.
  ```
  private static final int FLIP_PRESS_THRESHOLD = 500; // Half a second
  ```
- The following diagram shows which methods are called during which events.



Flipping/Inverting Colors

- BlockH should **NOT** flip/invert colors when
  - BlockH is being dragged
  - Click is not on the BlockH.
  - Click is not long enough (not held for at least 0.5 seconds)
- When flipping BlockH, **DO NOT** just alter the colors of the left and right bar, and translate the entire BlockH. You **must** flip by repositioning the necessary rectangles without changing their original color. For instance, after flipping right, `rightBar` will end up on the left side of the canvas, but it should still be referred to as `rightBar` and it should remain blue.
- When flipping the BlockH, the flip should be centered on the bar that was clicked.
- If part of the BlockH is off the canvas, you can still flip the BlockH based on its parts that are still left on the canvas
- On mouse exit, everything is reset to the initial state, which includes the canvas reverting back to white.

<u>**Skeleton Code Examples**</u>

FlippingBlockH.java:

```
import objectdraw.*;

public class FlippingBlockH extends WindowController {
  private Text instr1, instr2, instr3, instr4;
  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 INSTR4_X = INSTR1_X;
  private static final int INSTR4_Y = INSTR3_Y + 20;
```

```
  private static final int CANVAS_WIDTH = 750;
  private static final int CANVAS_HEIGHT = 750;

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

  // additional variables you might need

  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 Acme.MainFrame(new FlippingBlockH(), args, CANVAS_WIDTH, CANVAS_HEIGHT);
  }
}
```

BlockH.java:

```
import objectdraw.*;

public class BlockH {
  private FilledRect leftBar, centerBar, rightBar;

  // Dimensions of center bar
  private static final double CENTER_BAR_WIDTH = 50;
  private static final double CENTER_BAR_HEIGHT = 25;

  // Dimensions of side bars
  private static final double SIDE_BARS_WIDTH = 25;
  private static final double SIDE_BARS_HEIGHT = 125;

  …

  // additional variables you might need and …

  public BlockH( Location point, DrawingCanvas canvas ) {} // ctor

  public boolean contains( Location point ) {}

  public boolean inLeftBar( Location point ) {}

  public boolean inCenterBar( Location point ) {}
  public boolean inRightBar( Location point ) {}

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

  public void removeFromCanvas() {}

  public void flipLeft() {}

  public void flipRight() {}

  public void showColors( boolean on ) {}

  // additional methods you might need
}
```

Timer.java:

```
public class Timer {
  private double startTime; //Time when Timer started or reset

  //Create timer, initializing startTime with current time
  public Timer(){
    startTime = System.currentTimeMillis();
  }

  //Return number of milliseconds since last reset
  public double elapsedMilliseconds(){
    return System.currentTimeMillis() - startTime;
  }

  //Reset startTime
  public void reset(){
    startTime = System.currentTimeMillis();
  }
}
```

EXTRA CREDIT (5 points)

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

```
$ cd ~/pa3
$ cp FlippingBlockH.java EC_RotatingBlockH.java
```

By using the Unix commands above, you are creating a copy of FlippingBlockH.java and naming that copy EC_RotatingBlockH.java. **BOTH FILES ARE NEEDED IF YOU ATTEMPT THE EXTRA CREDIT.**

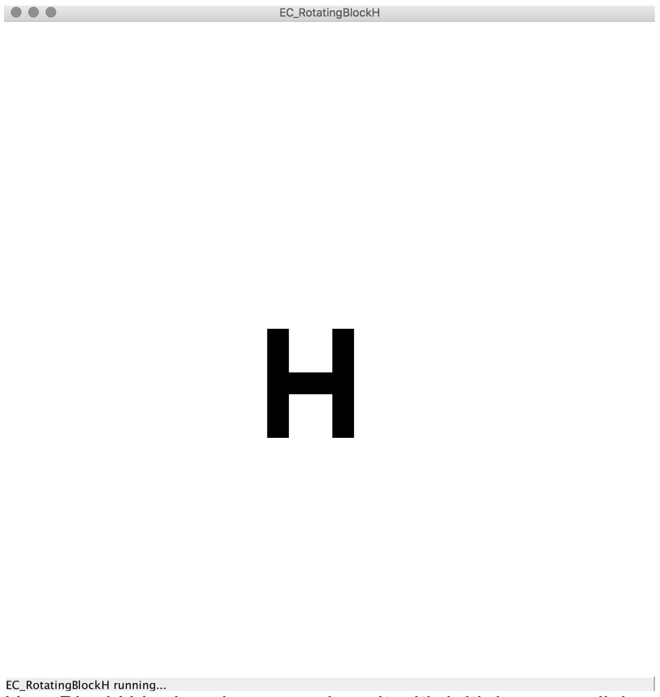
Make sure to change the class definition in EC_RotatingBlockH.java to the new name:

```
public class EC_RotatingBlockH extends WindowController {
   …
}
```

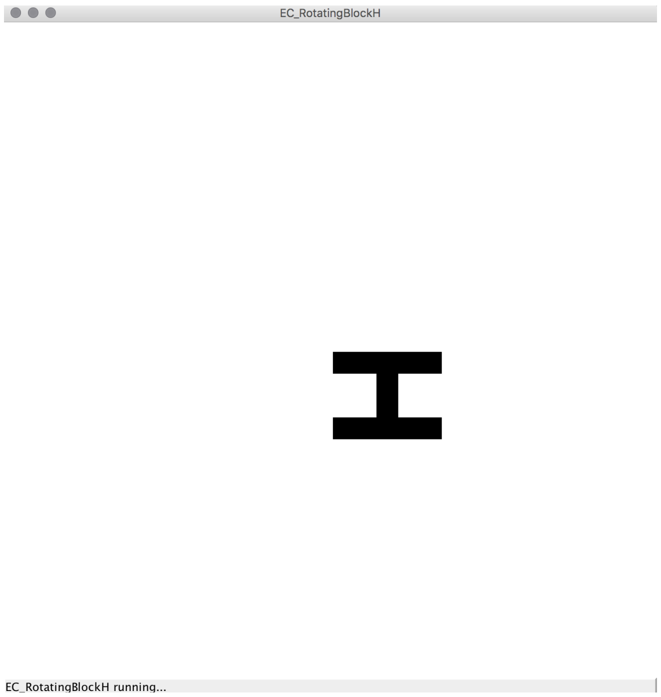
The following features must be implemented for extra credit:

- Click for > 0.5 seconds on the the **left bar**, BlockH will rotate **counter-clockwise** by 90 degrees around the bottom corner of the left bar.
- Click for > 0.5 seconds on the **right bar**, BlockH will rotate **clockwise** by 90 degrees around the bottom corner of the right bar.
- Click for > 0.5 seconds on the **center bar**, BlockH will be **repositioned upright in the middle** of the window pane. BlockH must be centered based on the middle point of the center bar.

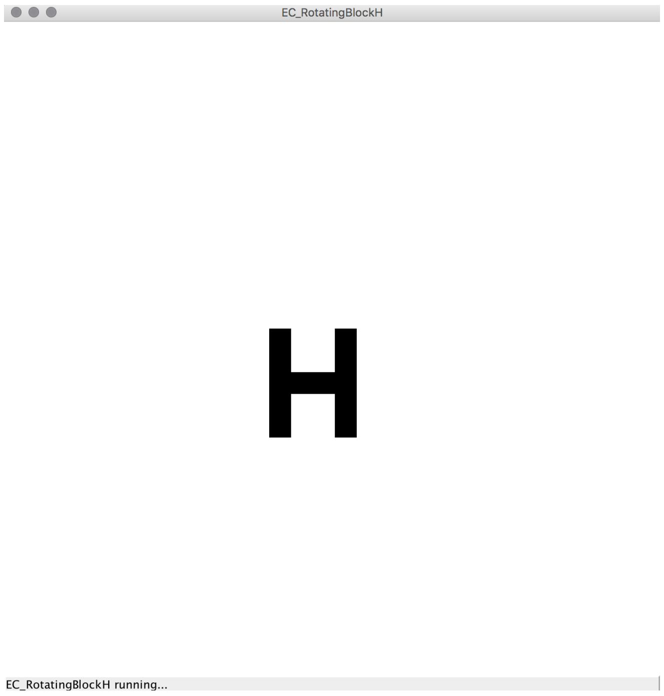
Sample Output:



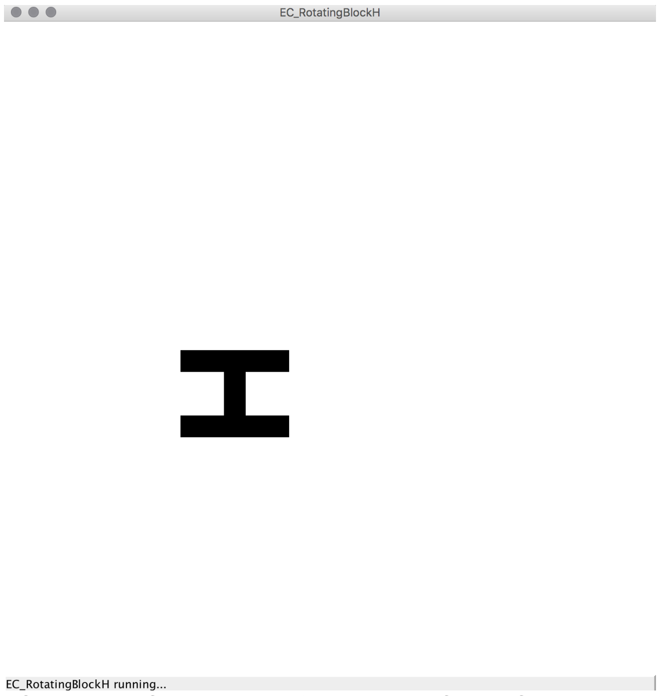
How BlockH looks when you place it with initial mouse click.



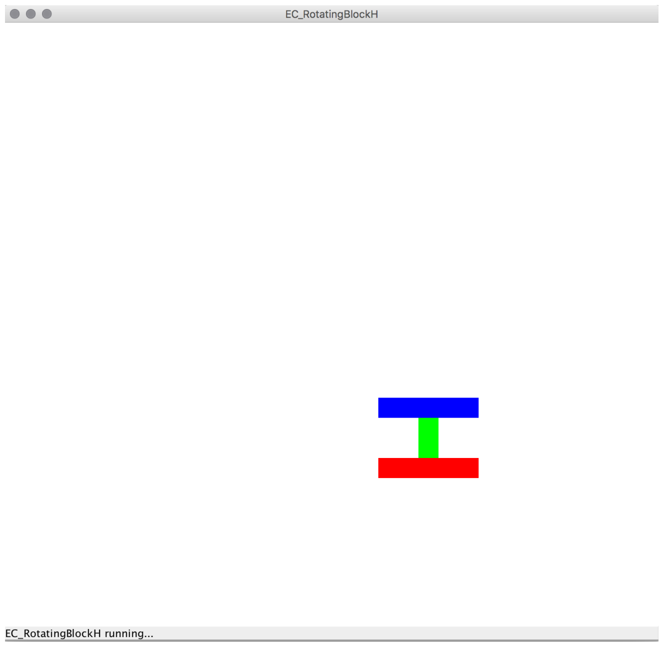
After clicking for more than 0.5s on the right bar of BlockH, it is rotated right.



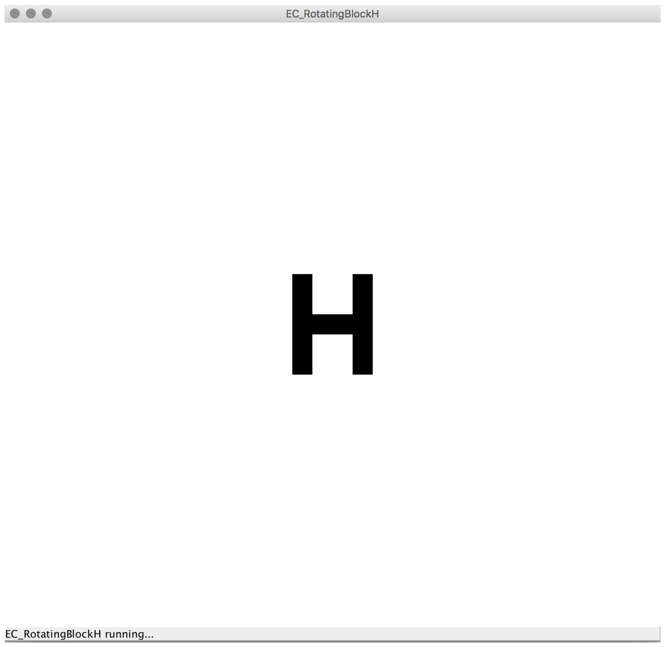
Again, how BlockH looks when you place it with initial mouse click.



After clicking for more than 0.5s on the left bar of BlockH, it is rotated left.



Dragging BlockH after the last rotate left.



After clicking for more than 0.5s on the center bar of BlockH, it is centered upright in the middle of the window.

Just to be explicit, no matter what orientation the BlockH is currently in, clicking on the center bar for over half a second will center the figure upright in the center of the canvas. (The colors of the BlockH and canvas will no longer be inverted when clicking on the center bar.)

Note: You will NOT receive extra credit for repositioning BlockH in the middle if you hardcoded BlockH 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 BlockH is placed):

```
Click to display a Block H centered at the mouse click.
Mouse press in any part of the image and drag to move image around.
Mouse click in the left or right block of the H for more than 0.5 seconds to rotate the Block H.
Mouse click in the center block of the H for more than 0.5 seconds to center the Block H upright.
```

Turnin

Always recompile and run your program right before turning it in, just in case you commented out some code by mistake.

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. Verify will help you check which files you have successfully submitted. **We will not take any late files you forgot to turn in.** It is your responsibility to make sure you properly turned in your assignment.

Files to be collected:

- Parallelograms.java
- BlockH.java
- FlippingBlockH.java
- Timer.java
- objectdraw.jar
- Acme.jar
- README

Additional files to be collected for Extra Credit:

- EC_RotatingBlockH.java

The files that you turn in must be EXACTLY the same name as those above.

NO LATE ASSIGNMENTS ACCEPTED.

DO NOT EMAIL US YOUR ASSIGNMENT!

<u>Start Early and Often!</u>