Unit Nine Lecture Three:

Topic 1: Abstract Classes

"An abstract class is a class for which you cannot create objects. (A class for which you can create objects is sometimes called a concrete class.)" Note, you can still have an object reference whose type is an abstract class.

example: (Assume MyShape is an abstract class)

```
MyShape shape1;  // is acceptable - just an object reference

MyShape shape1 = new MyShape(2,3,8,9);  // not acceptable -
                                        // trying to create
                                        // an object
```

Classes that are declared to be abstract usually contain abstract methods.

"The reason for using abstract classes is to force programmers to create subclasses. By specifying certain methods as abstract, you avoid the trouble of coming up with useless default methods that others might inherit by accident."

"Abstract classes differ from interfaces in an important way - they can have instance fields, and they can have some concrete methods.

Topic 2: Abstract Methods

An abstract method has no implementation (no code beyond the header statement). This forces subclasses of the abstract class to specify concrete implementations of this method.

Topic 3: An example of an abstract class and a subclass

```
// This is an abstract super class.

import java.awt.Graphics;

public abstract class MyShape
{
      private int xCoor1;     // Instance Fields
      private int yCoor1;
      private int xCoor2;
      private int yCoor2;

      public MyShape(int x1, int y1, int x2, int y2)  // parametric constructor
      {
            xCoor1 = x1;
            yCoor1 = y1;
            xCoor2 = x2;
            yCoor2 = y2;
      }

      public int getxCoor1()    // accessor methods
      {
            return xCoor1;
      }

      public int getyCoor1()
      {
            return yCoor1;
      }

      public int getxCoor2()
      {
            return xCoor2;
      }

      public int getyCoor2()
      {
            return yCoor2;
      }

      public abstract void draw(Graphics g);
      // This is an abstract method, it contains no code.
      // All classes that inherit from this super class must have
      // a concrete draw method.  This is setting up polymorphism -
      // depending on what object is being used, the super class
      // knows what draw method to call.
}
```

```
// This is a subclass of the abstract super class.

import java.awt.Graphics;
import java.awt.Color;

public final class MyRectangle extends MyShape
{
      public MyRectangle(int x1, int y1, int x2, int y2)
      {
            super(x1, y1, x2, y2);
      }

      public void draw(Graphics g)   // required of all subclasses of
      {                              // the abstract class MyShape
            int a,b,c,d;
            int width, height;
            int minX, minY;

            a = getxCoor1();
            b = getyCoor1();
            c = getxCoor2();
            d = getyCoor2();

            minX = Math.min(a,c);
            minY = Math.min(b,d);

            width = Math.abs(c - a);
            height = Math.abs(d - b);

            g.setColor(Color.blue);
            g.drawRect(minX, minY, width, height);
      }
}
```

Topic 4: Calling the super of JFrame

If you have a class that extends JFrame, such as...

```
public class MyClass extends JFrame
```

and if in the default constructor of that class you call the super of JFrame, such as...

```
public MyClass()
{
      super("Unit 10 Assignment 3");
```

It will print Unit 10 Assignment 3 in the Title Bar of the JFrame.

Setting the Background of a JFrame

```
import javax.swing.JFrame;
import java.awt.Graphics;
import java.awt.Color;
import java.util.ArrayList;
import java.awt.Container;

public class U9A3 extends JFrame
{
      private ArrayList<Vehicle> list = new ArrayList<Vehicle>();

      public u9a3()
      {
            super("Unit 9 Assignment 3");

            Container container = getContentPane();
            container.setBackground(Color.yellow);
            .
            .
            .
      }

      public static void main(String args[])
      {
            u9a3 window = new u9a3();

            window.setSize(500, 500);
            window.setVisible(true);


            window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      }

      public void paint(Graphics g)
      {
            super.paint(g);   // colors the background of the JFrame

            for (Vehicle x : list)
            {
                  x.draw(g);
            }
      }
}
```

Unit Nine Assignment Three: Cars & Trucks