Programming Assignment 2 (100 Points)

Due: Thursday, October 16 by 11:59pm

This assignment has two programs: one a Java application that reads user input from the command line (TwoLargest) and one a Java applet similar to several examples in the textbook (Mickey).

README (10 points)

You are required to provide a text file named **README,** NOT Readme.txt, README.pdf, or README.doc, with your assignment in your pa2 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 to work your program. What sort of inputs or events does it take and what are the expected outputs or results? If two or more integer values were entered a number of times or less than two values were entered, how did you determine what to display as the largest and/or second largest value? What did you do to ensure only one Mickey silhouette was displayed with multiple mouse clicks, how did you prevent any exceptions being thrown in the console/terminal window when you move the mouse out and in and out and in focus in the canvas?

Write your READMEs as if it was intended for someone who is computer illiterate. 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:

1. What is the Linux command to rename a file? What is the command to copy a file?
2. What happens when you select text and then middle click in vim when in insert mode?
3. What is the name of the application that processes commands on the command line?
4. What are three ways to enter insert mode in vim?
5. How do you switch from insert mode to command mode in vim?
6. What happens when you type ":split" in command mode in vim?
7. What is a method?

STYLE (20 points)

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.

- Use reasonable comments to make your code clear and readable.
- Use class headers and method header blocks to describe the purpose of your program and methods (see below).
- Use reasonable variable names.
- Use static final variables to make your code as general as possible.
- 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 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.
- Do not use magic numbers or hard-coded numbers other than 0 and 1.

You will be specifically be graded on commenting, file headers, class and method headers, meaningful variable names, sufficient use of blank lines, not using more than 80 characters on a line, perfect indentation, and no magic numbers/hard-coded numbers other than zero.

Example file header comment:

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

Example class and method header comment:

```
/*
 * Name: Class or method name
 * Purpose: Briefly describe the purpose of this class or 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.
 */
```

CORRECTNESS (70 points, 35 points each program)

Setting up your pa2 directory:

You will need to create a new directory named **pa2** in your cs11f home directory on ieng6.ucsd.edu. The '>' character represents the command line prompt. Type the commands that appear after the command line prompt.

```
> cd
> mkdir pa2
```

The first command (cd) changes your current directory to your home directory. cd stands for change directory. By default, if you do not specify a directory to change to the command will put you in your home directory.

The second command (mkdir pa2) makes a new directory named pa2. This new directory will be in your home directory since you did a cd beforehand.

Now type

```
> cd pa2
```

This will change your current working directory to the new pa2 directory you just created. All files associated with this programming assignment must be place in this directory. And in general, you should do all your work on this programming assignment in this pa2 directory.

Now copy over a required jar file from the public directory (note the **.** as the last argument)

```
> cp ~/../public/objectdraw.jar .
```

To change your current working directory back to your home directory, you can type **cd** as you did before.

Program 1 - TwoLargest.java

Write a Java application with a main() to find the two largest distinct integers entered on the command line by the user. All of your code can be in main(). See PA1 for an example of the definition of main().

Use a **Scanner** object to read user input from standard input (**System.in**).
The Scanner Javadocs: http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

The basic framework to use a Scanner is:

```
// Construct a Scanner that produces values scanned from standard input
Scanner input = new Scanner(System.in);

while (input.hasNext()) {          // while there is more input (user has not hit EOF)
    number = input.nextInt();      // read next integer


    …
}
```

Make sure your prompt and output messages exactly match the sample prompts and messages word-for-word. See below for examples.

Any valid integer may be entered. We will only enter valid integers.

Numbers entered may be repeated.

There is no need to handle invalid input (for example, non-numeric input). We will not be testing for it. We will only be testing for valid positive and negative integers.

Use the **EOF** (end-of-file) keyboard sequence [<Ctrl>+D on Unix/Mac or <Ctrl>+Z on Windows] as the first character sequence on a line and then <Enter> to indicate no more input. (^D represents the user entering <Ctrl>+D).

Handle any number of entries (special case of less than 2 numbers entered).

You **cannot** use any data structure (like an array, ArrayList, Vector, List, etc.) to hold the input and then sort them. Just use two variables to keep track of the largest and second largest distinct values entered and any other variables you may need.

Distinct largest and second largest values means the largest and second largest values cannot be the same value. For example, in the series of values 1, 2, 1, 1 there are only two distinct values (1 and 2). In the series of values 1, 1, 1, 1 there is only one distinct value (1).

Only valid integers will be tested. You do not need to do any error checking for invalid input.

Compiling

To compile your code, use the following:

> **javac TwoLargest.java**

Running

To execute your application, use the following:

> **java TwoLargest**

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

Example 1:

```
java TwoLargest
Enter a series of integers; EOF to Quit.
99 -5 99
^D
Largest distinct number entered was 99
Second largest distinct number entered was -5
```

Example 2:

```
java TwoLargest
Enter a series of integers; EOF to Quit.
3
^D
Largest distinct number entered was 3
```

Example 3:

```
java TwoLargest
Enter a series of integers; EOF to Quit.
8 0
5 -19
55
13
-2
^D
Largest distinct number entered was 55
Second largest distinct number entered was 13
```

Example 4:

```
java TwoLargest
Enter a series of integers; EOF to Quit.
44 44
^D
Largest distinct number entered was 44
```

Example 5:

```
java TwoLargest
Enter a series of integers; EOF to Quit.
^D
No numbers entered.
```

Example 6:

```
java TwoLargest
Enter a series of integers; EOF to Quit.
-2147483648
^D
Largest distinct number entered was -2147483648
```

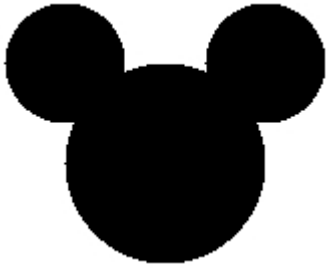
Program 2 - Mickey.java

Write a Java applet that does the following (use the textbook examples as guides):

When the applet first starts up, display instructions on two lines using Text objects. See the screenshots below for exact wording of instructions.

> 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.

On a mouse click, hide the instructions, and if there is no Mickey silhouette already drawn then draw a Mickey silhouette with the face of the Mickey centered at the point of the mouse click. The silhouette is made up of 3 FilledOvals - a face, left ear, and right ear. See the screenshots and constants defined below.



Only one Mickey figure should be drawn no matter how many mouse clicks occur.

On a mouse press, if the Mickey silhouette has been drawn, check if the point of the mouse press is anywhere in the silhouette (in other words, you are grabbing the Mickey figure). You can grab either ear or face. Of course, on mouse release you are no longer grabbing it.

On a mouse drag, if the Mickey figure has been grabbed (mouse press is inside the figure's boundaries), drag the Mickey figure around with the mouse drag. You will need to move all 3 parts of the figure. See the examples in the textbook. The figure should continue to look like a Mickey silhouette during and after the drag.

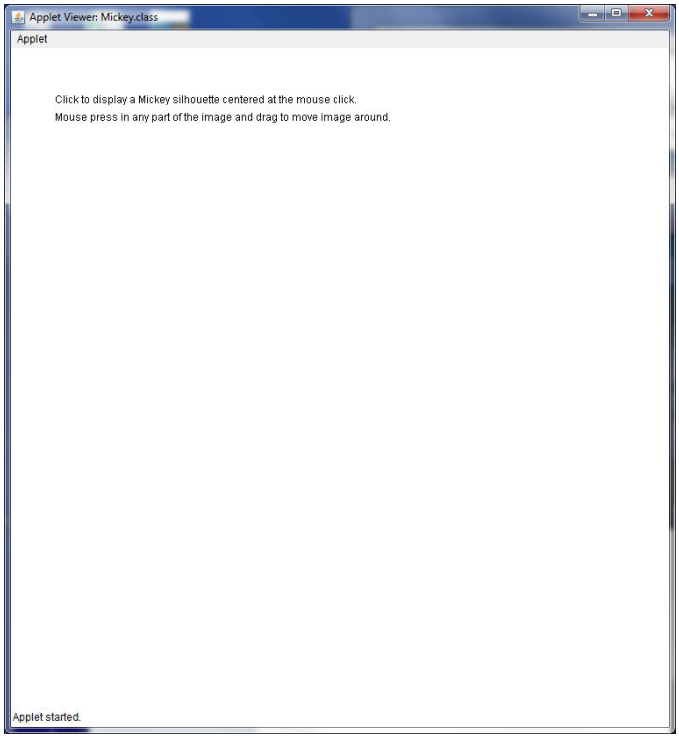
On a mouse exit (when the mouse pointer exits the canvas), if the Mickey figure had been drawn, remove the entire figure from the canvas and reset the logic you are using to keep track of whether a Mickey figure has been drawn or not (you can do this in many ways).

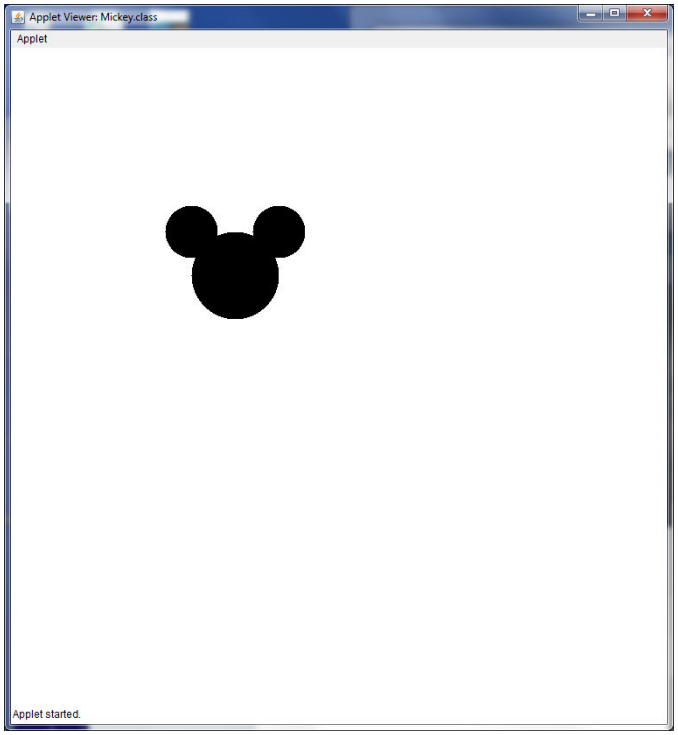
On a mouse enter (when the mouse pointer (re)enters the canvas), show the instructions.

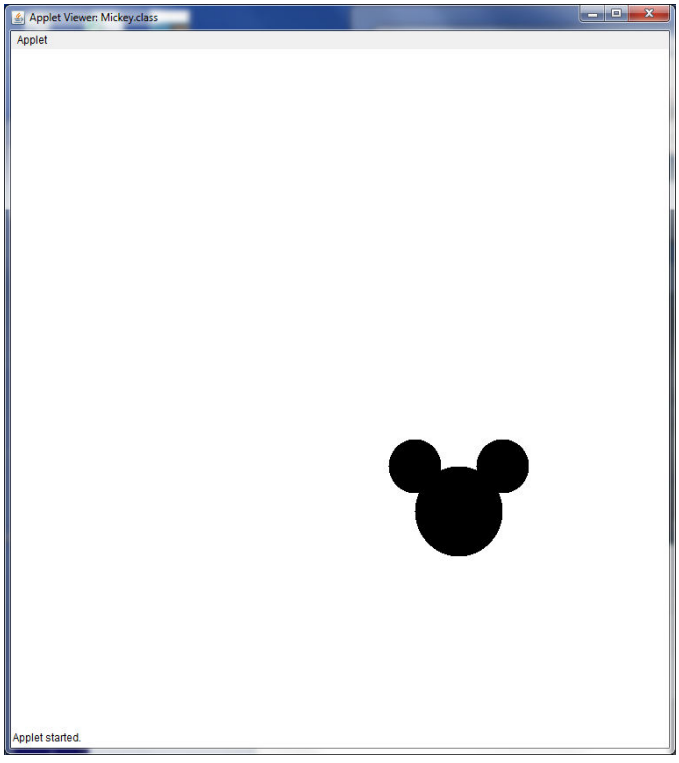
Many of the checks in the event handlers above are to prevent blindly trying to remove non-existing objects from the canvas, displaying more than one figure with multiple mouse clicks, checking if you are trying to grab a non-existing figure, etc. Without these checks you will likely get exceptions thrown in the terminal window were you ran appletviewer. **No exceptions should be thrown!**

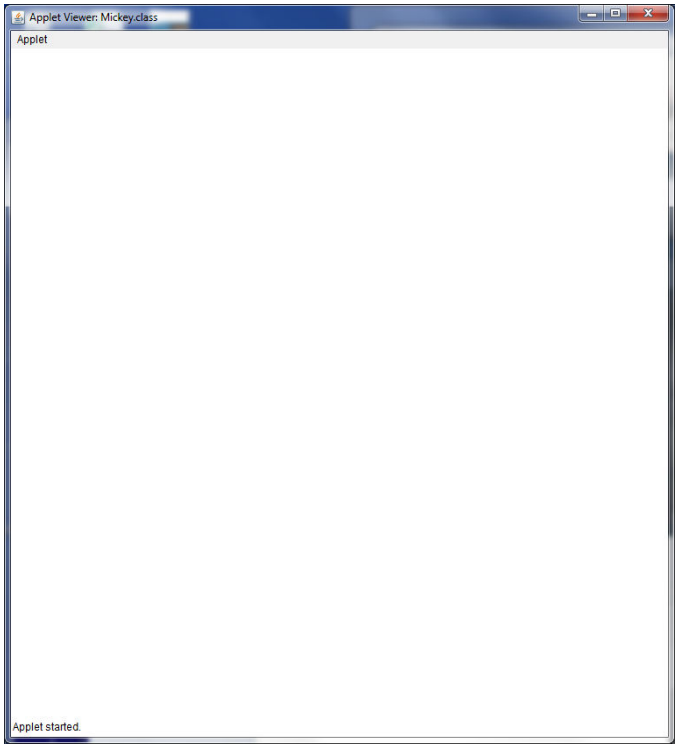
To get you started here are some constants you should use:

```
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 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.
```

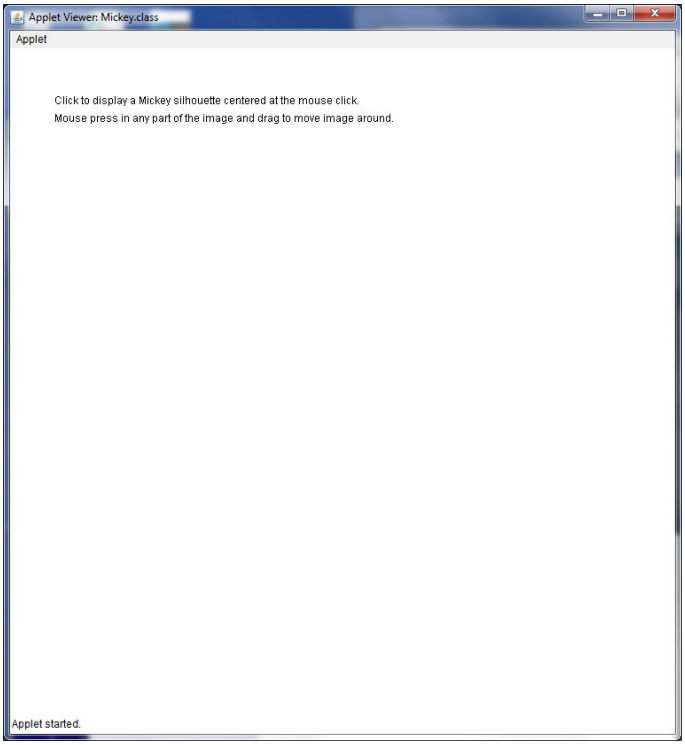


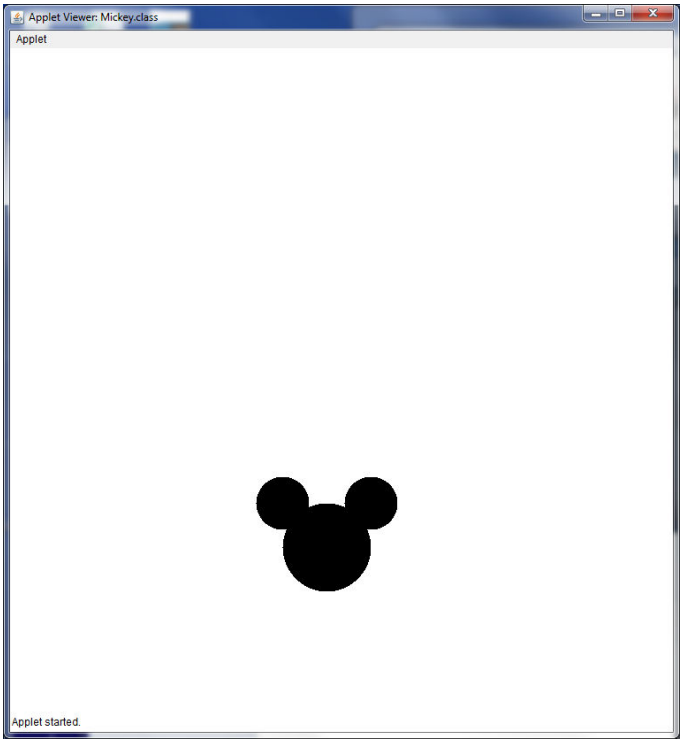






At start-up, on mouse click, dragging the figure around, on mouse exit.





On mouse (re)enter, on mouse click.

Compiling:

To compile your code, use the following:

```
> javac -cp ./objectdraw.jar:. Mickey.java
```

Running

To run (and view) your applet, first create an html file named **Mickey.html**, which contains the following code:

```
<html>
   <body>
     <applet
       code="Mickey.class"
       archive="objectdraw.jar"
       width="750"
       height="750">
     </applet>
   </body>
</html>
```

Then use the appletviewer command specifying this html file:

```
> appletviewer Mickey.html
```

You must have all the files in your pa2 directory and run appletviewer in your pa2 directory for it to display correctly.

Extra Credit (5 points)

Write a Java application with a main() to find the **three** (3) largest distinct integers entered on the command line by the user. Before you start the extra credit, make sure that TwoLargest.java is running perfectly. From there, type the following commands:

```
> cd ~/pa2
```

```
> cp TwoLargest.java ThreeLargest.java
```

By using the previous unix commands, you are creating a copy of TwoLargest.java and naming that copy ThreeLargest.java. BOTH FILES ARE NEEDED IF YOU ATTEMPT THE EXTRA CREDIT

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

```
public class ThreeLargest {
      …
}
```

When implementing the changes to ThreeLargest.java, the same restrictions apply that were included in TwoLargest.java.

That means:

Any valid integer can be used as input.

The same integer can be entered multiple times. The output must contain DISTINCT numbers.

NO DATA STRUCTURES (ie no LinkedLists, ArrayLists, HashTables, etc). You will only need an additional variable to keep track of the third largest number.

Everything else mentioned in the TwoLargest section

Turnin

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

> **`turnin pa2`**

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 pa2`**

If you are unsure if 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:

- Mickey.html
- Mickey.java
- TwoLargest.java
- objectdraw.jar
- README

Additional files to be collected for Extra Credit:

- ThreeLargest.java

NO LATE ASSIGNMENTS ACCEPTED.

DO NOT EMAIL US YOUR ASSIGNMENT!

Start Early and Start Often!