CSCI 201L Midterm – Written
Fall 2015
10% of course grade

1. **Inheritance** – Answer the following questions about inheritance.
   a. Does Java allow overloading, overriding, and redefining of methods? **(0.5%)**

   b. Explain what is meant by **overloading**. Provide a code snippet as an example. **(0.5%)**

   c. Explain what is meant by **overriding**. Provide a code snippet as an example. **(0.5%)**

   d. Explain what is meant by **redefining**. Provide a code snippet as an example. **(0.5%)**

2. **Exception Handling** – Java, unlike C++, has both checked and unchecked exceptions. Explain why unchecked exceptions in Java do not need to be handled by a corresponding `try` block. **(0.5%)**

3. **User Interface Design –** Explain two differences between JDialog and JFrame. **(0.5%+0.5%)**

4. Generics – Answer the following questions based on the generic code below.

```
1   public class Problem4 {
2     public static<T, T1 extends T> void meth(T1 [] list) {
3           T num = list[0];
4           System.out.println(num);
5     }
6
7     public static void main(String[] args) {
8           Integer[] integers = {1, 2, 3, 4, 5};
9           String[] strings = {"London", "Paris", "New York"};
10
11          Problem4.<Number, Integer>meth(integers);
12          Problem4.<Object, Object>meth(strings);
13    }
14  }
```

a. Does the code throw a compile-time error, have a warning, throw a runtime error, or compile and run? **(0.5%)**

Answer either question (b) or (c) based on your answer to (a) above. If you answer both questions (b) and (c), you will not get credit for either one.

b. If the code throws a compile-time error, has a warning, or throws a runtime error, explain the problem and provide a solution to it in the above code. **(0.5%)**

c. If the code compiles and runs, explain what is happening on lines 11, 12, 2, and 3. **(0.5%)**

5. **Layout Managers –** For each layout manager below, explain its functionality. Include whether the preferred height and width of a component is acknowledged, how components are laid out when added, and what happens to the components when the frame is resized. Draw (no code needed) a typical GUI that uses each layout manager. **(0.5% + 0.5% + 0.5%)**
    a. **BoxLayout (Y_AXIS)**

    b. **GridLayout**

    c. **GroupLayout**

6. **Inner Classes** – Your friend from UCLA sees some of your GUI code and tells you that it won't compile because you have instantiated an abstract class. You respond by saying that this is an anonymous inner class. Explain to your less-educated UCLA friend what is actually happening with an anonymous inner class. Use the following code as an example. **(1.0%)**

```
1  public class Problem6 {
2    public Problem6(B b) {
3      b.b();
4    }
5    public static void main(String [] args) {
6      new Problem6(new B() {
7        public void b() {
8          System.out.println("b");
9        }
10     });
11   }
12 }
13 abstract class B {
14   abstract void b();
15 }
```

7. **Option Panes and Dialogs –** Give two differences and two similarities between option panes and dialog boxes in Java. **(0.25% + 0.25% + 0.25% + 0.25%)**

Similarity #1 –

Similarity #2 –

Difference #1 –

Difference #2 –

8. Graphics – Draw the GUI rendered by the following code. **(1.0%)**

```
1   import java.awt.Graphics;
2   import javax.swing.JButton;
3   import javax.swing.JFrame;
4
5   public class Problem8 extends JFrame {
6      public static final long serialVersionUID = 1;
7      public Problem8() {
8        super("Problem 8");
9        setSize(200, 200);
10       add(new MyButton());
11       setVisible(true);
12     }
13     public static void main(String [] args) {
14       new Problem8();
15     }
16  }
17  class MyButton extends JButton {
18     public static final long serialVersionUID = 1;
19     protected void paintComponent(Graphics g) {
20       super.paintComponent(g);
21       g.drawString("hi", 5, 15);
22       g.drawLine(0, getHeight(), getWidth(), 0);
23       g.fillArc(0, 0, getWidth(), getHeight(), -180, 45);
24     }
25  }
```

9. Tables and Trees – Draw the GUI that is generated by the following code. **(1.0%)**

```
1   import java.awt.FlowLayout;
2   import javax.swing.JFrame;
3   import javax.swing.JScrollPane;
4   import javax.swing.JTable;
5   import javax.swing.JTree;
6   import javax.swing.table.DefaultTableModel;
7   public class Problem9 extends JFrame {
8      public static final long serialVersionUID = 1;
9      public Problem9() {
10       super("Problem 9");
11       setSize(300, 300);
12       setLayout(new FlowLayout());
13       Object [][] data = new Object[][] {
14         {1, "CSCI", 103},
15         {2, "CSCI", 104},
16         {3, "EE", 101}
17       };
18       Object [] names = new Object[] {"#", "Prefix", "Number"};
19       DefaultTableModel model = new DefaultTableModel(data, names);
20       model.moveRow(2, 2, 0);
21       model.addColumn("Grade");
22       model.setValueAt("B", 0, 3);
23       model.setValueAt("A", 1, 1);
24       JTable jt = new JTable(model);
25       JScrollPane jsp = new JScrollPane(jt);
26       add(jsp);
27       String [] classes = {"103", "104", "201", "170", "270"};
28       JTree tree = new JTree(classes);
29       tree.setRootVisible(true);
30       JScrollPane jsp1 = new JScrollPane(tree);
31       add(jsp1);
32       setVisible(true);
33     }
34     public static void main(String [] args) {
35       new Problem9();
36     }
37  }
```