True or False: 30 points (2 points each)

1.) True or False: Inner classes can be useful for exporting object state in an encapsulated way.

2.) True or False: IP is a reliable network protocol.

3.) True or False: Private class members are not inherited by derived classes.

4.) True or False: A non-static inner class can access all of its surrounding class's protected fields, but not its private fields.

5.) True or False: Synchronizing methods can help to prevent deadlock.

6.) True or False: Two methods in the same class cannot have the same method signature.

7.) True or False: The resultant type of an expression is the widest type of the expression's operands.

8.) True or False: Variables a and b are references to the same object. If the value of a is changed, b will change as well.

9.) True or False: A new thread can be started by calling the thread's run method.

10.) True or False: A subclass can access its parent's protected members if the subclass is in a different package than its parent.

11.) True or False: Anonymous inner classes must extend a base class or implement an interface.

12.) True or False: A hashtable can often perform its operations in constant time, even if it contains a large number of elements.

13.) True or False: An exception that occurs in a method must be handled in that method.

14.) True or False: A variable cannot have an abstract class as its type.

15.) True or False: A cast is required to convert from an int to a double.
Fill in the Blank: 45 Points (3 points each)

16.) To write an object to a stream, that object's class must implement the ________________ interface.

17.) A(n) ________________ occurs when two or more threads simultaneously use a shared resource.

18.) A(n) ________________ is an abstraction for one side of a network connection.

19.) ________________ is a principle which prevents external entities from invalidating an object's state.

20.) The ________________ keyword is used by an object to refer to itself.

21.) Unchecked ________________ are typically caused by programmer error.

22.) Polymorphism, or the ability of a variable to store many different types of related objects, is made possible by ________________ and ________________.

23.) Recursion terminates when a(n) ________________ is reached.

24.) ________________ and ________________ are objects that can be used to define thread-safe access to shared resources.

25.) The ________________ method is one way a thread can notify another thread that it should terminate.

26.) To connect to an application on another computer, you need that computer's IP address and a ________________ used by the remote application.

27.) The ________________ statement can be used to skip the remainder of statements in a loop and start at the beginning of the next iteration.

28.) A(n) ________________ represents a position in a collection and allows a programmer to step over each element.

29.) To ensure semantically equivalent objects are treated equally by Java collections, a programmer should override the ________________ and ________________ methods.

30.) A programmer wants to store an ordered, known number of objects. The programmer should use a(n) ________________ to store the objects.
Multiple Choice: 40 Points (5 points each)

31.) What output will the following code produce?

```java
int a = 5;
int b = a++ + 7;
System.out.println(a + " , " + b);
```

a.) 5, 12  
b.) 6, 13  
c.) 5, 13  
d.) 6, 12  
e.) None of the above.

32.) Identify the problem with the following piece of code:

```java
protected Server (int port) throws IOException {
    this.port = this.port;
    ServerSocket ss = new ServerSocket(this.port);
}
```

a.) A ServerSocket requires that an ip address be passed to its constructor.  
b.) The ServerSocket constructor cannot throw an IOException.  
c.) The variable port is never assigned.  
d.) All of the above.

33.) What do the methods accept from java.net.ServerSocket and readObject from java.io.ObjectInputStream have in common? Select all that apply.

a.) They are both static.  
b.) They are both blocking method calls.  
c.) They are both part of the java.net.Communicator interface.  
d.) None of the above.

34.) What is the value of num at the end of the loop below?

```java
int count = 15;
int num = 0;
for (int i=0; i<count*1/2; i++)
    num++;
```

a.) 15  
b.) 7  
c.) 0  
d.) 14
35.) Select the correct constructor call for initializing the Hashtable below:

    Hashtable<String, Handler> clients;

a.) new Hashtable();
b.) new Hashtable<? extends String, ? extends Handler>();
c.) new Hashtable<String, Handler>();
d.) new Hashtable<Object, Object>();

36.) You are designing a program and have two otherwise unrelated classes which share some common internal functionality. The programming pattern best suited for this is:

a.) Inheritance
b.) Aggregation
c.) Polymorphism
d.) Multiple Representation

37.) Select all that apply. The Java Programming Language:

a.) Is an interpreted language.
b.) executes inside of a natively compiled application.
c.) is an object oriented language.
d.) None of the above.

38.) Select all that apply. Methods known as *mutators*:

a.) Must always be private.
b.) Can help maintain encapsulation by ensuring a value to be assigned to instance variables is valid.
c.) Should be avoided in program design.
d.) Can transform four normal turtles into radical pizza scarfing ninja dudes.

Short Answer: 60 Points (10 points each)

39.) In a few sentences, explain how generics in Java help to promote type-safety.

40.) Briefly explain how deadlock can occur in a multi-threaded application.

41.) In a few sentences, explain the purpose of exceptions in Java.

42.) Although interpreted languages do suffer some overhead in comparison to natively compiled languages, they do offer some advantages. Identify at least one advantage of executing inside of an interpreted environment.
43.) Using the `java.io.OutputStream` class as an example, explain how polymorphism and multiple representation can help to make programs more flexible.

44.) Briefly describe the purpose of an application protocol in a networking application.

**Principles in Practice: 75 Points (15 points each)**

This is meant to evaluate your understanding of programmatic control structures and object relationships needed to solve problems in Java. You will *not* be penalized for minor syntax errors.

45.) The following code excerpt is intended to save data to a file using an OutputStream. Identify the programming error in the code and correct it.

```java
try {
    output.write(data);
    output.close();
}
catch (IOException ioe) {
    System.out.println("Saving failed.");
}
```

46.) Using the stack trace, identify the problem with the following code excerpt:

Line No.
10: `public ClientWriter (Socket s) {
...
14: this.getOutputStream = this.socket.getOutputStream();
15: }`

```
Exception in thread "main" java.lang.NullPointerException
  at ClientWriter.<init>(ClientWriter.java:14)
  at Server.main(Server.java:140)
```

47.) Modify the following code fragment to print a message to the command line when the button is clicked.

```java
public MyGUI () {
    this.messageButton = new JButton("Press Me!");
}
```
48.) Ken has written the following code for his multi-threaded Library application and cannot figure out why his program is not getting past a certain point. Briefly explain Ken's problem, and alter his code to correct it.

```java
public synchronized Book checkOutBook (String title) throws Exception {
    while (!inStock(title))
        Thread.sleep(1000);
    Book theBook = this.availableBooks.get(title);
    return theBook;
}
```

```java
public synchronized void returnBook (Book book) {
    this.availableBooks.put(book.getTitle(), book);
}
```

49.) Using the following class definitions, implement the insert method for BinaryTree using recursion. Use the back of this page if you need to.

```java
public class MappedItem {
    protected MappedItem leftChild, rightChild;
    protected int key;
    protected Object value;
}
```

```java
public class BinaryTree {
    protected MappedItem root;

    public SortedIntegerList () {
        this.root = null;
    }

    public void insert (MappedItem m) {
    // (closing bracket omitted)
    }
```