Practice Problems
Event Handling, Networking, Threads

1 **Which library must be imported in order to use GUI event handling in Java?**

`java.awt.event.*`

2 **What is the *Event Dispatch Thread*?**

A background thread devoted to handling GUI events from the *Abstract Window Toolkit* (AWT). Events are generated by GUI components and sent to the **event queue**. The *Event Dispatch Thread* is responsible for dispatching these events from the event queue to the appropriate listeners.

3 **Consider the following class declaration.** Which methods *must* `Foo` implement?

```
class Foo implements ItemListener { ... }
```

```
public void itemStateChanged(ItemEvent ie) { ... }
```

4 **Consider the following code.** Write a private, inner class called `Listener` within `ClickCounter`. The class should provide the appropriate logic to increment `counter` and redisplay the `Label` object `l` when the `Button` object `b` is clicked. *Be sure to implement the appropriate interface!*

```
public class ClickCounter extends Applet {
    Button b;
    Label l;
    int counter = 0;

    public void init() {
        l = new Label("clicked " + counter + " times.");
        this.add(l);
        b = new Button("click me");
        this.add(b);
        b.addActionListener(new Listener());
    }

    // solution
    private class Listener implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            counter++;
            l.setText("clicked " + counter + " times");
        }
    }
}
```

5 **Consider the following code.** Write a private, inner class called `Listener` within `WhichButton`. The class should provide the approriate logic to print to the console which button was clicked – i.e. if `a` is clicked you should print a message to the console saying so, if `b` is clicked you should do the same. *Be sure to implement the appropriate interface!*

```
public class WhichButton extends Applet {
    Button a, b;
    Listener lis = new Listener();

    public void init() {
        a = new Button("a");
        b = new Button("b");
        this.add(a);
        this.add(b);
        a.addActionListener(lis);
        b.addActionListener(lis);
    }

    //solution
    private class Listener implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            if(ae.getSource() == a)
                System.out.println("a was clicked.");
            else if(ae.getSource() == b)
                System.out.println("b was clicked.");
        }
    }
}
```

6 **Consider the following class declaration.** Which methods *must* `Foo` implement?

```
class Foo implements KeyListener { ... }
```

```
public void keyPressed(KeyEvent ke) { ... }
public void keyReleased(KeyEvent ke) { ... }
public void keyTyped(KeyEvent ke) { ... }
```

7 **Write an applet called `ClickLocator`.** It should display the X and Y coordinates of the mouse each time that it is clicked within the applet. Include a private, inner class called `Listener` which implements the `MouseListener` interface. It should also have a `paint()` method.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class ClickLocator extends Applet {
    int x, y;

    public void init() {
        this.addMouseListener(new Listener());
    }

    public void paint(Graphics g) {
        g.drawString(x + ", " + y, x, y);
    }

    private class Listener implements MouseListener {
        public void mouseClicked(MouseEvent me) {
            x = me.getX();
            y = me.getY();
            repaint();
        }

        public void mouseEntered(MouseEvent me) {}
        public void mouseExited(MouseEvent me) {}
        public void mousePressed(MouseEvent me) {}
        public void mouseReleased(MouseEvent me) {}
    }
}
```

8 **Re-write the `Listener` class from the previous question using an adapter class.** *You only need to write the listener class, nothing else is necessary.*

```
private class Listener extends MouseAdapter {
    public void mosueClicked(MosueEvent me) {
        x = me.getX();
        y = me.getY();
        repaint();
    }
}
```

9 **Name one advantage and one disadvantage of using an adapter class?**

Advantage: there is no need to implement unneeded methods in the associated interface. For instance, in the `MouseAdapter` class above.

Disadvantage: you must `extend` an adapter class, which means you may not `extend` any other class.

10 What is the difference between low-level events and high-level (or, semantic) events?

Low-level events generally refer to individual input and output events related to underlying hardware – i.e. mouse clicks, key presses, etc.

High-level events generally refer to some event within the GUI which is not necessarily triggered by a single or specific low-level event – i.e. a button on a GUI can be pressed via a mouse click, clicking enter on the keyboard, tapping a touch screen, etc. However, in each scenario it simply makes sense to speak of a button click regardless of the triggering low-level event.

11 What is the difference between `paint()` and `repaint()`?

The `paint()` method is overridden in order to supply the logic for how to paint a particular component. It is called by the Event Dispatch Thread whenever the component needs to be redrawn – i.e. when the window is maximized after being minimized. This method is not called explicitly by the programmer.

The `repaint()` method is a means for the programmer to have some control over when certain components get redrawn. A call to this method schedules a call to the `paint()` via the Event Dispatch Thread.

12 What is the *Transmission Control Protocol* (TCP) responsible for? What are TCP tranmission units called?

TCP is responsible for partitioning data into small, numbered transmission units called **segments**. It is concerned with the reliability of the transmission and ensures that all tranmission units have arrived, sorts them, requests to resend any missing units, etc.

13 What is the *Internet Protocol* (IP) responsible for? What are IP tranmission units called?

IP is responsible for keeping track of the source and destination of segments received from TCP. When passed to IP, segments become **packets** which are the IP transmission units. IP is concerned with routing packets to their destination often through several different machines and is not concerned with the order in which packets are received are whether they are received.

14 What two peices of information are required to open a socket?

An IP address and a port number.

15 In Java, what is the difference between a `Socket` and a `ServerSocket`?

The `ServerSocket` class plays a passive role in network communication and is unidirectional. It resides on the server end of the client-server model and listens for an incoming connection. One simply needs to specify a local port in order to create a `ServerSocket`. The `ServerSocket` then listens on that port.

The `Socket` class, on the other hand, plays a more active role and is bidirectional. It is created in order to initiate protocol exchanges with a server. In order to create a `Socket` one must specify an IP address with which to establish a connection and a remote port at that IP address on which the server is listening.

Note that a `Socket` is initially created on the client. However, when a connection has been established, the `ServerSocket` object creates a regular `Socket` object on the server. So, both client and server have regular `Socket` objects, though the `ServerSocket` object is used to establish the connection from the client.

16 In *Hypertext Tranfer Protocol* (HTTP), which of the client and server sends the request and which the response?

The client sends the request, the server sends the response.

17 Imagine you were to write a program that read multiline data over a network. Explain why you would need to establish a protocol to handle this situation and how you might design such a protocol. *No need to write code here, just English.*

A protocol is required because the program would need to know when to stop reading data from the network stream. One way to establish a protocol would be to use a predetermined delimiter to signify the end of a given data segment.

18 What are the two ways to create threads using the Java libraries?

(1) `extend` the `Thread` class.
(2) `implement` the `Runnable` interface.

19 What is a context switch?

When the processor switches from one thread (or, process) to another.

20 **Consider the following code.** Then, answer the proceeding questions.

```
class FooBar {
    public static void main(String [] args) {
        Thread t = new Thread() {
            public void run() {
                System.out.println("Foo!");
            }
        };

        t.start();
        System.out.println("Bar!");
    }
}
```

What will the code above print?

Bar!
Foo!

Is it guaranteed to print this?

No, this is not guaranteed as there may be a context switch before the reaching the print statement in `main()`.

21 **Consider the following code.** Then, answer the proceeding questions.

```
class FooBar {
    public static void main(String [] args) throws InterruptedException {
        Thread t = new Thread() {
            public void run() {
                System.out.println("Foo!");
            }
        };

        t.start();
        t.join();
        System.out.println("Bar!");
    }
}
```

What will the code above print?

Foo!
Bar!

Is it guaranteed to print this? Why or why not?

Yes, this is guaranteed as there is a call to `join()` immediately after the call to `start()`. This means that the `main` thread will not continue executing until thread `t` terminates.

22 **Consider the code below.** Then, answer the proceeding questions.

```
class Foo {
    int x = 0;

    public void add() {
        if(x == 0) x++;
    }
}
```

Why is the code above problematic in a multithreaded environment? Explain.

The code above contains two race conditions. Namely, it performs an action on a variable after reading it and then checking its value against a condition. Since these operations are not atomic – i.e. each are separate execution statements – a context switch is liable to occur between these operations. If the context switch executes some other thread which makes a call to the same method, the data is liable to be corrupted.

How can you fix the problem? Explain.

Use the `synchronized` modifier to ensure only a single thread is executing the method at a time.

23 **Determine whether the following statements are true or false.**

true	All GUI events in Java are handled with a single thread
false	GUI components send events directly to their listeners
true	`KeyEvent`s are low-level events
false	`MouseEvent`s are high-level events
false	`ActionEvent`s are low-level events
true	`ItemEvent`s are high-level events
false	A client is said to listen on a port
true	A server is said to listen on a port
false	Mutable objects are thread safe
true	Immutable objects are thread safe