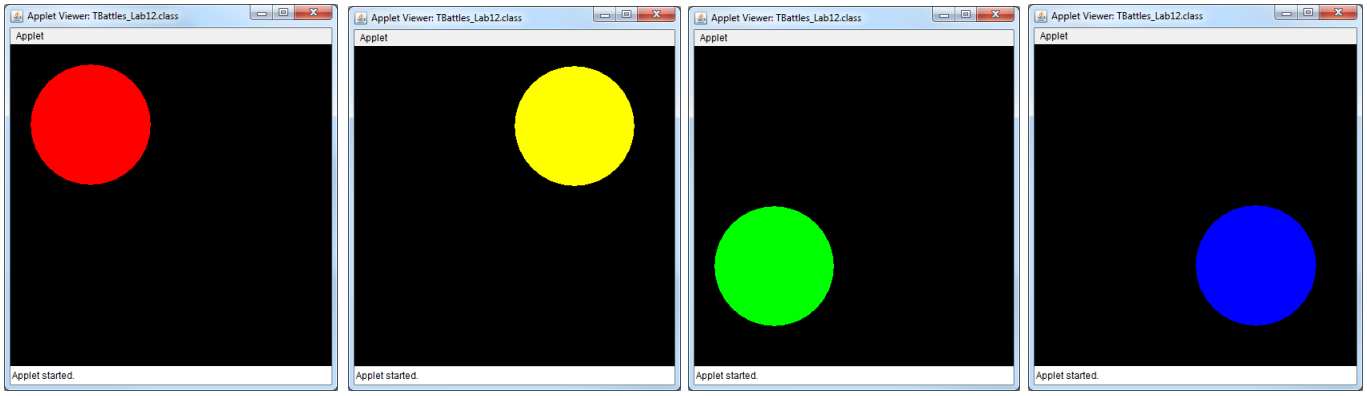
CIS 255 • Java Programming
Laboratory Assignment 12: Applets

In this laboratory assignment you will work with Java Applet syntax to create an applet that draws a circle with a different color in a different position every time the user clicks any location within the applet area.

Background

An applet can respond to various mouse actions within the applet area by including at least one `private` inner listener class for mouse events. The listener method can cause the coordinates for a particular shape to change when the applet area is repainted. Your applet will alternate between four different positions and four different colors for a filled circle, moving to the next position / color combination in the series when the user clicks anywhere within the applet area.



Step 1: Launch TextPad and Save a New Java File

Since a Java class for an applet does not follow the same structure as an application (a class with a `main` method), starting a Java file from scratch is best. Launch your source code editor (e.g., TextPad), and save the file as a Java file to ensure that the program highlights the code according to Java syntax rules. Make sure you include your first initial and last name and the laboratory assignment number as usual (**FLast_Lab12.java**) and that you specify the file location.

Step 2: Begin Your Code with Comments, Import Statements, and Inheritance

Add comments at the top of the code to include your name, the course number, the laboratory assignment number, and a brief description of the program's contents. You may also wish to include the date.

The class needs to import several packages. Use wildcard `import` statements for the packages `javax.swing`, `java.awt`, and `java.awt.event` (the sub-package must be imported separately).

The class you are writing will be derived from `JApplet` so that you have access to all applet-related syntax. Write the class header for your file name with an `extends` clause for `JApplet`.

As you continue your code, you may go ahead and add comments to the statements or groups of statements that you believe deserve additional explanation, or you may wait until after you have written your entire class definition to add comments.

Step 3: Declare the Fields

The following should be declared as `private` members of the class (not locally in a method); assign the initial values for these fields in their declarations:

Description	Data Form	Initial Value
The fill color for the circle	`Color` object	`Color.red`
How many clicks have been performed	Whole number	1
The current x coordinate	Whole number	25
The current y coordinate	Whole number	25
The width of the circle (constant)	Whole number	150
The height of the circle (constant)	Whole number	150

You must declare a field of type `Color` to store the color of the circle to be drawn, as the `paint` method will use the value of this variable when drawing the circle. The listener method will assign a different value to this variable as the user clicks the mouse. The value assigned to a `Color` object can be any of the `Color` constants discussed in chapter 12.

Do not put these declarations inside a method! They should only be within the brackets for the class. Note that you can assign initial values to member variables as they are declared.

Step 4: Define the `init` Method to Set Up the Applet

A class written for an applet does not need a constructor: instead, it contains a `public void` method named `init` with no parameters. Write the header for this method and the brackets for the body.

The body of this method requires two statements. First, obtain the content pane for the applet, and set its background color to black. Second, add a mouse listener: the argument to the method should be a new object of a mouse listener class to be written later (for now, come up with a name for this class that indicates that it will respond to the user's clicks on the applet area).

Step 5: Define the `paint` Method to Draw the Circle

The method that draws shapes on the applet area is a `public void` method named `paint`, and its parameter is a `Graphics` object, `g`. Write the header and brackets for this method.

Within these brackets, the first step that must be taken is to call the version of `paint` from the superclass (the class being extended, `JApplet`). When calling the superclass version of a method that the class is overriding, precede the method name with the keyword `super` and the dot operator. Perform this superclass method call with the `Graphics` object as the argument.

Failure to call the superclass `paint` method may hinder the applet from displaying the shapes in the method as desired!

With that done, the method can draw the circle. First, set the color for the `Graphics` object to the `Color` object field declared in the class. (This object's value will change within the listener method.)

The argument to this method call should be the name of the `Color` field declared in the class, not the value `Color.red`! This allows the applet to use a different color for the oval in each position as the user clicks the mouse button.

Then, use the `Graphics` object to draw a solid oval with the x and y coordinates and the width and height from the member variables and constants.

The coordinates for the oval are based on a rectangle that bounds the oval; the x and y coordinates are the upper-left corner, and the width and height are that of the rectangle.

Step 6: Define the Private Inner Listener Class

In order for the applet to respond to the mouse clicks, the class must have a `private` inner listener class that contains mouse-related methods. The class could implement the `MouseListener` interface, but that would entail writing definitions for all five mouse actions included in that interface, and this applet should respond to only one of these actions. Instead, have the class extend the `MouseAdapter` class that already includes empty bodies for all five methods; the class then requires only a definition for the one action the applet should respond to, a `public void` method named `mouseReleased` with a `MouseEvent` parameter named `e`.

Overloading is writing several methods in a single class with the same name and different parameter lists; writing a new definition for a method in a derived class with the exact same header as a method definition in the base class is called *overriding* instead of overloading.

Overriding `mouseReleased` rather than `mouseClicked` ensures that the applet responds to the user's click even if the mouse moves as the user holds the button down.

The initial action in this method is to set the color, x coordinate, and y coordinate for the next oval based on how many clicks have been performed before the current click:

Current Value of Clicks Field	New Color	New X	New Y	New Value of Clicks Field
1	Yellow	200	25	2
2	Green	25	200	3
3	Blue	200	200	4
4	Red	25	25	1

Use a series of `if` and `else if` statements to check the current value of the fields storing the number of clicks. Within each `if` statement block, set the color, x coordinate, y coordinate, and new value for the number of clicks.

Finally, to trigger the clearing of the canvas and the drawing of the new circle, invoke the method `repaint` with no arguments.

Do not attempt to call `paint` directly! To make changes to the canvas, invoke the method `repaint` instead.

Although the user's click on the applet area triggers the drawing of the new oval, the call to draw the oval should be within the method `paint`, not within the method `mouseReleased`. By calling `repaint`, `mouseReleased` ensures that the next oval is drawn properly.

Step 7: Compile Your Code

Compile your Java class and correct any syntax errors as the compiler finds them. Remember, however, that you do not "execute" an applet by itself: you embed it in an HTML file.

Step 8: Create an HTML File that Embeds the Applet

Follow one of the examples in the textbook (e.g., Code Listing 14-7) to create an HTML file with the applet (.class file) embedded in the body of the page. Give the applet area both a width and a height of 375 pixels. The page must have a title; add headings, paragraphs, and other HTML elements if desired. Save the web page using the file name **FLast_Lab12.html** (note the different filename extension).

Step 9: Launch the Web Page in a Web Browser

Open the HTML file in a Web browser. If the browser indicates that the Java applet is blocked, allow the browser to display the applet. Click on the applet area multiple times and make sure that the applet draws the circle with a different color in a different location according to the sequence in step 6. **Make sure that the sequence repeats** (after the blue circle in the lower-right corner, the applet should return to the red circle in the upper-left corner).

Step 10: Add Inline Comments and Submit Your Code

Make sure that your Java source code has **three or more inline comments** near the statements that you believe warrant the additional description (especially those related to applet programming). These can be simple line comments at the end of a line of code that needs explanation, or block comments where each precedes a series of statements described in the comment. You are not required to add comments to the HTML file, as these have different syntax than Java comments.

Follow the instructions on the course website to attach and submit your source code.

Make sure you complete the submission process! If you attach your file without submitting the assignment, your file will not be received!