Examine the four classes below. Following the classes, there is some code that uses these classes.

```
public class A {
  private int num = 0;

  public int silly() {
    num = num + 1;
    return num;
  }
}

public class B1 extends A {
  private int bNum = 0;

  public int mystery() {
    bNum = this.silly();
    bNum = bNum + 1;
    return bNum;
  }

  public int dupMethod() {
    return 65;
  }
}
```

```
public class B2 extends A {
  private int bNum = 0;

  public int silly() {
    return super.silly() + 1;
  }

  public int dupMethod() {
    return 73;
  }

  public int bizarre() {
    return this.silly() - 10;
  }
}

public class C extends B1 {

  public int mystery() {
    int temp = super.mystery();
    return temp + 10;
  }

  public int silly() {
    return 55;
  }
}
```

For every line in the following code, state what the line of code will output, or state that the line of
code will cause a compile error.

```java
public class Run {
  public static void main(String[] args) {
    A a = new A();
    A aa = new B1();
    A aaa = new B2();
    A aaaa = new C();

    B1 b1 = new A();
    B1 bb1 = new B1();
    B1 bbb1 = new B2();
    B1 bbbb1 = new C();

    B2 b2 = new A();
    B2 bb2 = new B1();
    B2 bbb2 = new B2();
    B2 bbbb2 = new C();

    C c = new A();
    C cc = new B1();
    C ccc = new B2();
    C cccc = new C();

    C ccccc = (C) new A();
    B1 bbbbb1 = (B1) new B2();
    A aaaaa = (C) new C();

    A newA = new A();
    System.out.println(newA.silly());
    System.out.println(newA.mystery());
    System.out.println(newA.dupMethod());
    System.out.println(newA.bizarre());

    newA = new B1();
    System.out.println(newA.silly());
    System.out.println(newA.mystery());
    System.out.println(newA.dupMethod());
    System.out.println(newA.bizarre());

    B1 newB1 = new B1();
    System.out.println(newB1.silly());
    System.out.println(newB1.mystery());
    System.out.println(newB1.dupMethod());
    System.out.println(newB1.bizarre());

    B2 newB2 = new B2();
    System.out.println(newB2.silly());
    System.out.println(newB2.mystery());
    System.out.println(newB2.dupMethod());
```

```
      System.out.println(newB2.bizarre());

      C newC = new C();
      System.out.println(newC.silly());
      System.out.println(newC.mystery());
      System.out.println(newB2.dupMethod());
      System.out.println(newC.bizarre());

   }
}
```

Examine the class below. Between the methods mystery and silly, there is at least one line of code that will not run. Cross out all lines in these methods that will not compile.
In the main method, for every line of code, state what the line of code will output, or state that the line of code will cause a compile error.

```
public class D {

   private int num = 0;
   private static int sNum = 0;

   public static void mystery() {
      this.num++;
      sNum++;
   }

   public void silly() {
      this.num++;
      sNum++;
   }

   public String toString() {
      return "num:   " + this.num + "        sNum:   " + sNum;
   }

   public static void main(String[] args) {
      D d1 = new D();
      D d2 = new D();
      D.silly();
      D.mystery();
      d1.silly();
      d2.silly();
      System.out.println(d1.toString());
      System.out.println(d2.toString());
   }
}
```

A "magic square" is a square of positive integers, in which no number is repeated, and in which the numbers along each of the rows, columns and diagonals add up to the same value.

Here are two examples of magic squares:

```
8 1 6
3 5 7
4 9 2

17 24  1  8 15
23  5  7 14 16
 4  6 13 20 22
10 12 19 21  3
11 18 25  2  9
```

In this question you will complete helper methods for the static method isMagic in the class MagicSquare to return a boolean value indicating if its parameter is a magic square.

```
public class MagicSquare
{
  public static boolean isMagic(int[][] a) {
    int sum = sumOneRow(a);
    return isSquare(a) && allPositive(a) && rowSumsOK(a, sum) &&
              colSumsOK(a, sum) && diagSumsOK(a, sum) &&
              allUnique(a);
  }
}
```

Write the helper methods required by the isMagic method. You may find it helpful to create additional helper methods.

Create a class `StudentList` that stores a list of students. It must allow a new student to be added only if no student with the same ID number already exists in the list. It must allow existing students to be removed given an ID number. There is no known maximum number of students and you must use a partially-filled array to store the `Student` objects.

The `StudentList` information is stored in a text file. When the program begins, the `StudentList` constructor may be passed the name of a text file, in which case it will create the `StudentList` and the `Student` objects by reading in the text file. The text file is a list of students' information, as follows:

```
<student name>
<student ID number>
<student name>
<student ID number>
(etc.)
```

Add a `StudentList` constructor that takes a file name as a parameter.

Add a save method that will store the current `StudentList` in a text file with the provided name (and in the format above). If the method cannot save to the specified file because of an exception, it should return a value of false; otherwise the method should return true to indicate it saved properly.

Redo everything above using binary files instead of text files.

```
public class Student
{
  private String name;
  private int id;

  public Student(String name, int id){
      this.name = name;
      this.id = id;
  }

  public String getName() { return this.name;}

  public int getIDNumber() { return this.id;}
}
```
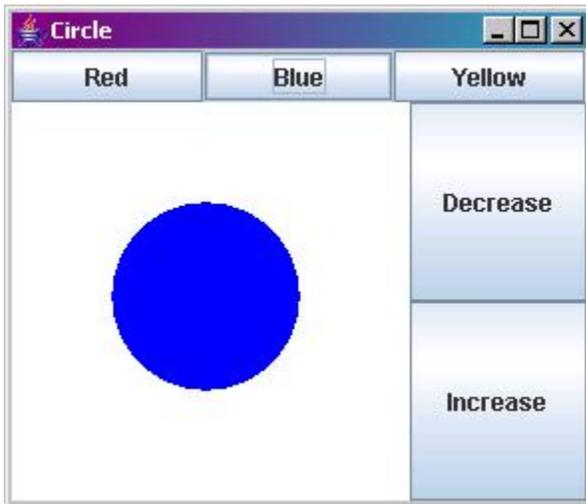
Question: GUIs

Create a set of classes that will display the GUI below. The "Red", "Blue", and "Yellow" buttons should change the colour of the circle to the specified colour, and the "Decrease" and "Increase" buttons should change the diameter of the circle by 5 pixels. All changes should appear immediately after the button is pressed.

There should be one class for the model, and there should be three classes that extend JPanel. You should use a BorderLayout when adding the components to the JFrame.

Question: Inheritance & GUIs

Using the following class descriptions, answer the questions which follow.

Note: Code is only shown for the **toString()** methods. All others have been implemented as described in the comments, but not shown.

```
public class Vehicle
{   private String make   = "";
    private String model  = "";
    private int    year   = 0;
    private String colour = "";
    private String licence = "";
    private String owner  = "";
    private double cost   = 0.0;

1   public Vehicle(String make,
        String model, int year,
        String colour) {...}

    // Sets owner and licence.
2   public void buy(String owner,
        String licence) {...}

    // Sets the dealer's cost of the vehicle.
3   public void setDealerCost(double cost)
    {...}

    // Returns cost of the vehicle + 10% markup.
4   public double getStickerPrice() {...}

    // Used to retrieve information
    // about the Vehicle.
5   public String toString()
    { return make + model + year +
        colour + licence + owner +
        cost;
    }

    // Returns owner of the vehicle
6   public String getOwner() {...}
}
```

```
public class Jeep extends Vehicle
{   private boolean fourWheelDriveExists;
    private boolean winchExists;
    private boolean topOn = true;

7   public Jeep(String make, String
        model, int year, String colour,
        boolean hasFourWheelDrive,
        boolean hasWinch)
    {...}

    // Defaults to no winch and no 4WD.
8   public Jeep(String make, String
        model, int year, String colour)
    {...}

    // Returns cost + 10% markup + $1000 luxury tax.
9   public double getStickerPrice() {...}

    // If top is on, remove; else put on.
10      public void swapTop() {...}

    // Used to retrieve information
    // about the Jeep.
11   public String toString()
    { return "Jeep: " + super.toString() +
        fourWheelDriveExists + winchExists + topOn;
    }

    // Returns TRUE if top is on
12   public boolean getTopOn()
    {...}

}
```

a) Identify which of **Vehicle** and **Jeep** is the *superclass* and the *subclass*.

b) In which method(s) does *overloading* takes place?	
c) In which method(s) does *overriding* takes place?	

d) Write the code for the following constructor:

public Jeep(String make, String model, int year, String colour, boolean hasFourWheelDrive, boolean hasWinch)

e) Write the output for each of the following code fragments. Assume that each code fragment is completely independent of any other and that all code fragments are in a class different from **Jeep** and **Vehicle**.

Jeep jeep = new Jeep("Jeep", "Cherokee", 1991, "red");
jeep.setDealerCost(25000);
jeep.buy("Fred", "RAZZ");
System.out.println(jeep.getStickerPrice());	

Vehicle j = new Jeep("Jeep", "Cherokee", 1991, "red"); System.out.println(j.toString());	

f) Describe the error message caused by each of the following code fragments. Assume that each code fragment is completely independent of any other and that all code fragments are in a class different from **Jeep** and **Vehicle**.

Jeep jeep = new Jeep(); System.out.println(jeep.toString());	
Vehicle jalopy = new Vehicle("Ford","Taurus", 1985, "Teal"); System.out.println("" + jalopy.cost);	

g) State whether the following code fragments are valid or invalid.

Jeep j = new Vehicle("Toyota", "Prius", "2005", "Beige);	
Vehicle v = new Jeep("Ford", "Explorer", "2004", "Red", "true", "false");	
Jeep j = (Jeep) (new Vehicle("Toyota", "Prius", "2005", "Beige));	

h) Fill in the code for the **Jeep's getStickerPrice()** method.

   public double **getStickerPrice**()

i) Write a method **getInventoryValue(...)** in the **VehicleCollection** class that, some of which may be of the class Jeep (recall Jeeps have a luxury tax), returns the total sticker price of all the vehicles.

   public double **getInventoryValue**()

```java
public class VehicleCollection
{
  private Vehicle[] vehicles;
  private int numVehicles = 0;


  //the constructor
  public VehicleCollection(int size)



  //Please complete this method
  public double getInventoryValue()


  //adds a vehicle in the next available spot of the array
  public void add(Vehicle theVehicle)


  //returns true if there is a vehicle with that owner's name
  public boolean contains(String owner)


  //removes the vehicle of the specified owner
  //and replaces it with the vehicle at the end of the array
  public void remove(String owner)


  //returns the information of the given vehicle
  public String getCarInfo(String owner)


  //returns the information of the given owner
  public String getOwnerInfo(String owner)


  //finds and returns the index of the specified vehicle
  private int find(String owner)

}
```

Question: Graphical User Interfaces

The picture at right shows a Graphical User Interface for the Vehicle Collection. A partial implementation of the GUI components is shown below.

Suppose you want to include a **remove** button in the VehicleGUI. The **remove** button would function as follows: when you want to remove a vehicle, you'd type the name of the owner to be removed in the **ownerTF** text field and click on the **remove** button.

The following code contains declaration of other GUI components; you should assume that all code related to all the other GUI components shown in the above picture has already been written; you need only add the code related to the **remove** button.

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

public class VehicleGUI extends JFrame implements ActionListener
{ private VehicleCollection myvehicles;
  private JLabel title = new JLabel("Owner name");
  private JTextField ownerTF = new JTextField(30);
  private JTextArea info = new JTextArea(10,30);
  private JButton listOwner = new JButton("List Owner");
  private JButton carInfo = new JButton ("List Car");

  public VehicleGUI(VehicleCollection thevehicles)
  { Container pane = this.getContentPane();
    pane.setLayout(new FlowLayout());
    pane.add(title); pane.add(ownerTF); pane.add(listOwner); pane.add(carInfo); pane.add(info);
```

Some methods available in the `JButton` class:
```java
 public JButton(String text)
 public void addActionListener(ActionListener al)
```