1. A Fraction can be implemented as a class with two instance variables: numerator and denominator, like this:

```java
public class Fraction {
    private int numerator;
    private int denominator;
    public Fraction (int num, int denom) {
        numerator = num;
        denominator = denom;
    }
}
```

a. Modify the Fraction constructor so that it throws an exception named DivideByZero if the denominator is 0. Below is code for the constructor that works properly as long as the denominator is not 0. Change it to throw an exception when the denominator is 0.

b. Complete the method below. It takes 2 int parameters and creates a Fraction using those values. If the Fraction constructor throws DivideByZero, your function should handle the exception by displaying “Infinity” using System.out.println.

```java
public void seekInfinity (int n, int d) {

    Fraction f = new Fraction (n, d);

}
```
2. Indicate what output the following program produces. You can assume that the program compiles without any errors.

```java
public class Food {
    public String getTaste() {
        return "like chicken";
    }

    public String getGoodness() {
        return "Yum!";
    }
}

public class Lemon extends Food {
    public String getTaste() {
        return "sour";
    }

    public String getGoodness() {
        return "Pucker";
    }
}

public class Rat extends Food {
    public String getGoodness() {
        return "Yuck";
    }
}

public static void main(String[] args) {
    Rat r = new Rat();
    System.out.println(r.getGoodness() + " Tastes " + r.getTaste());

    Food f = new Lemon();
    System.out.println(f.getGoodness() + " Tastes " + f.getTaste());
}
```
3. HTML is the language that is used to define Web pages. In HTML, you mark the portions of a Web page that you want specially formatted with tags. For example, the HTML:

Spring break is almost <b>here</b>!

would display the word “here” in boldface, like this:

Spring break is almost here!

If you wanted to display the text without applying the formatting commands, you could just remove all of the HTML tags. In this case the result would be:

Spring break is almost here!

Fortunately, this is straightforward because tags always start with the character “<” and end with the character “>”. The code below looks for all HTML tags and removes them. The one tag it treats specially is “\<br\>” which causes HTML formatting to insert a newline. When the code below finds that tag, it replaces it with a “\n” so that a newline appears in the unformatted text.

Unfortunately, the function below does not quite work. Find and fix the error. You may assume that this function compiles without error. You can also ignore the fact that HTML is case-insensitive.

Here is a reminder of the behavior of the String methods used below:

- **int indexOf (String str, int pos)** - This searches for the first occurrence of str beginning after position pos and returns the index. If str is not found, it returns -1.

- **String substring (int pos)** - Returns the substring beginning at index pos and going to the end of the string.

- **String substring (int begin, int end)** - Returns the substring beginning at index begin and ending with the character immediately before index end.
/* Remove HTML tags from text. Replace <br> tags with a newline character. 
message - the original text. 
Preconditions: message should not be NULL. message should use correct HTML syntax. */
String removeHTMLTags (String message) {
    // Find the first tag
    int tagPos = message.indexOf( "<", 0 );

    while ( tagPos != -1 ) {
        // Remove an HTML tag from the message
        String before = message.substring( 0, tagPos );

        int tagEnd = message.indexOf( ">", tagPos );

        String after = message.substring( tagEnd + 1 );

        if ( message.substring( tagPos, tagEnd + 1 ).equals("<br>")){
            message = before + "\n" + after;
        } else {
            message = before + after;
        }
    }

    return message;
}
4. Answer the questions on the next page about the class below.

```java
/*
* Created on Feb 8, 2004
* A simple program to draw long straight lines
*/
public class RubberBand extends WindowController {
  // The last line drawn
  private Line currentLine;

  // The point where the user depressed the mouse
  private Location start;

  // Remember where the user pressed the mouse and draw the first
  // line.
  public void onMousePress( Location point )
  {
    start = point;
    currentLine = new Line( point, point, canvas);
  }

  // Erase the last line drawn and draw a new line to the current
  // mouse location. This gives the user the illusion of the first
  // line changing length and direction as the mouse moves.
  public void onMouseDrag( Location point )
  {
    currentLine.removeFromCanvas();
    currentLine = new Line( start, point, canvas);
  }
}
```

a. How many **method declarations** are there? What are their **signatures**?

b. What are the **names** of all the **methods called**?

c. How many **instance variables** are declared? What are their **names**?

d. How many **local variables** are declared? What are their **names**?

e. How many **parameters** are declared? What are their **names**
5. In this question, I show you the signatures of several methods and constructors that are defined in the FilledRect class. I would like you to show me how you would call those methods and constructors. If the method or constructor returns a value, I would like you to put the call on the right hand side of an assignment statement so that the returned result is saved in a variable. For the purposes of this question, you can assume that the following variables have been declared:

`FilledRect rect`  
`FilledRect rect2`  
`DrawingCanvas myCanvas`  
`Color c`  
`Location loc`

a. public `FilledRect(double x, double y, double width, double height, DrawingCanvas canvas)`

b. public `void move (double dx, double dy)`

c. public `Color getColor()`

d. public `void moveTo (Location point)`

e. public `boolean overlaps (FilledRect otherRect)`
6. Consider the following interface and class declarations. For each part of the question that follows, indicate if the statement is valid or not. Consider a statement to be valid if it would compile and execute without throwing an exception, given just the information shown below. If it is not valid, briefly explain why not. You can assume that the interfaces and classes shown all compile and work correctly. … means that part of the definition is not shown because it is not relevant to the question.

```java
public interface Shape {
    ...
}

public class Polygon implements Shape {
    /** Creates a new polygon. */
    public Polygon() {...}

    ...
}

public class Rectangle implements Shape {
    /** Creates a new Rectangle with the given width and height. */
    public Rectangle (int width, int height) {...}

    ...
}

a. Shape s = new Shape();

b. Shape s = new Rectangle(3, 4);

c. Polygon p = new Rectangle (3, 4);

d. Polygon p = new Polygon();

e. Polygon p = new Shape();
```
7. The following program is incorrect. When the user runs the program, both buttons are displayed as expected. However, no matter which button the user clicks on, the message: "I said don't click me!" is displayed. Show how to change the code so that it displays “Clicked” when the user clicks on the button labeled “Click me” and displays "I said don't click me!” when the user clicks on the button labeled “Don't click me”.

```java
public class Bug extends JPanel implements ActionListener {
    private JButton clickMe;
    private JButton dontClickMe;

    public Bug () {
        JButton clickMe = new JButton ("Click me");
        clickMe.addActionListener(this);
        add (clickMe);

        JButton dontClickMe = new JButton ("Don't click me");
        dontClickMe.addActionListener(this);
        add (dontClickMe);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == clickMe) {
            System.out.println("Clicked");
        }
        else {
            System.out.println("I said don't click me!");
        }
    }

    public static void main (String[] args) {
        JFrame f = new JFrame();
        f.setSize(new Dimension(200, 200));
        Bug bug = new Bug();
        Container contentPane = f.getContentPane();
        contentPane.add(bug, BorderLayout.CENTER);
        f.setVisible(true);
    }
}
```
8. Consider the following two (partial) class definitions: Car and Airplane. Design a new class Vehicle to be a superclass of Car and Airplane. Show what you would put in the superclass and how you would change the subclasses. These classes are very incomplete. You are not expected to complete them, just to show how you would use a superclass to capture the information already in these subclasses.

```java
public class Car {
    private static final int TOP_SPEED = 95;
    private int speed;
    private String licensePlateNumber;
    private int passengers;
}

private class Airplane {
    private static final int TOP_SPEED = 500;
    private int speed;
    private String airline;
    private int passengers;
}
```