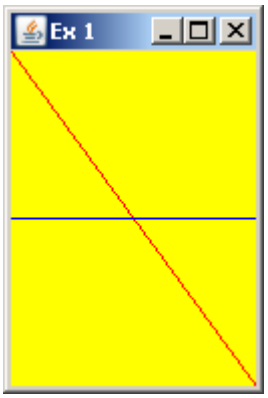
AP CS Unit 6: Inheritance Programs

1. In this exercise I will give you all the code and ask you to run it. Then you must answer questions about the code. The later exercises will build on what you learn here.

This project consists of two files.

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

public class Ex1 {
	public static void main( String [] args ) {
		JFrame f = new JFrame();
		f.setSize( 200, 300 );
		f.setTitle("Ex 18");
		f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		Panel_1 p = new Panel_1( Color.yellow  );

		Container pane = f.getContentPane();
		pane.add( p );
		f.setVisible( true );
	}
}
```

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

public class Panel_1 extends JPanel {
	public Panel_1( Color c ) {
		setBackground( c );
	}

	public void paintComponent( Graphics g ) {
		super.paintComponent( g );
		int w = getWidth();
		int h = getHeight();

		g.setColor( Color.red );
		g.drawLine( 0, 0, w, h );
		g.setColor( Color.blue );
		g.drawLine( 0, h/2, w, h/2 );
	}
}
```



When you run the program it should generate a figure like the one shown on the previous page. As you resize the window, the lines should adjust so that the red line always goes from diagonally from corner to corner and the blue line is always across the middle.

Answer the following questions. You will need to go the Java API (Google "Java API 6") to answer most of the questions.

- The Panel_1 class refers to three methods setBackground, getWidth, and getHeight.

  Where are they defined? _____

  _____

- The drawLine method has 4 formal parameters: what do they represent? _____

  _____

- What is the effect if you delete the line super.paintComponent(g);

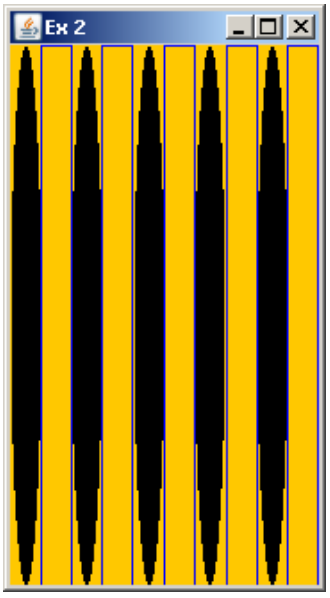
  _____

You may notice that the paintComponent method is never actually called by anything in this program. The JVM is actually running the program and when it detects any event that requires the window to be redrawn, the JVM calls the paintComponent method.

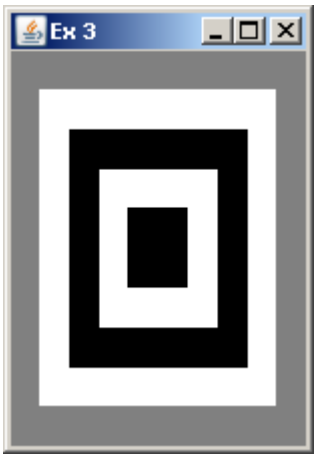
2. The graphics class has 4 methods from drawing rectangles and ovals. Here are their signatures.

```
void drawOval( int x, int y, int width, int height )    // draws the outline of an oval
void drawRect( int x, int y, int width, int height )    // draws the outline of a rectangle
void fillOval( int x, int y, int width, int height )    // draws a solid oval
void fillRect( int x, int y, int width, int height )    // draw a solid rectangle
```

In this exercise your panel object should draw alternating ovals and rectangles. The ovals should be solid and the rectangles not solid. They should alternate color as well. If the window is resized the shapes should resize as well. The two figures are of the same program but with the window resized.



3. In this program the panel has a background color and 4 solid rectangles, one inside the other and with alternating colors. As you resize the window, the rectangles change as shown.



4. You will need three files. Here are the first two.

```
public class Coin{
    private int value;

    public Coin( int v ){
        value = v;
        System.out.println( value );
    }

    public int getValue(){
        return value;
    }
}
```

```
public class MagicCoin extends Coin {
    private boolean lucky;

    public MagicCoin( int v ){
        super( v );
        lucky = Math.random() < 0.5;
        System.out.println( lucky );
    }

    public boolean lucky(){
        return lucky;
    }
}
```

Complete the following file.

```
public class RunCoins{
    public static void main( String [] args ){
        Coin [] c = new Coin[10];
        for ( int n=0; n<10; n++ )
            c[n] = get();

        add the value of all 10 coins and display this

        calculate the number of lucky magic coins in the array
    }

    private static Coin get(){
        50% of the time this returns a regular coin and 50% of the time it returns a magic
coin.  The value of any coin is a random number between 5 and 20
    }
}
```

5. In this project you will have 5 classes. The first two are the Animal and Ape classes. You need to finish the Ape constructor.

```
public class Animal {
        private int weight;

        public Animal( int w ) {
                weight = w;
        }

        public int getWeight(){
                return weight;
        }

        public void speak(){
                System.out.println("??");
        }
}
```

```
public class Ape extends Animal {
        public Ape( int w ){
                you write this
        }

        public void speak(){
                System.out.println("I'm an ape");
        }
}
```

The third class is the Zebra class which is very similar to the Ape class except its speak method prints out the message "I'm a zebra"

The fourth class is the Zoo class. It represents a very small zoo - it only has 3 cages.

The add method allows the client to add an Animal to the zoo and select its cage. If *cage* is 1, then assign *a* to a1. Same deal for values 2 or 3. Do nothing if cage is less than 1 or greater than 3

The totalWeight method returns the total weight of the all the animals in the zoo. If a cage is empty (in other words a1, a2, and/or a3 is null), then that counts as zero.

The speak method calls the speak method for each animal in the zoo. If a cage is empty, display "Cage 1 is empty" (or Cage 2 or 3 as appropriate"

```
public class Zoo {
        private Animal a1, a2, a3;

        public Zoo() {
                a1 = null;
                a2 = null;
                a3 = null;
        }

        public void add( Animal a, int cage ){
                // your code
        }

        public int totalWeight() {
                // your code
        }

        public void speak(){
                // your code
        }
}
```

The fifth class is shown below.

public class Runner {
  public static void main(String[] args) {
  Zoo z = new Zoo();
  z.add(new Ape(800), 2);
  z.add(new Zebra(300), 3);
  System.out.println("total wt: " + z.totalWeight());
  z.speak();
  System.out.println();
 	
  z.add(null, 2);
  System.out.println("total wt: " + z.totalWeight());
  z.speak();
  System.out.println();
 	
  z.add(new Zebra(250), 1);
  System.out.println("total wt: " + z.totalWeight());
  z.speak();
  }
}
	This should print the following:
	
	total wt: 1100
	Cage 1 is empty.
	I'm an ape
	I'm a zebra
	
	total wt: 300
	Cage 1 is empty.
	Cage 2 is empty.
	I'm a zebra
	
	total wt: 550
	I'm a zebra
	Cage 2 is empty.
	I'm a zebra

6. This program involves five interacting Classes.

```
public class Valuable {
        private double value;

        public Valuable(double v) {
                value = v;
        }

        public double getValue(){
                return value;
        }

        public boolean equals( Object obj ){
                // returns false if obj does not refer to a Valuable object
                // returns true if obj is a Valuable of equal value; otherwise false
        }
}

public class Jewel extends Valuable{
        private String color;

        public Jewel( double w, String c) {
                super(w);
                color = c;
        }

        public String getColor(){
                return color;
        }
```

```
	public boolean equals(Object obj){
		// returns false if obj does not refer to a Jewel object
		// returns true if obj is a Jewel of equal value and equal color; otherwise false
	}
}

public class Gold extends Valuable {
	private int weight;

	public Gold( double v, int w ) {
		// you write this
	}

	public int getWeight(){
		return weight;
	}

	public boolean equals(Object obj){
		// returns false if obj does not refer to a Gold object
		// returns true if obj is a Gold of equal value and equal weight; otherwise false
	}
}
public class TreasureChest {
	private Valuable v1, v2, v3;

	public TreasureChest() {
		empty();
	}

	public void empty(){
		v1 = null;
		v2 = null;
		v3 = null;
	}

	public void add( Valuable v ){
		if ( v.equals( v1 ) ){
			System.out.println( "This equals v1 and cannot be added" );
			return;
		}
		if ( v.equals( v2 ) ){
			System.out.println( "This equals v2 and cannot be added" );
			return;
		}
		if ( v.equals( v3 ) ){
			System.out.println( "This equals v3 and cannot be added" );
			return;
		}

		if ( v1 == null )
			v1 = v;
```

```
            else if ( v2 == null )
                v2 = v;
            else if ( v3 == null )
                v3 = v;
        }

        public double getValue(){
            // returns the total value of all the Valuables; if none then it returns 0
        }

        public int getGoldWeight(){
            // returns the total weight of all the gold; if there is no gold then it returns 0
        }

        public String getJewelColors(){
            String temp = "";

            if ( v1 instanceof Jewel )
                temp = ((Jewel)v1).getColor() + "  ";
            if ( v2 instanceof Jewel )
                temp += ((Jewel)v2).getColor() + "  ";
            if ( v3 instanceof Jewel )
                temp += ((Jewel)v3).getColor();

            temp = temp.trim();             // removes any leading or trailing spaces
            return temp;
        }
}

public class Runner {
        public static void main(String[] args) {
            TreasureChest tc = new TreasureChest();
            print( tc );

            Jewel j1 = new Jewel( 300, "red" );
            Gold g1 = new Gold( 250.5, 4 );
            tc.add( j1 );
            tc.add( g1 );
            print( tc );

            Jewel j2 = new Jewel( 200, "red" );
            tc.add( j2 );
            print( tc );

            tc.empty();
            Jewel j3 = new Jewel( 300, "red" );
            tc.add( j1 );
            tc.add( j2 );
            tc.add( j3 );
            print( tc );

            tc.empty();
```

```
            Gold g2 = new Gold( 100.1, 5 );
            Gold g3 = new Gold( 100.1, 5 );
            Gold g4 = new Gold( 100.1, 3 );
            tc.add( g1 );
            tc.add( g2 );
            tc.add( g3 );
            tc.add( g4 );
            print( tc );
        }

        public static void print( TreasureChest x ){
            double v = x.getValue();
            System.out.println( "Total value: " + v );
            int w = x.getGoldWeight();
            System.out.println( "Total weight of gold: " + w );
            String s = x.getJewelColors();
            if ( s.length() == 0 )
                System.out.println( "No jewels" );
            else
                System.out.println( "Jewel colors: " + s );
            System.out.println( "********" );     }
}
```

When you run the main class in the Runner class, you should get the following output:

```
Total value: 0.0
Total weight of gold: 0
No jewels
********
Total value: 550.5
Total weight of gold: 4
Jewel colors: red
********
Total value: 750.5
Total weight of gold: 4
Jewel colors: red  red
********
This equals v1 and cannot be added
Total value: 500.0
Total weight of gold: 0
Jewel colors: red  red
********
This equals v2 and cannot be added
Total value: 450.70000000000005
Total weight of gold: 12
No jewels
********
```