Sketch the GUI generated by the following code:

```
public static void quiz2() {
    JFrame frame = new JFrame("Quiz 2");
    frame.setSize(300,500);

    Container pane = frame.getContentPane();

    BorderLayout layout = new BorderLayout();
    pane.setLayout(layout);

    JButton btnQuit = new JButton("Quit");
    JButton btnHelp = new JButton("Help");
    JButton btnAdd = new JButton("Add");
    JButton btnSave = new JButton("Save");
    JButton btnDelete = new JButton("Delete");
    pane.add(btnQuit, BorderLayout.NORTH);
    pane.add(btnHelp, BorderLayout.SOUTH);
    pane.add(btnAdd, BorderLayout.EAST);
    pane.add(btnSave, BorderLayout.WEST);
    pane.add(btnDelete, BorderLayout.CENTER);

    frame.validate();
    frame.setVisible(true);
}
```

Concisely and precisely describe each of the following and explain how they differ:

FlowLayout

BorderLayout

GridLayout

(a) What does it mean to **implement the ActionListener interface**?

(b) Once an object from a class that implements the ActionListener interface has been instantiated, how is it typically used?

Consider the following class. Comments appear where code is missing. Fill in the correct Java code to accomplish the task listed in the comments.

```
public class GUI {

  private JButton btnQuit;
  private JButton btnHelp;
  private ButtonEventHandler btnEventHandler;

  private class ButtonEventHandler implements ActionListener {
    // Declaration of the method that is called when an event has occurred



    {
      if(event.getSource() instanceof JButton) {
        // Check to see if the event was triggered by the btnQuit object



        {
          System.out.println("You pushed my QUIT button");
        }
      }
    }
  }

  public static void main(String[] args) {
    GUI myGUI = new GUI();
  }

  public GUI() {
    btnEventHandler = new ButtonEventHandler();

    JFrame frame = new JFrame("Event handing sample");
    frame.setSize(200, 200);
    frame.setLocation(10, 10);
    frame.setResizable(false);

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    frame.setVisible(true);

    btnQuit = new JButton("Quit");
    // Register the btnQuit object with the btnEventHandler   **********



    btnQuit.setBounds(10, 125, 80, 30);

    frame.add(btnQuit);
    btnHelp = new JButton("Help");
    btnHelp.setBounds(100, 125, 80, 30);
    frame.add(btnHelp);

  }
}
```

What will be displayed when the following code is executed?

```
try {
  Double.parseDouble("quiz 4");
  System.out.println("1");
} catch (ArrayIndexOutOfBoundsException e) {
  System.out.println("2");
} catch (NumberFormatException e) {
  System.out.println("3");
} catch (RuntimeException e) {
  System.out.println("4");
} finally {
  System.out.println("5");
}
  System.out.println("6");
```

What will be displayed when the following code is executed?

```
try {
  Double.parseDouble("quiz 4");
  System.out.println("1");
} catch (ArrayIndexOutOfBoundsException e) {
  System.out.println("2");
} catch (NumberFormatException e) {
  System.out.println("3");
} catch (RuntimeException e) {
  System.out.println("4");
} finally {
  System.out.println("5");
}
  System.out.println("6");
```

Show how to create a DataInputStream object that will read from a file called "quiz6.dat" located in the program's project folder. Show, with code, how to read data from the file into your program. You can assume that the data is already in the file and in any order you wish. Be sure to explain what your sample code does.

For each class, list at least on method for writing data to a file and explain how it works/what it does.

DataOutputStream

FileOutputStream

PrintWriter

ObjectOutputStream

Complete the following method that will prompt the user to enter as many non-negative integers as they want followed by one negative integer. The method should return a list of all of the non-negative integers entered by the user. You may choose the method of input (console or GUI).

```
public List<Integer> getNonNegativeIntegers() {



}
```

Write a method called averageLength which accepts a list of strings (in either an ArrayList or a LinkedList) and returns the average length of all the strings in the list. For full credit, your method should work with both ArrayLists and LinkedLists.

Create two classes.
The first class is an abstract base class called Robot with two methods: getColor() and walk(). Both methods should return `void` and accept no arguments. The walk() method should be an abstract method.

The second class, called Transformer, inherits from Robot and should implement any methods necessary to make it possible to create objects from the Transformer class.

Create two classes.
The first class is an abstract base class called PartA with two methods: first() and second(). Both methods should return `void` and accept no arguments. The second() method should be an abstract method.

The second class, called PartB, inherits from PartA and should implement any methods necessary to make it possible to create objects from the PartB class.