[**Student may use one single-sided** 8.5×11 **inch sheet with handwritten reference material.**] Show all of your work clearly in the space provided or on the additional page at the end of the exam. Be sure to **read each problem carefully**. Note that the exam is double sided. Due to time constraints, you are not required to document your source code.

1. (20 points) Circle TRUE or FALSE

TRUE | FALSE A javafx.scene.control.Button is an event source.

TRUE | FALSE The Scene Builder produces an FXML file and associated controller class.

TRUE | FALSE The DataOutputStream class can be used to read Java primitive values out of a file in binary format.

TRUE | FALSE An Alert object can be used to create a dialog that gets text input from the user.

TRUE | FALSE A FlowPane can be used to graphically display components (e.g., labels, buttons) left to right.

TRUE | FALSE The addAll() method can be used to add components to an HBox.

TRUE | FALSE RuntimeExceptions must be handled in order for your code to compile.

TRUE | FALSE <Label fx:id="label" /> in an FXML file specifies that the controller class will have the following attribute: Label label;.

TRUE | FALSE A `try` block must be immediately followed by either a `catch` or `finally` block.

TRUE | FALSE An Error object may be thrown, but you should not attempt to catch an Error object.

2. (5 points) Explain the purpose of the fx:controller attribute in an FXML file.

3. (5 points) Describe two options for handling checked exceptions.

4. (10 points) The ShapeCreatorApp constructor from Lab 7 accepts a Scanner instance that should be assocated with an input file. Complete the code below that creates a ShapeCreatorApp object that is associated with the file specified by the user.

```
public static void main(String[] ignored) {
    String filename = JOptionPane.showInputDialog(null, "Enter a filename");
    Scanner input;

        new ShapeCreatorApp(input);

}
```

5. (15 points) Consider the following code:

```
public static void main(String[] ignored) {
    Scanner in = new Scanner(System.in);
    try {
        one(in.next(), in.next());
    } catch (NumberFormatException e) {
        System.out.println("Must enter int");
    } catch (IndexOutOfBoundsException e) {
        System.out.println("Need longer word");
    }
    System.out.println("Buy bye");
    in.close();
}

private static void one(String word, String number) {
    int times = Integer.parseInt(number);
    for(int i=0; i<times; ++i) {
        two(word.substring(i));
    }
}

private static void two(String substring) {
    System.out.println(substring + ": " + substring.length());
}
```

Determine the output if the user:

(a) types This 2 and then **Enter**

(b) types 3 and then **Enter** and 33 and then **Enter**

(c) types turn page soon and then **Enter**

6. (15 points) Complete the program below that reads an integer and a double from two files (that were written to by the solution to Quiz 7) from two files – quiz7.bin (binary format) and quiz7.txt (text format) and displays whether or not the values in the two files match.

```
public static void main(String[] ignored) {
    int binI;
    double binD;
    int txtI;
    double txtD;

    if(binI==txtI && binD==txtD) {
        System.out.println("Both files contain the same values");
    } else {
        System.out.println("The files contain different values");
    }
}
```

7. (10 points) Sketch what the GUI described in the following FXML file looks like:

```
<VBox xmlns="http://javafx.com/javafx/null" xmlns:fx="http://javafx.com/fxml/1"
      fx:controller="Exam">
   <children>
    <Label fx:id="textPrompt" text="Text:" />
    <TextField fx:id="textInput" onAction="#textChanged"
               promptText="Type line of text" />
    <Label text="History:" />
    <TextArea fx:id="historyOutput" editable="false"
              prefHeight="200.0" prefWidth="200.0" />
    <Button onAction="#clear" text="Clear" />
  </children>
</VBox>
```

8. (20 points) Implement the controller class that can be used with the FXML file in the previous problem. The controller should enable the following behavior:

- When the user hits **Enter** while in the text field, the text in the text field is added as the first line of text in the text area (all other lines of text shift down by one)
- When the button labelled **Clear** is clicked, all text in the text area is removed