CSCI 201L Midterm – Written SOLUTION
Summer 2016
10% of course grade

1. **Abstract Classes and Interfaces** – Give two differences between an interface and an abstract class in which all of the methods are abstract. **(0.5% + 0.5%)**

Interfaces are Java's solution to only having single inheritance, though you can inherit from more than one object through interfaces.

Abstract classes still inherit from the Object class (or possibly other classes), and all of the methods in those classes are not abstract.

Abstract classes can have variables that are not static and final.

Other answers could be acceptable.

2. **Serialization** – If an object does not implement the `Serializable` interface, a `NotSerializableException` will be thrown when trying to serialize that object. If an object *does* implement the `Serializable` interface, there are still two types of variables in the class that will not be serialized. State the two types of variables that will not be serialized and explain why they are not serialized. **(0.5% + 0.5%)**

Transient variables are specifically defined so that they will not be serialized.

Static variables are not serialized because classes are not serialized but rather object are serialized. Static variables are not associated with an object but are associated with a class.

3. **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%)**

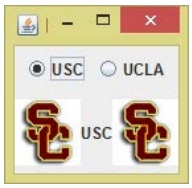
Unchecked exceptions can be handled by good programming practices instead of exception handling.

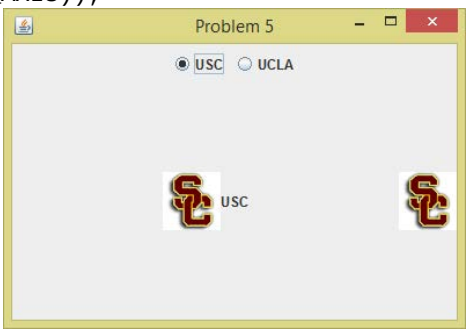
Exception handling is a very expensive operation (takes about 10x as long to execute as good programming practice), so handling unchecked exceptions through exception handling should be avoided.

4. **GUI Programming -** Draw the GUI that is generated by the following code. **(1.0%)**

Assume the image `usc.jpg` is 

Assume the image `ucla.jpg` is 






```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Box;

import javax.swing.BoxLayout;
import javax.swing.ButtonGroup;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;

public class Problem5 extends JFrame {
	private JLabel jl, jl1, jl2;
	private ImageIcon ii, ii1;
	public Problem5() {
		super("Problem 5");
		setSize(200, 300);
		setLocation(500, 300);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JPanel jp = new JPanel();
		jp.setLayout(new BoxLayout(jp, BoxLayout.X_AXIS));
		ii = new ImageIcon("usc.jpg");
		ii1 = new ImageIcon("ucla.jpg");
		jl = new JLabel(ii);
		jp.add(Box.createGlue());
		jp.add(jl);
		jl1 = new JLabel("USC");
		jp.add(jl1);
		jp.add(Box.createGlue());
		jl2 = new JLabel(ii);
		jp.add(jl2);
		add(jp, BorderLayout.CENTER);

		JRadioButton jrb = new JRadioButton("USC", true);
		JRadioButton jrb1 = new JRadioButton("UCLA", false);
		ButtonGroup bg = new ButtonGroup();
		bg.add(jrb);
		bg.add(jrb1);
		JPanel jp1 = new JPanel();
		jp1.add(jrb);
		jp1.add(jrb1);
		add(jp1, BorderLayout.NORTH);
		setVisible(true);
	}
	public static void main(String [] args) {
		new Problem5();
	}
}
```

5. **Inheritance** – There are two errors in the following code. Describe the errors and modify the code to fix them. **(2.0%):**

```
1  interface C {
2     public int m();
3     public abstract int n();
4  }
5
6  abstract class D implements C {
7     public int m() {
8           return 3;
9     }
10 }
11
12 abstract class E extends D {
13    public int n() {
14          return 4;
15    }
16    public int o() {
17          return 5;
18    }
19 }
20
21 class F extends E {
22
23 }
24
25 public class Problem3 {
26    public static void main(String [] args) {
27          F f = new F();
28          f.m();
29          f.n();
30          f.o();
31          E e = new E();
32          e.m();
33          e.n();
34          e.o();
35          C c = new F();
36          c.m();
37          c.n();
38          c.o();
39    }
40 }
```

0.5% E is an abstract class and therefore can't be instantiated.
Fix **(0.5%)**
The line could be changed to:
E e = new F();

0.5% The method o is undefined for class C. The only methods that can be called on an object are those which are defined in the compile-time type of the object.
Fixes **(0.5% for one of the following)**
1. We could add an abstract method o into interface C.
2. We could change line 35 to say E e = new F();

6. **Inner Classes** – `KeyListener` is an interface that contains three methods – `keyPressed`, `keyTyped`, and `keyReleased`, all three of which take a `KeyEvent` as a parameter. Write the code using an anonymous inner class to add a `KeyListener` to a `JTextField` that prints out "Hello" to the command line when a user releases the 'h' key on the keyboard. **(1.5%)**

```
JTextField jtf = new JTextField();
jtf.addKeyListener(new KeyAdapter() {
      public void keyReleased(KeyEvent ke) {
            if (ke.getKeyCode() == KeyEvent.VK_H) {
                  System.out.println("hello");
            }
      }
});
```

7. **Garbage Collection** – The garbage collector provides one of the biggest advantages to Java over C++, though many Java developers inaccurately believe that a Java program cannot crash due to running out of memory. Write a snippet of code that would cause a Java program to throw an `OutOfMemoryError`. **(1.0%)**

```
String str = "";
while(true) {
  str += "a";
}
```

Other answers may be acceptable.

8. **Software Engineering –** Scrum is arguably the most popular software engineering methodology in use right now. Give two reasons why you think that is the case. **(0.5% + 0.5%)**

Software engineering has turned toward needing more frequent releases of products, making agile methodologies more useful.

Web and mobile development has continuous updates to keep the users interested, so more frequently release cycles are needed.

Agile methodologies focus on people and customers rather than merely focusing on processes and documentation. This makes programmers happier.

Programming teams with good programmers tend to desire scrum since there is a shared sense of responsibility. Team work over individual work.

Other answers could be acceptable.

9. **Anonymous Inner Classes –** The following code may appear to some people to be instantiating an interface, which we learned is not possible in Java. If the following code does not compile, explain why and fix it. If the following code does compile, explain what is happening on lines 7-14. **(1.0%)**

```
1   public class Problem6 {
2     public Problem6(I i) {
3       i.a();
4       i.b();
5     }
6     public static void main(String [] args) {
7       new Problem6(new I() {
8         public void a(float f) {
9           System.out.println("a");
10        }
11        public void b() {
12          System.out.println("b");
13        }
14      });
15    }
16  }
17  interface I {
18    void a(int i);
19    void b();
20  }
```

0.5% The method a(int) is not overridden in the anonymous inner class, so it will need to be abstract and cannot be instantiated.

Fixes **(0.5% for one of the following)**

1. Change int to be float on line 18, or
2. Change float to be int on line 8