I. Fill in the Blank (30 pts)

1. We can use the method **addImage** to add an image to a MediaTracker.

2. We can visualize a **RandomAccessFile** as an array of bytes stored in a file.

3. We can use a **FileInputStream** to open an external file in order to read bytes from it.

4. In order for a thread to obtain a lock on an object, it can enter a **synchronized** method or block of code in the object.

5. A method that is **deprecated** is not guaranteed to be supported in future releases of a language.

6. The subclasses of **Reader** are meant for reading from character files.

7. When using the readLine() method to read from a BufferedReader, we know we are done reading when that method returns **null**.

8. A running thread leaves the running state and enters the waiting state by calling the method **wait()**.

9. The unchecked exceptions are **Error**, **RuntimeException**, and their subclasses.

10. In order to get a reference to an applet’s code, we use the method **getCodeBase()**.

11. A system is said to be **fault-tolerant** if it can provide either full or reduced functionality even after an error or failure has occurred.

12. **Polymorphism** is a dynamic binding that occurs at runtime between a method call and its correct implementation.

13. If we have an instance of a Calendar called **c**, then to obtain the current minute, we use the code `c.get(Calendar.MINUTE)`.
14. When an exception occurs we say that an exception has been **thrown**.

15. When we call the method `start()` on a Thread, it enters the **ready** state.

16. In order to make the currently executing thread pause for one minute, we would use the code `Thread.sleep(60000)`.

17. The method **repaint()** when called in an applet is a request to the context of the applet to call it’s paint method as soon as possible.

18. The head of the branch of the Java tree which describes exceptional behaviors is **Throwable**.

19. If a method might cause a checked exception to occur, we can sometimes declare that by using a **throws** clause in the method header.

20. The method **notify()** informs only one waiting thread to leave the waiting state and go back to the ready state.

21. **JApplet** and **JFrame** are two classes in the javax.swing package that are not lightweight components.

22. Given an ObjectOutputStream object, we can write the bytes of an object created from a Serializable class definition to a file with the method **writeObject**.

23. Serializable is a **tag** interface since it doesn’t contain any methods or constants.

24. Subclasses of **Exception** are usually errors during programming that we can recover from.

25. When calling the method readObject on an ObjectInputStream, we need to deal with a potential **ClassNotFoundException** in addition to a possible IOException from just opening a connection to a file.

26. We use the method **Thread.currentThread()** to get a reference to the currently executing thread.

27. If we create a FileWriter with a constructor that only accepts one String parameter, then if the file exists already, it will be **overwritten**.

28. Java supports a **termination** model of exception handling.
**II. Matching** (10 pts) Please choose the most appropriate choice for each term from the candidates on the right.

| 29. -1 I | A. A superclass of exceptions that we normally can recover from. |
| 30. Unchecked exceptions P | B. A superclass of exceptions that we normally can't recover from. |
| 31. Runnable R | C. The package that contains the AudioClip class and the Calendar class. |
| 32. java.applet U | D. The method used within an applet to retrieve information sent in through a param tag. |
| 33. Error B | E. A piece of code that executes once an exception occurs. |
| 34. Exception Handler E | F. An abstract superclass in the java.io package whose subclasses are meant to read from character files. |
| 35. throw N | G. A piece of code that causes the program to exit anytime a RuntimeException occurs whether we have a catch block or not. |
| 36. getParameter D | H. All classes except Error, RuntimeException, and their subclasses. |
| 37. Writer K | I. The value returned from the read method on a BufferedReader when there are no more characters to read. |
| 38. finally block J | J. A block of code attached to a try or try-catch that always executes no matter what. |

K. An abstract superclass in the java.io package whose subclasses are meant to write to character files.

L. All subclasses of Exception.

M. The first valid position in a String.

N. Causes an exception to occur.

O. The package that contains the Calendar class.

P. Error, RuntimeException, and their subclasses.

Q. The interface a class definition implements in order to become a deprecated class definition.

R. The interface a class definition implements in order to run in its own thread.

S. A method in a JFrame used to load a parameter from HTML Code that didn't load the JFrame.

T. Precedes a list of checked exceptions a method might cause.

U. The package that contains the AudioClip class.

V. A block of code attached to a try or try-catch that may or may not execute.
III. Short Answer (30 pts)

39. (5 pts) What are the elements of the data hierarchy? If there are counterparts for the various elements in Java, list those.

  - bits
  - bytes
  - characters
  - fields - instance variables
  - records - class definitions
  - files

40. (3 pts) What are three properties of a well-defined recursive method?

  - It must call itself either directly or indirectly
  - It must have an exit criterion
  - It must solve a “smaller” problem each time it is called to work toward the exit criterion

41. (4 pts) In what situations is a finally block optional? When is it not optional?

A finally block is optional if a try block has at least one associated catch block. It is not optional if a try block doesn’t have a corresponding catch block.

42. (5 pts) Suppose we have a class definition called MyClass which is abstract and contains an abstract method called draw. Is it possible to have an instance of MyClass called myClass and call the draw method on it? If so, how is this possible? What is this an example of?

Yes it is possible if we create a subclass of MyClass that fills in the missing pieces of MyClass. We can create an instance of a subclass and call it a superclass. This is an example of polymorphism.

43. (3 pts) What are three ways for a thread in the waiting state to go back to the ready state?

  - A call to notify() by another thread holding a lock on the object
  - A call to notifyAll() by another thread holding a lock on the object
  - An interruption
44. (4 pts) Three applet instance methods walk into a bar. The bartender calls them Larry, Moe, and Curly. Larry says that he is the first method called by the applet’s context when the applet is executed. Curly says he’s sick of getting called everytime there is a request to repaint the applet. Moe says that he’s also tired of being called whenever the applet’s page is returned to. What methods are represented by Larry, Moe, and Curly?

   **Larry is the init() method**  
   **Moe is the start() method**  
   **Curly is the paint(Graphics g) method**

45. (4 pts) Suppose we have a class definition called MyClass which does not contain an extends clause nor a constructor in its class definition. Assuming that the Object class has a no-argument constructor, which of the following assignment statements is correct syntax? Explain.

   a) MyClass myClass = new Object();  
   b) Object o = new MyClass();

Since every instance of MyClass is an instance of Object, we can call a MyClass instance an Object. However, we can’t create an instance of the superclass Object and call it a MyClass instance. Therefore, option b is correct syntax.

46. (1 pt) Given that String is a subclass of Object, does the following make any sense? Explain.

   throw “Hello”;

   **No, it doesn’t make sense. We can only throw something that’s Throwable. Since String is a subclass of Object, it isn’t a subclass of Throwable**

47. (1 pt) Given an instance of an Image called `image`, how would we create an image whose width and height are 1/2 the width and height of the original?

   `image.getScaledInstance(image.getWidth(this)/2,image.getHeight(this)/2,Image.SCALE.DEFAULT)`
IV. Problem Solving and Coding (30 pts)

48. (5 pts) Is there any syntax error with the following code? If so, what is the problem?

```java
try {
    int num = Integer.parseInt("Hello");
} catch (Throwable t) {
    System.out.println(t);
} catch (Exception e) {
    System.out.println(e);
}
```

Yes, there is a syntax error. We can’t have a catch block for a superclass before a catch block for a subclass.

49. (5 pts) In the following code we have a BufferedInputStream called input. We want to read all of the bytes from the BufferedInputStream and eventually have them in an array referred to by bytes. What is wrong with the following code? Assume this is called from within a try block that has a catch block that catches IOException.

```java
byte[] bytes = new byte[0];
byte[] buffer = new byte[4000*1024];
int length = 0;
while ((length = input.read(buffer,0,buffer.length)) != -1) {
    byte[] temp = new byte[bytes.length+buffer.length];
    System.arraycopy(buffer,0,temp,0,bytes.length);
    System.arraycopy(buffer,0,temp,bytes.length,buffer.length);
    bytes = temp;
}
input.close();
```

Within the while loop, we make the temporary array too large. Since the last time we read from the stream, there may not be 4 MB, so only a portion of buffer contains bytes. Instead of making the temporary array be the size of the original plus the size of buffer, we should make it the size of the original plus the number of bytes that were read.
50. (5 pts) Suppose that we have a class definition, MyClass, which contains the method get Object and suppose this method potentially causes a ClassNotFoundException which is a subclass of Exception but not of RuntimeException. If we have a reference to a MyClass object called myClass, is there a syntax error when a class definition containing the following method is compiled? Explain.

```java
public String toString() {
    String output = myClass.getObject().toString();
    return(output);
}
```

Yes there is a syntax error. Since the getObject() method may throw a ClassNotFoundException, and it is not an unchecked exception, we must either catch it or declare it.

51. (5 pts) Is there any syntax error in the following code? If not, how many times is the word Hello printed when the code is interpreted? Explain.

```java
import java.io.*;
public class Question51 {
    public static void main(String[] args) {
        try {
            FileReader file = new FileReader("review.txt");
        } catch (IOException ie) {
            finally {
                System.out.println("Hello");
            }
            if (true)
                throw new RuntimeException();
            System.out.println("Hello");
        }
    }
}
```

Hello will only be printed once. The finally block always executes. When the RuntimeException is thrown, the program exits so it doesn’t reach the last line. (You also received credit if you mentioned the missing brace)
52. (5 pts) Write a class definition for a JFrame which runs in its own thread. The activity of the thread is to print out integers beginning at 0. Assume that you have at your disposal a method to create a JButton which accepts a label, width, height, xlocation, ylocation, and an ActionListener. Add two JButton, suspend and resume, to the JFrame's definition so that suspend will cause the thread to go into the waiting state, and resume will move the thread from the waiting state to the ready state. Recall that the ActionListener interface contains the method public void actionPerformed(ActionEvent e).

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

public class Question52 extends JFrame implements Runnable, ActionListener {
    private JButton suspend;
    private JButton resume;
    private int counter;
    private Thread thread;
    private boolean keepGoing;
    private boolean suspended;
    private JButton createJButton(String label,
        int width,
        int height,
        int xlocation,
        int ylocation,
        ActionListener listener) {
        JButton button = new JButton(label);
        button.setSize(width, height);
        button.setLocation(xlocation, ylocation);
        Insets insets = button.getInsets();
        insets.left = 0;
        insets.right = 0;
        button.setMargin(insets);
        button.addActionListener(listener);
        return(button);
    }

    public Question52(String title) {
        super(title);
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container container = getContentPane();
        container.setLayout(null);
        container.add(suspend = createJButton("Suspend", 100, 20, 50, 50, this));
        container.add(resume = createJButton("Resume", 100, 20, 160, 50, this));
        setVisible(true);
        thread = new Thread(this);
        keepGoing = true;
        thread.start();
    }

    public void paint(Graphics g) {
        super.paint(g);
        g.drawString(counter + "", 50, 200);
    }

    public void run() {
        counter = 0;
        while (keepGoing) {
            try {
```

Thread.sleep(1000);
} catch (InterruptedException ie) {
}
synchronized (this) {
    while (suspended)
        try {
            wait();
        } catch (InterruptedException ie) {
        }
    counter++;
    repaint();
}

public synchronized void actionPerformed(ActionEvent e) {
    if (e.getSource() == suspend)
        suspended = true;
    else if (e.getSource() == resume) {
        suspended = false;
        notify();
    }
}

public static void main(String[] args) {
    Question52 question52 = new Question52(“Question 52”);
}
53. (5 pts) Write a class definition called MyClass which has two instance variables, an int and a String. In this class definition you should place an instance method which will write the current instance of the class to a file called “output”. Make sure you include all necessary code.

```java
import java.io.*;
public class Question53 implements Serializable {
    private int number;
    private String string;
    public Question53(int number, String string) {
        this.number = number;
        this.string = string;
    }
    public void store() {
        try {
            FileOutputStream file = new FileOutputStream("output");
            ObjectOutputStream output = new ObjectOutputStream(file);
            output.writeObject(this);
            output.close();
        } catch (IOException ie) {
        }
    }
}
```