CECS 174 – Lecture 31 – Applets Continued

Applets – An applet has special functions that it uses to create, run, and stop the applet. They are called when it is first loaded, started, halted, and unloaded. Applets use these methods rather than requiring a main(). They are:

`init()` – this method is called when the applet is first loaded. It usually has initialization functions to carry out.

`start()` – this method is called when the applet is started or restarted. It begins any processes that are placed here.

`stop()` – this method gives your applet the ability to paused or halted. After being halted, it will not be reinitialized, just restarted.

`destroy()` – this is a method that is called when the applet is closed. It should clean up any objects that were created and used.

These methods can be especially helpful when creating animated objects in an applet. Creating an animation requires that the window is refreshed many times a second, this requires a timer to keep track of the amount of time until the next frame change. These methods can be used to create, start, and stop a timer for our animation

Animated Applet Example:

Create an applet that displays an animated ball bouncing around the screen:

1. Information we need:
    a. size of the ball (25 pixel radius)
    b. location of the ball (init at 50, 50)
    c. speed of the ball. (5 pixels per tick)
    d. color of the ball (gray)
    e. size of the applet window (defined by window)
    f. color of the applet window (pink)
2. Things we need:
    a. timer – to produce the animation of the ball
    b. drawing tools
        i. circle
        ii. color
3. Things we need to do
    a. initialize ball and window variables
    b. move the ball
    c. test if the ball hit a wall

Timer – The timer is used to schedule tasks that will be executed at a given time interval. We need to import the java.util.Timer class in order to use it.

```
Timer timer = new Timer();
```

Task – A set of statements that will be executed by the timer that scheduled the task.

```
TimerTask task = new TimerTask(){
     public void run(){
          …
     }
}
```

A timer executes a task by using the schedule method. The task is executed starting at the first time (0) and then repeated every given number of milliseconds (50). (Note - there are 1000 milliseconds in 1 second).

```
timer.schedule (task, 0, 50);
```

When we are done with a task, usually when we are done with the program, we want to cancel the task and stop the timer.

```
task.cancel();
timer.purge();
timer.cancel();
```

Graphics – to draw some shapes, we need to set up the drawing tools.

```
import java.awt.*;

public void paint(Graphics g)
```

Colors – to set the color of the drawing pen. Anything drawn after that statement will be that color. If you change the color of the pen, everything after that will be the new color.

```
g.setColor(Color.GRAY);
```

Drawing – to draw a shape using one of the drawing methods.

```
g.fillOval(0, 0, 10, 10); //x, y, width, height
```

Background – when creating an animation, you'll often notice that the shapes that were previously drawn to the screen are still there. This can often create some neat drawings, but it is not the effect that we are going for in this applet. This can be remedied by simply redrawing the entire background over the old image and then drawing the shapes on top for each frame of the animation.

```
g.setColor(Color.PINK);

g.fillRect(0, 0, getWidth(), getHeight());
```

The complete program:

```
public class Ball {
	private int x; //Position - X is horizontal dir
	private int y; //Position - Y is vertical dir
	private int movX; //speed & direction ball is moving
	private int movY;
	private int radius; //size of ball

	public Ball(int r, int s){
		x = 50;
		y = 50;
		movX = s;
		movY = s;
		radius = r;
	}
	public int getX(){
		return x;
	}
	public int getY(){
		return y;
	}
	public int getMovX(){
		return movX;
	}
	public int getMovY(){
		return movY;
	}
	public int getRadius(){
		return radius;
	}
	public void setX(int xNum){
		x = xNum;
	}
	public void setY(int yNum){
		y = yNum;
	}
	public void setMovX(int xNum){
		movX = xNum;
	}
	public void setMovY(int yNum){
		movY = yNum;
	}
}
```

```
import javax.swing.*;
import java.util.*;
import java.util.Timer;
import java.awt.*;

public class Bouncy extends JApplet {
	Ball b;
	Timer timer; //timer allows for animation of the ball
	TimerTask task;

	public void init(){
		//initializing stuff
		setSize(400, 500); //set size of applet window
		b = new Ball(25, 5);
		//create timer
		timer= new Timer();
	}

	public void start() {
		//the timer is started when the page is loaded
		//the run() method of the timer class is called by the
		//timer every time interval
		task= new TimerTask() {
			public void run() {
				move();
			}
		};
		// timer starts at time 0 and ticks every .05 seconds
		timer.schedule( task, 0, 50 );
	}

	public void stop() {
		//stop when the user closes the applet
		task.cancel();
		timer.purge();
	}

	public void destroy() {
		//this is cleanup, destroy everything we created when
		//the program ends
		timer.cancel();
	}
```

```
	public void move() {
		//move ball
		b.setX( b.getX() + b.getMovX() );
		b.setY( b.getY() + b.getMovY() );

		//change direction if it hits the wall
		//getWidth() and getHeight() return the width and
		//height of the applet's window
		if (( b.getX() - b.getRadius()) <= 0){

			b.setMovX( b.getMovX()*-1 );

		}else if((b.getX() + b.getRadius()) >= getWidth()){
			b.setMovX( b.getMovX()*-1 );
		}

		if (( b.getY() - b.getRadius()) <= 0){

			b.setMovY( b.getMovY()*-1 );

		}else if((b.getY() + b.getRadius()) >= getHeight()){
			b.setMovY( b.getMovY()*-1 );
		}
		repaint();
	}

	public void paint(Graphics g) {
		//clear background
		g.setColor(Color.PINK);
		g.fillRect(0, 0, getWidth(), getHeight());
		//draw ball
		g.setColor(Color.GRAY);
		g.fillOval( b.getX() - b.getRadius(), b.getY()-
		b.getRadius(), b.getRadius()*2, b.getRadius()*2);
	}
}
```