Programming Assignment 6 (100 Points)
Due: 11:59pm Thursday, November 3

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 pa6 directory. There should be no file extension after the file name "**README**". Your README should include the following sections:

Program Description (3 points) : Provide a high level description of what your program does and how you can interact with it. Make this explanation such that your grandmother or uncle or someone you know who has no programming experience can understand what this program does and how to use it. For example, if you intentionally add padding to a line so it is easier to grab, say it here so the grader knows it was intentional. Write your README as if it were intended for a 5 year old. 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 Questions

1. Using vim/gvim, how can you open a file from the Linux command line so that you are taken directly to a specified line in the file?
2. How do you turn off special coloring of keywords and syntax in vim (this is most evident in gvim)? For example, by default your gvim colors comments differently from other parts of code (comments are colored blue by default). If you turn off syntax coloring then all the code in your file will be of same color, black by default. How do you turn the coloring back on?
3. What is the command that you type in to view more information on how to use vim/gvim? The specific command that we're looking for is the one you type in command mode in an open vim/gvim file, not on the terminal. Typing the command will bring up a help manual for vim.
4. How do you jump to a specific subject in the help manual for vim? (This is easily answered using the answer to question 3).

Java Questions

5. What structure in the run time stack holds the local variables and formal parameters with each method invocation?
6. What are the things that can be declared in a Java interface? Be specific with full access modifiers and keywords.

Academic Integrity Question

7. Why are professional engineers expected to act with integrity?

STYLE (20 points)
You will be specifically graded on commenting, file headers, class and method headers, meaningful variable names, judicious use of blank lines, not having more than 80 characters on a line, perfect indentation, no magic numbers/hard-coded numbers other than -1, 0, +1, **using setters and getters**, etc (see the style section of PA2's writeup).

CORRECTNESS (70 points)

For this assignment, you will extend your PA4 CosmicSphere program to include standard Java GUI components (JButton, JLabel, JSlider, JPanel) and standard Java Event Handling. You will want to reference Chapter 11 and the notes for Chapter 11!

Setting up your pa6 directory:

To begin the assignment you'll want to copy over your files from PA4.

```
    > mkdir ~/pa6
    > cp ~/pa4/CosmicSphere*.java ~/pa6
    > cp ~/pa4/*.jar ~/pa6
```

Now you can cd into pa6 and get started.

NOTE: Your PA4 program must work perfectly according to PA4 specifications before continuing. There WILL be deductions if your program has errors from PA4.

Begin the Coding! Huzzah!

1. Creating the Layout

The basic layout of the canvas with the horizontal and vertical lines and the cosmic spheres are the same along with their same functionality from PA4.

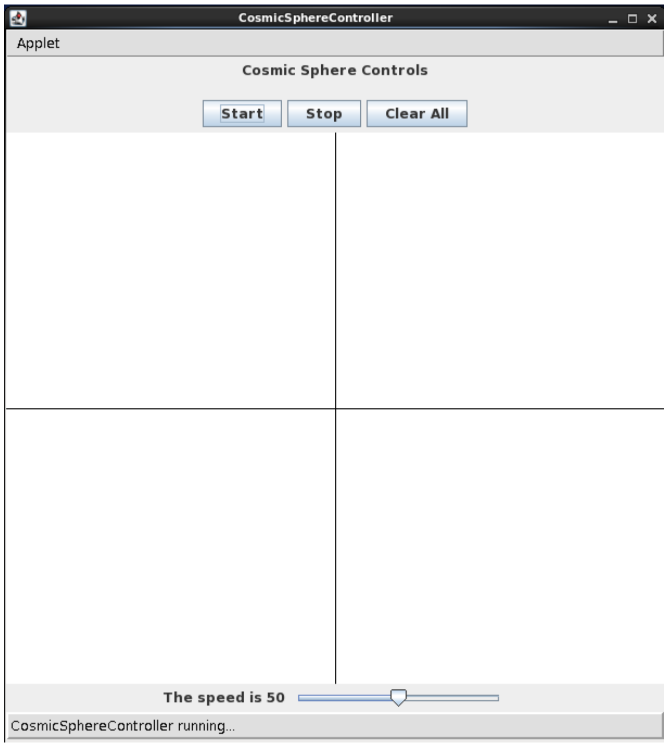
Things to add:

1. The **control panel** at the top with a centered label and three buttons.
   - Label: Cosmic Sphere Controls
   - Buttons: Start, Stop, Clear All

2. The **bottom panel** to control the speed of the cosmic spheres.
   - Label indicating the current speed
   - Slider to control the speed (minimum speed is 1, maximum speed is 100, default speed is 50)

Note: When creating this layout, create your buttons, panels, and sliders first. Don't forget to call **this.validate()** to set the GUI, and then create your horizontal and vertical lines. Do not worry about the event handling on the GUI components yet.

Program at startup:



The textbook is your best friend for this assignment!

2. Converting from objectdraw Event Handling to Standard Java Event Handling

Once you get the layout looking good, time to start using some standard Java event handling. You will need to change from using the objectdraw helper event handlers to the standard Java event handlers.

For example, change from using

```
public void onMouseClick( Location point )
```

to using (and defining)

```
public void mouseClicked( MouseEvent evt )
```

You will also need to adjust your class declarations so both CosmicSphereController and CosmicSphere implement the following interfaces:

```
MouseListener, MouseMotionListener, ActionListener, ChangeListener
```

Modify any code that used a Location object to use a MouseEvent object. Again, **Chapter 11 and the notes should be excellent guides**. You will need to provide an implementation for each abstract method declaration in the interfaces you implement. You will need to do this in both CosmicSphereController.java and CosmicSphere.java. **There should be no onMouse*() methods.**

However, we are still using objectdraw library objects like Line and FilledOval and their methods like contains() which takes a Location object. So when necessary, you will need to create Location objects from the MouseEvent's x and y - for example, evt.getX() and evt.getY() - see the Java API docs for MouseEvent and the objectdraw docs for Location.

Pro Tip: Bring your textbook and notes to the lab. They have pretty much everything you'll need to implement this assignment. You will probably need to use online docs for the objectdraw library and the standard Java API libraries.

3. Adding Standard Java Event Handling for the Buttons and Slider

Use only the event handling model discussed in class and in Chapter 11 in the textbook. **No inner classes** (private member inner classes or anonymous inner classes) for this assignment.

Top Control Panel:

	CosmicSphereController	**CosmicSphere**
Start Button	New spheres should start growing/shrinking upon creation	Current spheres should start growing/shrinking
Stop Button	New spheres should NOT start growing/shrinking upon creation	Current spheres should stop growing/shrinking
Clear All Button		Current spheres should remove themselves from the canvas. More info below.

Clarification on the Clear All Button: After clicking "Clear All", whichever mode ("Start" or "Stop") you were in before clicking "Clear All" is the mode you will stay in after clearing.

Bottom Speed Panel:

	CosmicSphereController	**CosmicSphere**
Speed Label	Update the value displayed when the slider is adjusted	
Speed Slider		Current spheres should adjust their growing/shrinking speed according to the current value of the slider. More info below.

Clarification on the speed slider: The smaller the speed value, the slower the cosmic shrinking and growing of the spheres; the higher the speed value, the faster the cosmic shrinking and growing of the spheres. You will have to do some calculation with the speed value extracted from the speed slider in order to have the speed of the spheres work correctly. When the speed is set to 100, we don't want the pause time (at the end of the forever loop in the run() method) to be zero, therefore a speed of 100 should correspond to a pause time of 1 millisecond, and a speed of 1 should correspond to a pause time of 100 milliseconds.

4. Adding New Features

At this point your program should be able to:

1. Perform all the functionality from PA4
2. Start and stop the spheres resizing
3. Add more spheres and start
4. Clear all and add and stop and clear some more and add ... etc.,

Once you get all that working, then you want to add the following features (remember to use the Java event handler methods instead of objectdraw where appropriate).

Changing the size of the canvas:

- Lines should stay proportional to where they were before adjusting the canvas size (same as PA4).
- Spheres should stay proportional to where they were before adjusting the canvas size (think about how you did this for the lines and apply this to the spheres).

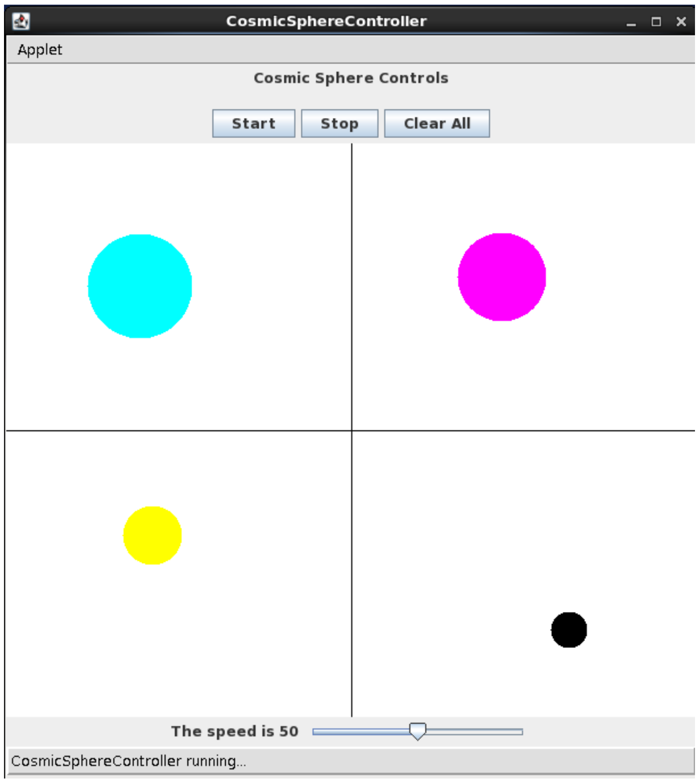
Dragging objects on the canvas:

- Use the same 5 pixel margins as in PA4 to prevent either line from being dragged off the canvas.
- Spheres can be grabbed and dragged around the canvas.
  - Spheres should change colors as they are dragged from one quadrant to another (based on the center of the sphere).
  - If spheres overlap each other you can grab multiple spheres in the overlapping area and drag them together around the canvas.
  - If a sphere overlaps one or both lines you can grab the sphere(s) and the line(s) at the same time and move them all together.
  - Do not allow the **center** of a sphere to be dragged past the 5 pixel margin.

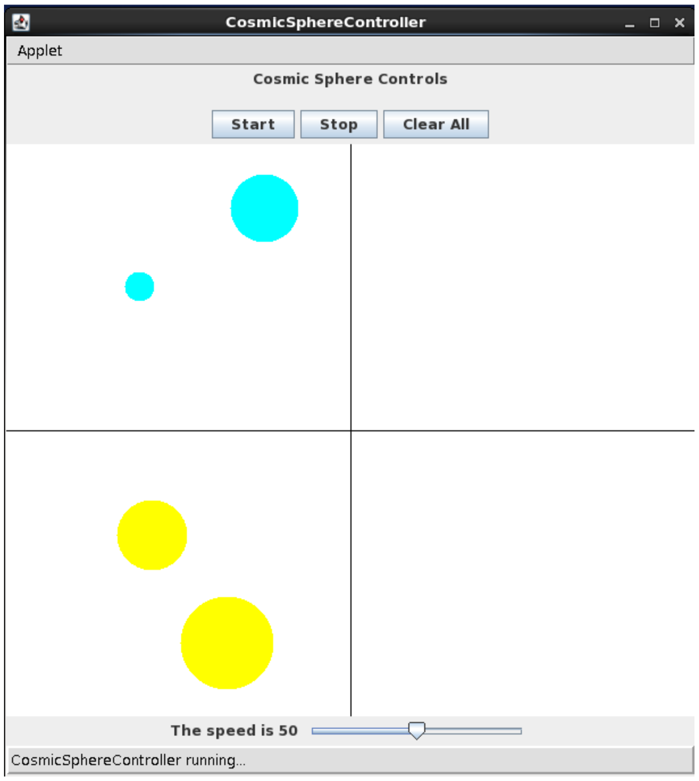
Implementation Hints:

- Some of you may want to implement the dragging event before the buttons and scroller event handling. It is up to you.
- You will probably need to modify the CosmicSphere constructor to deal with new information that needs to be sent to each sphere, so that each sphere can operate independently.
- There should be no mechanism (i.e., **no data structures**) in the controller object that knows anything about how many spheres there are or where they are or if they have been cleared from the canvas, etc. This means that each sphere handles the various events (mouse, button, scroller) on its own and makes its own decisions, based on things like where its center is located and how big it is in its resizing cycle.
- When a sphere is cleared from the canvas, it's run() method running in its own thread should end (change the forever loop in run() to have a condition that will fail when the sphere is cleared from the canvas).

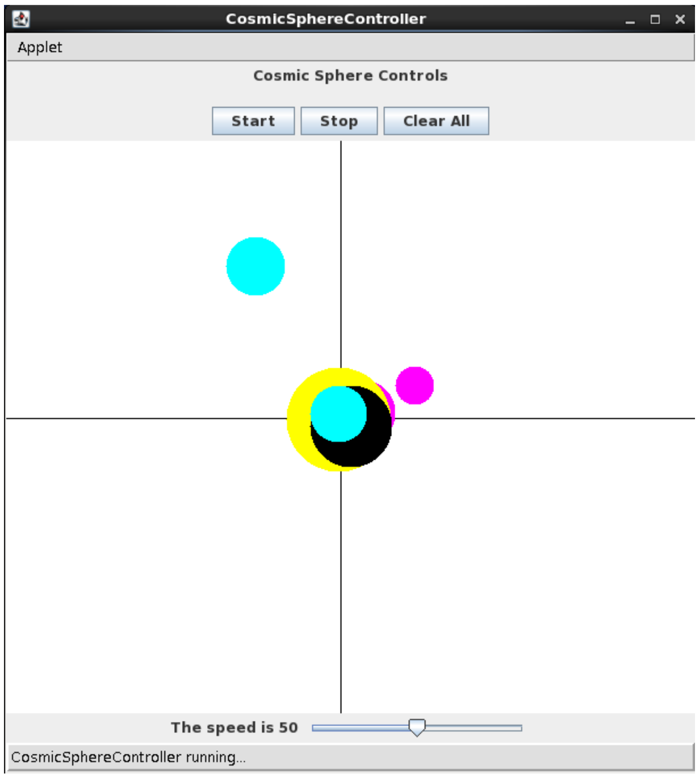
<u>**Sample Screenshots:**</u>



After clicking, spheres are created.

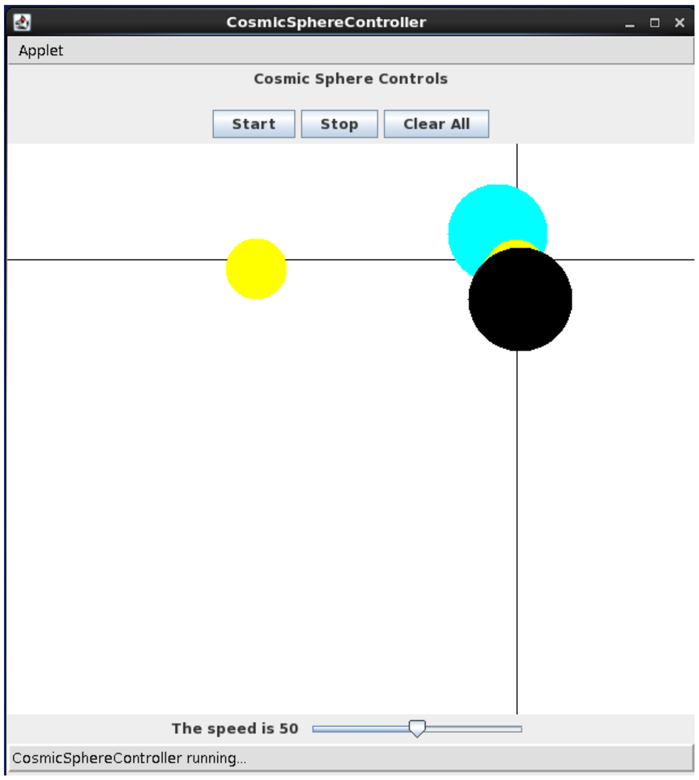


The spheres can be dragged around.



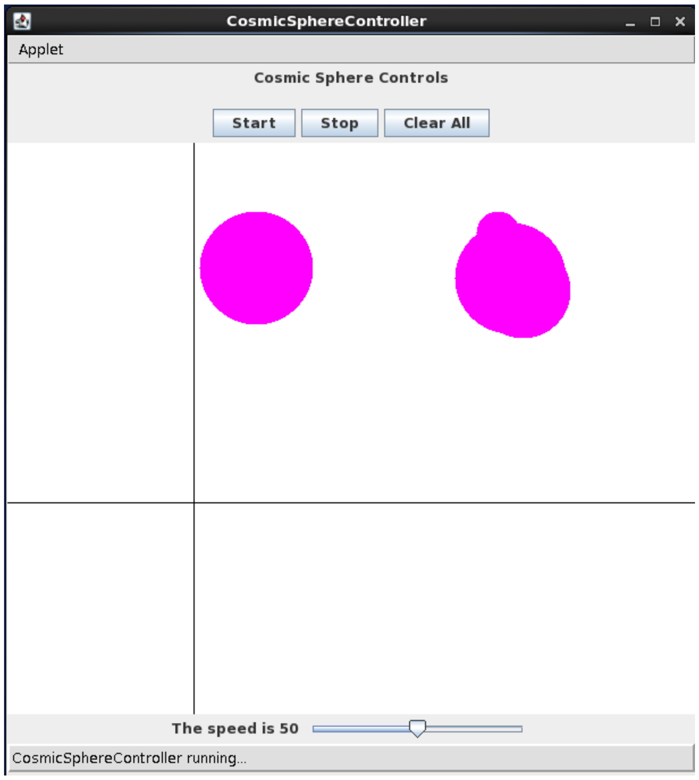
CosmicSphereController running...

Multiple spheres can be dragged at the same time.



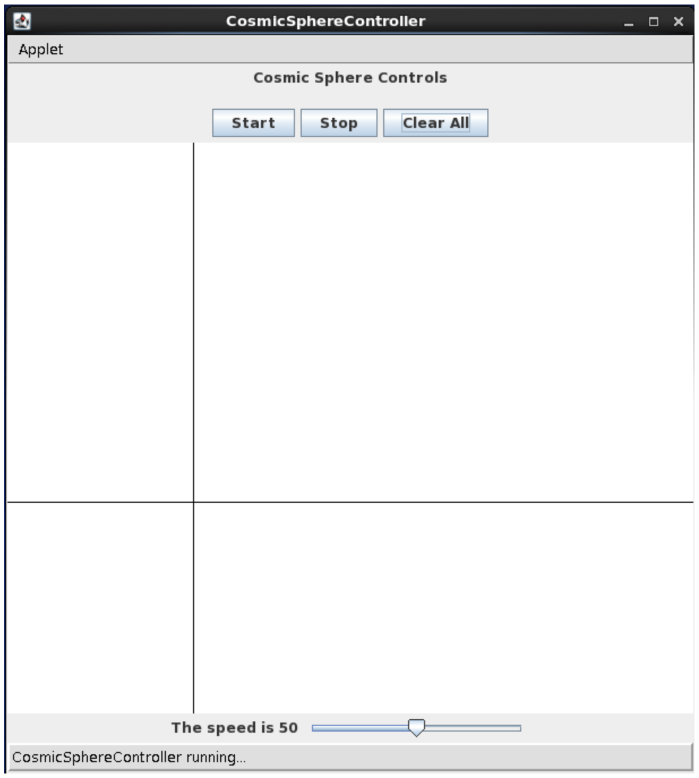
CosmicSphereController running...

Lines can be dragged at the same time as spheres.



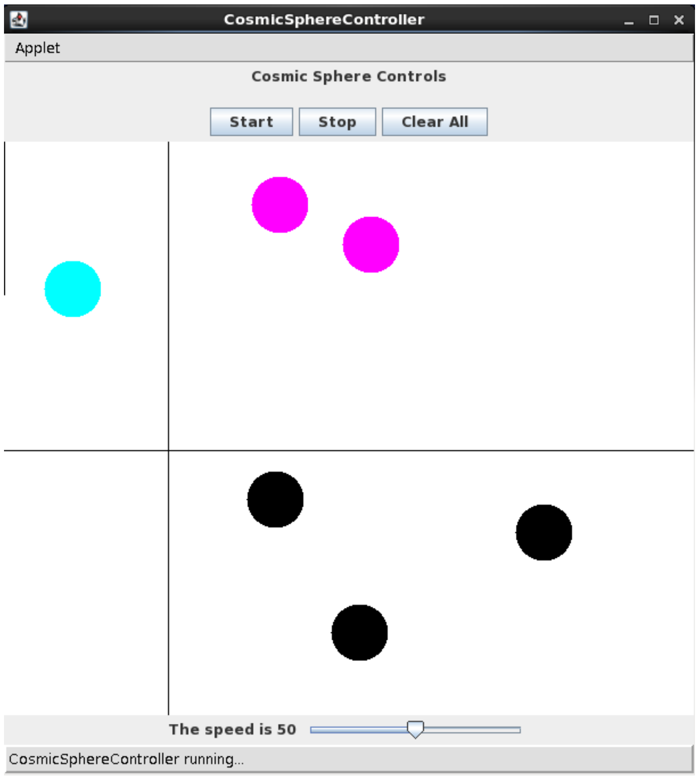
CosmicSphereController running...

The lines can also be dragged around on their own.

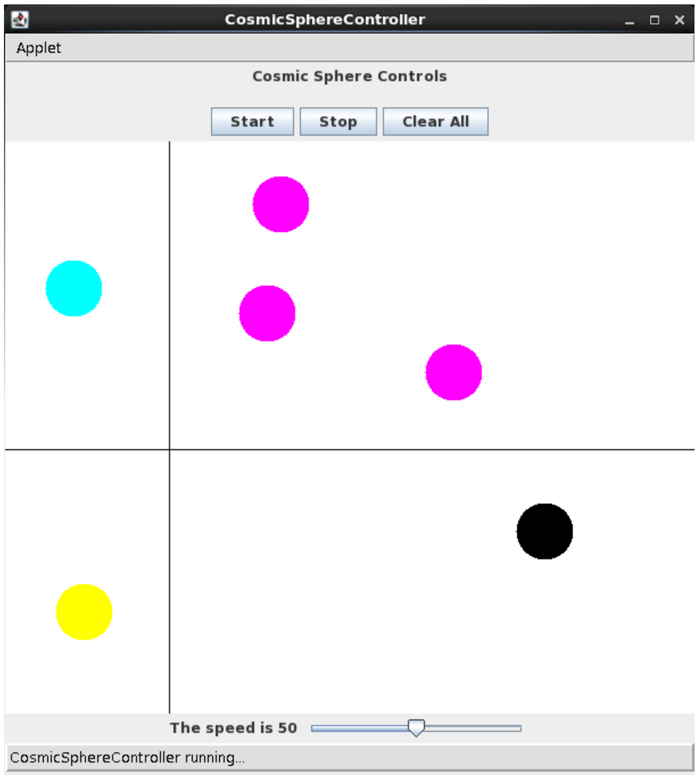


CosmicSphereController running...

After clicking the “Clear All” button, the spheres should be removed and the lines should stay in place.



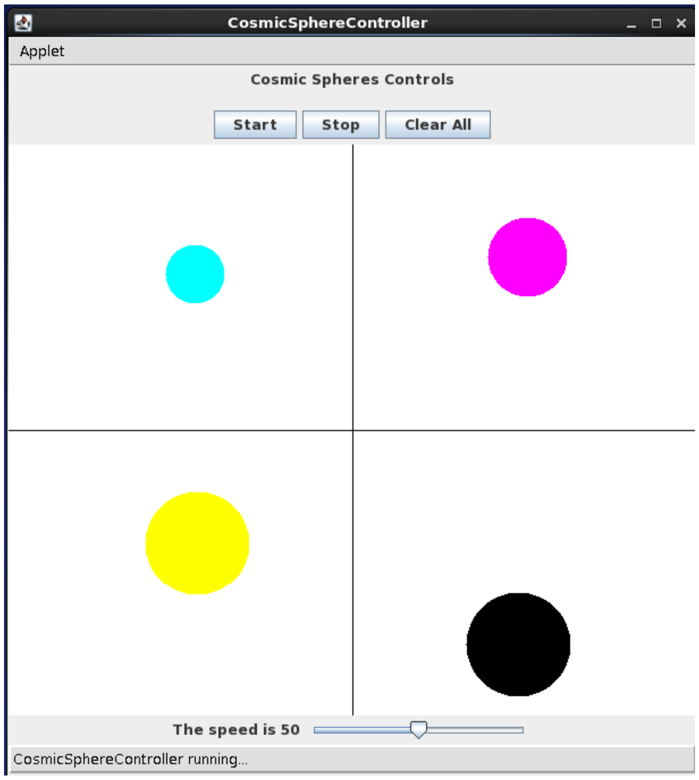
After clicking the “Stop” button, spheres can still be created, but they no longer grow.



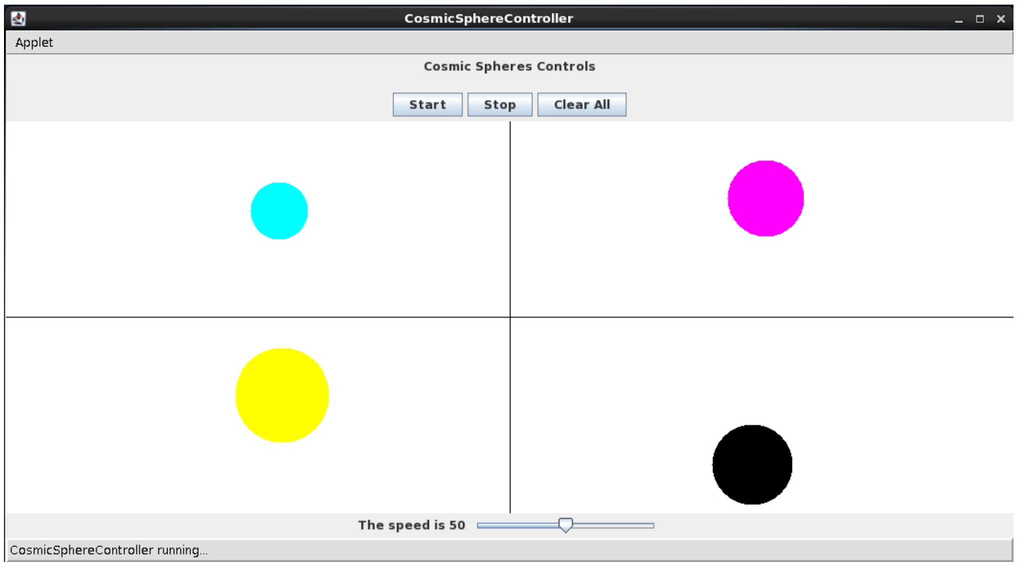
Lines and spheres can still be dragged in “Stop” mode.

When the size of the window is changed, both the lines and the center of the spheres should stay proportional to their original position.

Before adjusting window size:



After adjusting window size:



Turnin

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

```
> cse11turnin pa6
```

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 pa6
```

If you are unsure your program has been turned in successfully, 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:

- README
- Acme.jar
- Objectdraw.jar
- CosmicSphere.java
- CosmicSphereController.java

Extra files to be collected if you attempt the extra credit:

- EC_CosmicSphere.java
- EC_CosmicSphereController.java

Extra Credit (5 points)

For the extra credit, you will be implementing a black hole!

Setup: Create copies of your regular source files and name them **EC_CosmicSphere.java** and **EC_CosmicSphereController.java**.

Details: (please also see screenshots below)

The Black Hole Button:
First, replace the "Clear All" button with a "Black Hole" button. Upon clicking the black hole button:

- Disable all buttons (including the black hole button)
- Disable the speed slider
- Disable all actions triggered by mouse events (no dragging lines, no creating spheres, no dragging spheres)
- Create a black hole (black sphere with a diameter of 10 pixels) at the intersection of the lines
- All spheres should start to move towards the black hole (doesn't really matter how, it just needs to move into the black hole eventually)

Consuming Spheres with the Black Hole:
Once the center of a sphere is contained within the black hole, we want it to shrink to size 0, and be removed from the canvas. Once we remove it from the canvas we want to make the black hole bigger by 4 pixels since it "ate" the sphere. This means that the ending size of the black hole will differ depending on how many spheres it ate.

To reiterate:

- Once the center of a sphere is inside the black hole, it should start shrinking to size 0
- Once the size becomes 0, you should remove it from the canvas
- Every time the black hole "swallows" a sphere, the diameter of the black hole should grow by 4 pixels. (The center of the black hole should stay in the same spot)

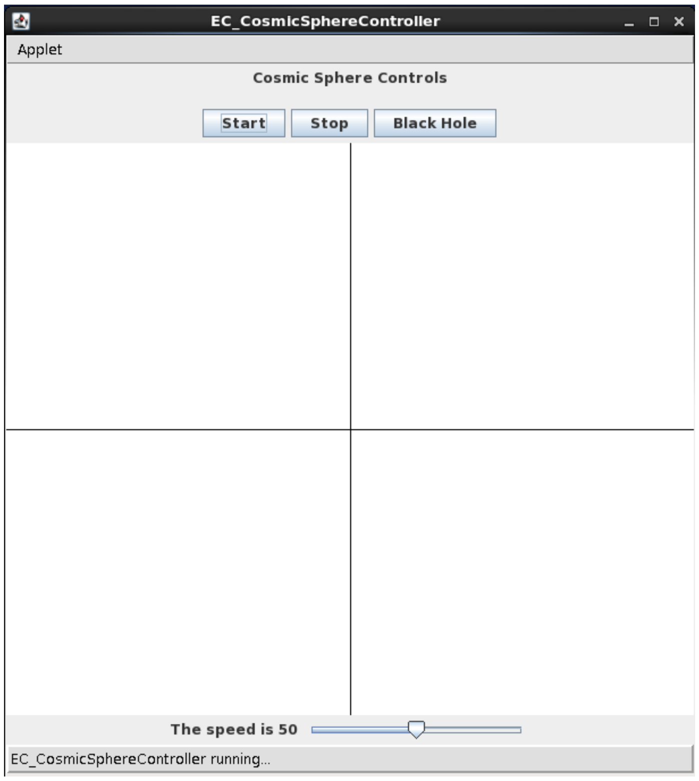
In order to make the black hole grow every time it eats a sphere, we need the following:

- A **static** int to keep track of the number of spheres created (increment this when you create a sphere, decrement this when a black hole eats a sphere)
- Because each sphere is running in its own thread, we need to **synchronize** the access to the static counter. This means we need two methods:
  - `public` **`synchronized static`** `void addSphere()`
    This method should increment the static variable keeping track of the number of spheres created. We want to call this method every time a sphere is created.
  - `public` **`synchronized static`** `void removeSphere( FilledOval blackHole )`
    This method should increase the diameter of the black hole by 4 pixels and decrement the static count of how many spheres there are on the canvas.

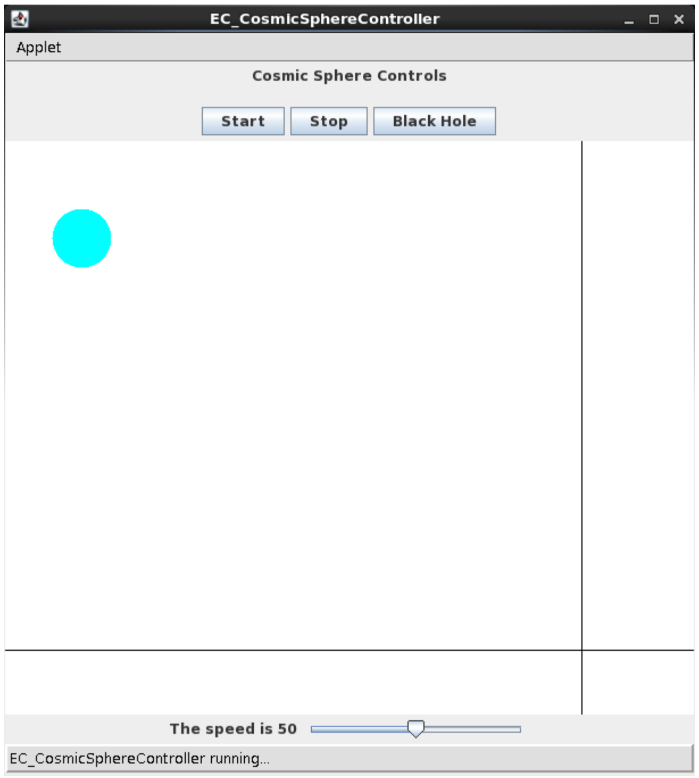
Lastly, after the black hole has consumed all of the spheres, reset the program to the original state after the **first mouse click** anywhere on the canvas. This means all the functionality that was enabled at the start of the program needs to be reenabled (buttons, sliders, clicking, etc.).

***Note: We will not be testing resizing the window for the extra credit.**

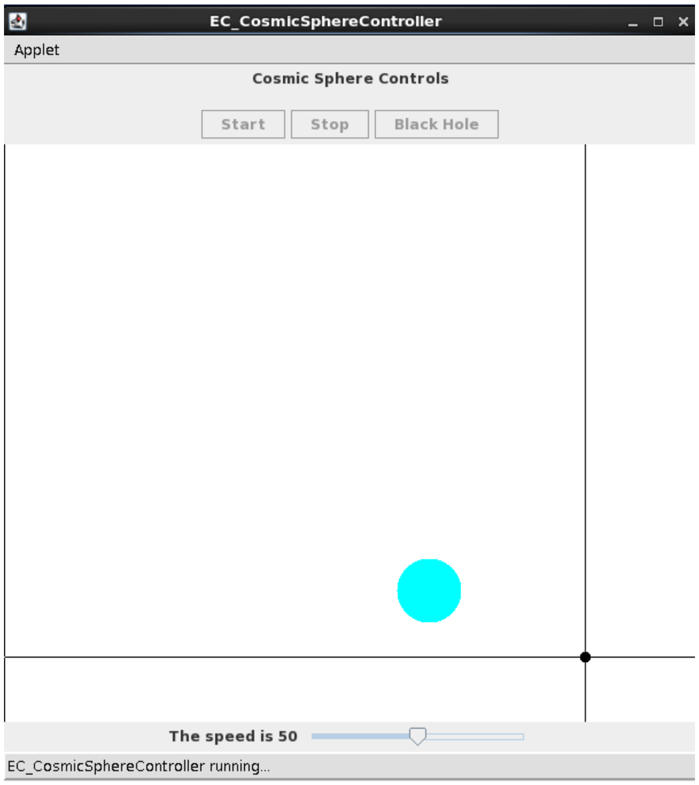
Sample Screenshots:



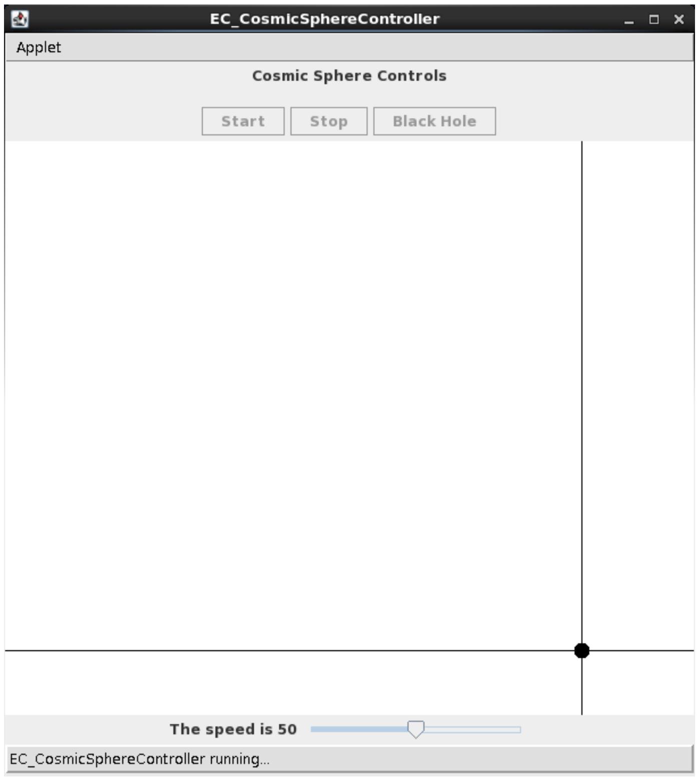
The program should start just like the non-extra credit version, with the "Clear All" button replaced by with a button named "Black Hole".



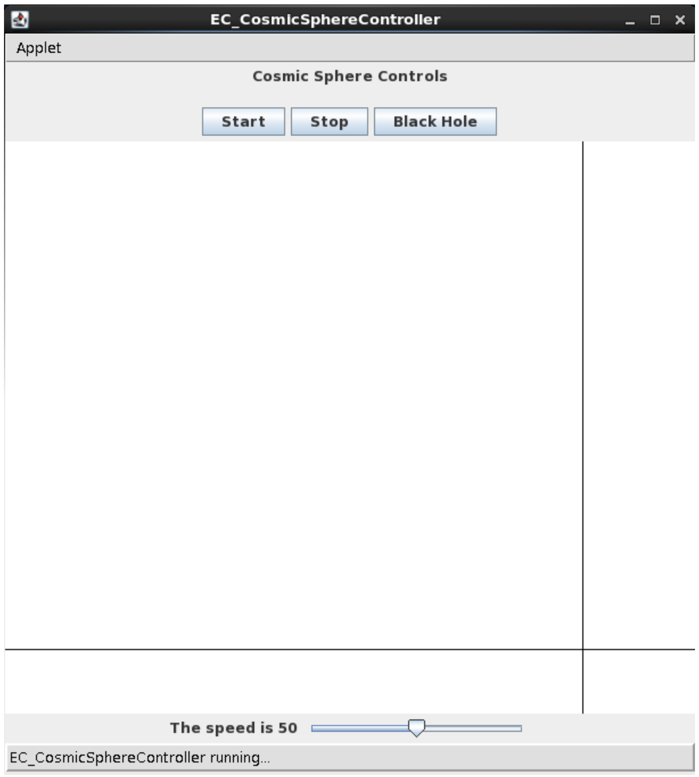
You can still create and drag spheres, and drag the lines.



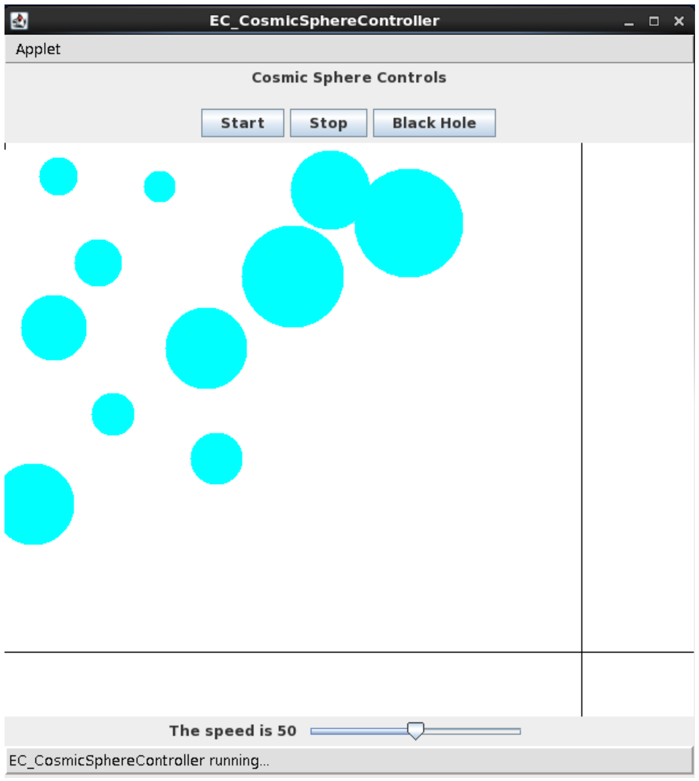
After clicking the black hole button, the buttons and slider are disabled, and the sphere starts moving towards the black hole.



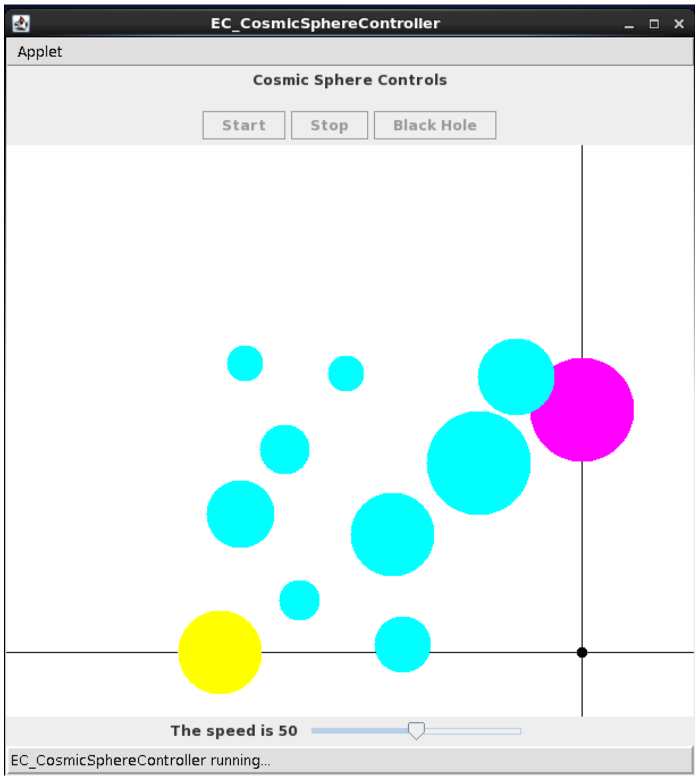
After the sphere has been eaten by the black hole, the black hole grows by 4 pixels and the buttons and slider are still disabled.



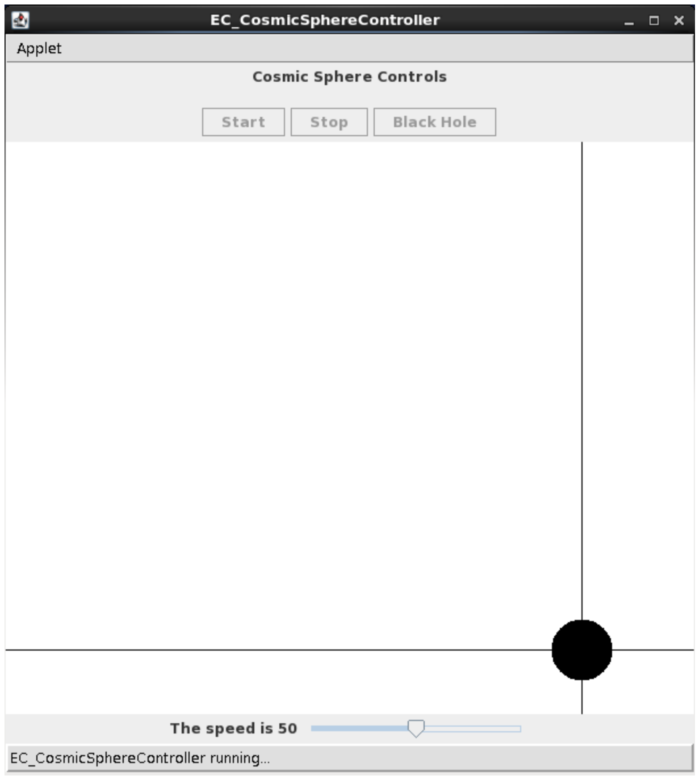
After clicking anywhere on the canvas, the buttons and slider are reenabled and the canvas is reset to the initial state.



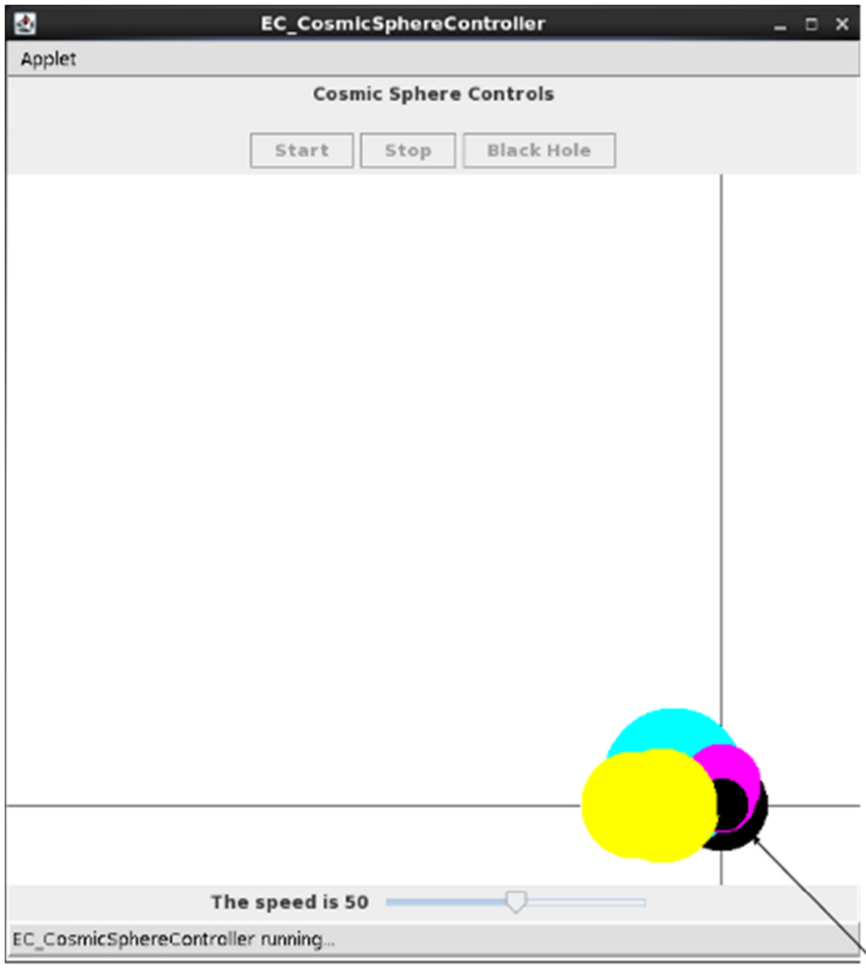
Let's try again with more cosmic spheres by clicking around on the canvas.



After clicking the black hole button, the buttons and slider are disabled, and the spheres start moving towards the black hole.



After all the spheres have been consumed by the black hole.



The black hole grows as it consumes the CosmicSpheres. Alas, they had no chance.