I. Fill in the Blank (30 pts)

1. In order to use an interface, a class definition uses the keyword **implements**.

2. A **wrapperclass** is a special class that is meant to encapsulate a primitive so that it can be used in places that require objects.

3. **Autoboxing** is a special feature that allows primitives to be automatically encapsulated into a special class without us having to create an instance of that class.

4. **Polymorphism** refers to a dynamic binding that takes place at runtime between a method call and implementation.

5. The method **actionPerfomed** is found in the ActionListener interface.

6. A **thread** is a separate line of execution.

7. If a subclass contains a class method with the same signature and return type as one in its immediate superclass, then the method from the superclass is **hidden** in the subclass.

8. **FlowLayout** is a layout manager that arranges components on a container from left to right.

9. If a class is declared **final**, it cannot be extended.

10. The method **itemStateChanged** is found in the ItemListener interface.

11. The **java.lang** package is automatically imported.

12. We use the **getCodeBase()** method to get a reference to the URL where the .class file of an applet is found.

13. The class **MouseListener** implements the methods in the MouseListener interface so that we can extend it and only override the methods we want.
14. We call the components in the java.awt package **heavweight** components because they rely heavily on their underlying platform.

15. The **start()** method is executed each time a user returns to a page containing an applet.

16. **Inheritance** involves creating new classes by extending existing class definitions and adding additional functionality.

17. If A extends B, then A is a **subclass** of B, and B is a **superclass** of A.

18. We use the keyword **extends** to implement inheritance.

19. We use the **play()** method on an AudioClip to play it once.

20. The class **Object** is the only one in the Java language that doesn’t have a superclass.

21. Thread and any class that wants to be able to run implements the **Runnable** interface.

22. The **init()** method is the first method executed when an applet begins executing and only executes once.

23. In order to refer to itself, an object can use the keyword **this**.

24. A class that is **abstract** is incomplete, and we can’t create an instance of it.

25. If we have a subclass of Component and we want to draw a circle on that Component so that it touches each edge of the component, we would use the line `g.drawOval(0,0,getWidth(),getHeight())`.

26. If a variable is declared with the keyword **protected**, then they can be accessed in the same class, same package, and any subclass.

27. If a local variable in an instance method has the same name as an instance variable in the class definition, then the local variable is said to **shadow** the instance variable.

28. A subclass can refer to its immediate superclass with the keyword **super**.
II. Matching (10 pts)

29. valid applet tag **K**
   A. Allows an entity to be accessed anywhere

30. single inheritance **F**
   B. A relationship that allows instances of B to be treated as if they are instances of A if A extends B

31. shutdown hook **G**
   C. A cane which is used to break a computer which won’t work

32. public String toString() **T**
   D. In an event handler returns a reference to the component that caused an event to occur

33. no access control keyword **O**
   E. Changing the implementation of an overridden instance method

34. method overriding **E**
   F. Means that we can only list one class after the keyword extends

35. getSource() **D**
   G. An unstarted thread that will be executed before the virtual machine shuts down

36. getDocumentBase() **Q**
   H. An inheritance that can’t be used by married people

37. autounboxing **S**
   I. A relationship that allows instances of A to be treated as if they are instances of B if A extends B

38. “is a” **I**
   J. Contains only the applet name

   K. Contains the name of the applet, the width, and the height

   L. In medical programs returns the source of a disease

   M. A method that has to be defined in a class before it can be used

   N. Allows an entity to be accessed in the same class, same package, and any subclass

   O. Allows an entity to be accessed in the same class and same package

   P. Allows a primitive to be automatically encapsulated into its corresponding wrapper class

   Q. Returns a reference to the URL where the .html file that loaded an applet is located

   R. Involves more than one method in a class definition with the same name

   S. Automatically extracts a primitive from its wrapper class

   T. A method inherited by all classes from Object which returns the String representation of an object

   U. A method to display the lyrics of the Brady Bunch and Joanie loves Chachi theme songs

   V. Returns a reference to the URL where the .class file for an applet is found
III. Short Answer (20 pts)

39. (2 pts) Does polymorphism apply to both instance methods and class methods? Explain.

No polymorphism only applies to instance methods. The compile-time type of a reference determines which class method is invoked.

40. (2 pts) Which of the following are valid assignments and why?

Object o = new String();
String s = new Object();

The first assignment is valid because a String “is a” Object, we can call a String an Object. However, an Object is not necessarily a String so we can’t call an Object a String.

41. (2 pts) Will the following cause a compile-time or runtime error? Explain.

Object o = new Object();
Thread t = (Thread)o;

There will be no compile-time error since Thread is a subclass of Object. The cast is allowed at compile-time. However, at run-time there is an error since an Object can’t be called a Thread.

42. (2 pts) What can we not do with an abstract class?

We cannot directly create an instance of it.

43. (2 pts) Suppose we create a JFrame with the following code in a main method.

JFrame frame = new JFrame(“Test”);
frame.setSize(500,500);
frame.setLayout(new FlowLayout());
if (3 == 4)
frame.add(new JLabel(“Test”));
else
frame.add(new JLabel(“Joanie”));

If this is the only code in the main method, what will show up on the frame when the program is run? Explain.

Nothing will display. A frame is not visible by default. Since there is no call to setVisible, the frame will not be displayed.
44. (2 pts) What is the initial sequence of execution when an applet is run in its context?

init(), start(), paint(Graphics g)

45. (2 pts) In the call referenceName.methodName(), if methodName is a class method, what determines which version of methodName() will be called?

The compile-time type of the reference

46. (2 pts) In the call referenceName.methodName(), if methodName is an instance method, what determines which version of methodName() will be called?

The runtime type of the object

47. (2 pts) Is the following allowed? Explain.

Double d = 1;

No, this not allowed. 1 will be autoboxed into an Integer, but Integer is not a subclass of Double.

48. (2 pts) What are two example of heavyweight classes in the javax.swing package?

JApplet and JFrame
IV. Discussion (10 pts)

49. (3 pts) What are the requirements to create a truly immutable object?

We make all instance variables private, make sure there are no mutator methods, and make sure there are no accessor methods for instance variables which are mutable.

50. (4 pts) What is the interface contract?

If a class implements an interface, it must give implementations to all of the methods inherited from the interface. In exchange, the class has the “is a” relationship with the interface.

51. (3 pts) What are two ways of creating a thread?

Creating a subclass of Thread and overriding the run method or creating a class that implements the Runnable interface and creating a Thread using an instance of the class.
V. Problem Solving and Coding (30 pts)

52. (5 pts) What is printed when the class Test is executed? Explain the output.

```java
public class Question52 {
    public void print() { System.out.println("Question 52"); }
    public static void print1() { System.out.println("Question 52"); }
}

class Test extends Question52 {
    public void print() { System.out.println("Test"); }
    public static void print1() { System.out.println("Test"); }
    public static void main(String[] args) {
        Question52 question52 = new Test();
        Test test = new Test();
        question52.print();
        test.print();
        question52.print1();
        test.print1();
    }
}
```

Test
Test
Question 52
Test

53. (5 pts) What happens when you try to compile and execute the following program? Explain your answer.

```java
public abstract class Question53 implements Comparable {
    private int num;
    public Question53(int num) {
        this.num = num;
    }
    public int compareTo(Object o) {
        return(-1);
    }
    public static void main(String[] args) {
        Question53 question53 = new Question53(1);
        Question53 question532 = new Question53(2);
        System.out.println(question53.compareTo(question532));
    }
}
```

There will be a compile-time error because the class is declared to be abstract, and we can't directly create an instance of it.
54. (5 pts) Is there a compile-time error in the following code? Will there be a runtime error? If not, then list all possible things that can be printed. You may use the back of each page of the exam and request more paper if you want. Explain your answer.

```java
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Question54 extends JApplet implements ActionListener, ItemListener {
    private int count;

    public void init() {
        JButton button = new JButton("Test");
        getContentPane().add(button);
        button.addActionListener(this);

        JComboBox box = new JComboBox(new String[]{"0", "3.1415", "5", "7"});
        getContentPane().add(box);
        box.addItemListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        System.out.println("Hello");
    }

    public void itemPerformed(ItemEvent e) {
        count++;
        System.out.println(count);
    }
}
```

There will be a compile-time error since we don’t implement the itemStateChanged method from the ItemListener interface.
public void paint(Graphics g) {
    super.paint(g);
    g.setColor(Color.blue);
    g.drawOval(getWidth()/2-100, getHeight()/2-100, 200, 200);
    g.setColor(Color.red);
    int[] xpoints = {getWidth()/2-100, getWidth()/2, getWidth()/2+100};
    int[] ypoints = {getHeight()/2, getHeight()/2-100, getHeight()/2};
    g.drawPolygon(xpoints, ypoints, 3);
}
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Question56 extends JApplet implements ActionListener {
    private JButton button;
    private JTextField field;

    public void init() {
        getContentPane().setLayout(new FlowLayout());
        getContentPane().add(button = new JButton("Press Me"));
        getContentPane().add(field = new JTextField());
        field.setPreferredSize(new Dimension(100,20));
        button.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == button) {
            int length = (int)(100*Math.random());
            String string = "";
            for (int counter=0;counter<length;counter++)
                string += (char)(97 + (int)(26*Math.random()));
            field.setText(string);
        }
    }
}