[**Closed book and notes.**] 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. (10 points) What are two ways to determine the source of an event while in the method called when an event occurs?

2. (10 points) Explain the role of the `throws` keyword. When is its use required?

3. (5 points) What is the difference between a JFrame and a JPanel?

4. (10 points) What type of value does the `instanceof` operator return? Give an example of when to use this operator.

5. (10 points) When is it beneficial to use an anonymous inner class?

6. (10 points) Under what circumstances would the `finally` block in the following code **NOT** execute?

```
public static void main(String[] args) {
  Scanner in = new Scanner(System.in);
  try {
    int i = in.nextInt();
    double x = -1.0;
    x = in.nextInt();
    System.out.println(i*x);
  } catch(RuntimeException e) {
    System.out.println("ouch");
  } finally {
    System.out.println("done");
  }
}
```

7. (15 points) Draw a picture of what the GUI associated with the following program would look like. (Draw your answer to the right of the code.)

```
1  public class ExamII extends JFrame {
     private JTextField a;
3    private JTextField b;
     private JTextField operator;
5    private JTextField result;

7    public static void main(String[] args) {
       new ExamII();
9    }

11   public ExamII() {
       this.setTitle("Exam II");
13     this.setSize(300, 200);
       this.setDefaultCloseOperation(EXIT_ON_CLOSE);
15     this.setLayout(new BorderLayout());
       JPanel panel = new JPanel();
17     panel.setLayout(new GridLayout(1, 3));
       a = new JTextField(5);
19     b = new JTextField(5);
       result = new JTextField(20);
21     result.setEditable(false);
       operator = new JTextField("+");
23     JButton equals = new JButton("=");
//       equals.addActionListener(new CalcHandler());
25     panel.add(a);
       panel.add(operator);
27     panel.add(b);
       add(panel, BorderLayout.NORTH);
29     add(equals);
       add(result, BorderLayout.SOUTH);
31     this.setVisible(true);
     }
33
     private static double calculate(String a, String b, String op) {
35     double first = Double.parseDouble(a);
       double second = Double.parseDouble(b);
37     char operand = op.charAt(0);
       double result = 0;
39     if(operand=='+') {
         result = first + second;
41     } else if(operand=='-') {
         result = first - second;
43     } else if(operand=='/') {
         result = first / second;
45     } else if(operand=='*') {
         result = first * second;
47     } else {
         throw new IllegalOperatorException("Not a valid operator");
49     }
     }
51 }
```

8.

(a) (20 points) Suppose line 24 of the code in the previous problem is uncommented. Implement CalcHandler as an inner class so that when the "=" button is pressed, the result returned from the calculate() method is displayed in the result text field. If the calculate() method throws an exception, the result field must be replaced with:

- "Not a valid operator" — if an the operator text field contains something other than the four basic math operations, or
- "Invalid input" — for anything else.

You may assume that the parent class of IllegalOperatorException is IllegalArgumentException. In addition, a `finally` block must be used to ensure that "calc" is displayed to the console each time the button is pressed.

(b) (10 points) In the above, CalcHandler was implemented as an inner class. Could it have been implemented as a stand-alone class? Justify your answer.