**Question 1**

What is the output for the following code?

**public** **class** Outer {

**private** **int** a = 7;

**class** Inner {

**public** **void** displayValue() {

System.*out*.println("Value of a is " + a);

}

}

}

**public** **class** TestSet {

**public** **static** **void** main(String[] args) **throws** Exception {

Outer mo = **new** Outer();

Outer.Inner inner = mo.**new** Inner();

inner.displayValue();

}

}

**Anwer:** 7

Explanation : An inner class instance can never stand alone without a direct relationship to an instance of the outer class.

you can access the inner class through a live instance of the outer class.

Inner class can access private member of the outer class.

**Question 2**

What is the result of attempting to compile and run the following code?

**public** **class** B {

**public** String getCountryName() {

**return** "USA";

}

**public** StringBuffer getCountryName() {

StringBuffer sb = **new** StringBuffer();

sb.append("UK");

**return** sb;

}

**public** **static** **void** main(String[] args) {

B b = **new** B();

System.*out*.println(b.getCountryName().toString());

}

}

**Answer:** Compile with error

Explanation : You cannot have two methods in the same class with signatures that only differ by return type.

**Question 3**

What is the result of attempting to compile and run the following code?

**public** **class** A {

**public** String getName() **throws** ArrayIndexOutOfBoundsException {

**return** "Name-A";

}

}

**public** **class** **C** **extends** A {

**public** String getName() **throws** Exception {

**return** "Name-C";

}

}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

A a = **new** C();

a.getName();

}

}

**Answer:** Compile with error

Exception is not compatible with throws clause in A.getName().

Overridden method should throw only same or sub class of the exception thrown by super class method.

**Question 4**

What is the result of attempting to compile and run the following code?

**public** **class** A {

}

**public** **class** B **implements** Serializable {

A a = **new** A();

**public** **static** **void** main(String[] args){

B b = **new** B();

**try**{

FileOutputStream fs = **new** FileOutputStream("b.ser");

ObjectOutputStream os = **new** ObjectOutputStream(fs);

os.writeObject(b);

os.close();

}**catch**(Exception e){

e.printStackTrace();

}

}

}

**Answer:** java.io.NotSerializableException: Because class A is not Serializable.

Explanations : It throws java.io.NotSerializableException: A Because class A is not Serializable.  
When JVM tries to serialize object B it will try to serialize A also because (A a = new A()) is instance variable of Class B. So thows NotSerializableException.

**Question 5**

What is the result of attempting to compile and run the following code?

public class B extends Thread {

public static void main(String argv[]) {

B b = new B();

b.run();

}

public void start() {

for (int i = 0; i < 10; i++) {

System.*out*.println("Value of i = " + i);

}

}

}

**Answer:** Clean compile but no output at runtime

Explanation: This is a bit of a sneaky one as I have swapped around the names of the methods you need to define and call when running a thread.  
 If the for loop were defined in a method called public void run() and the call in the main method had been to b.start() The list of values from 0 to 9 would have been output.

**Question 6**

What modifiers are allowed for variables in an Interface?

**Answer:** Interface Variables should be PUBLIC, Static and FINAL.

**Question 7**

What is a local, member and class variable?

**Answer:** A member variable is a variable that belongs to an object, whereas a local variable belongs to the current scope. Fields that have the static modifier in their declaration are called static fields or class variables.

**Question 8**

Can an anonymous class be declared as implementing an interface and extending a class?

**Answer:**

An anonymous class may implement an interface or extend a superclass, but may not be declared to do both.

**Question 9**

What is a Marker Interface? Give an example?

**Answer:**

A so-called **marker** interface is a Java interface which doesn't actually define any fields. It is just used to "mark" Java classes which support a certain capability -- the class marks itself as implementing the interface. For example, the [java.lang.Cloneable](http://java.sun.com/j2se/1.3/docs/api/java/lang/Cloneable.html) interface.

**Question 10**

What are the ways in which you can instantiate a thread?

**Answer:**

We can create a *thread* by either extending the *Thread* class or by implementing the Runnable

**Question 11**

What JSP lifecycle methods can I override?

**Answer:**

You cannot *override* the \_jspService() *method* within a *JSP* page. You *can* however, *override* the jspInit() and jspDestroy() *methods*

**Question 12**

How can I implement a thread-safe JSP page?

**Answer:**

You can make your JSPs thread-safe by having them implement the SingleThreadModel interface.

This is done by adding the directive <%@ page isThreadSafe="false" %> within your JSP page.

**Question 13**

Is there a way to reference the "this" variable within a JSP page?

**Answer:** The page implicit object is equivalent to "this", and returns a reference to the servlet generated by the JSP page.

**Question 14**

What are the implicit objects in a JSP?

**Answer:**

Implicit objects in jsp are the objects that are created by the container automatically and the container makes them available to the developers.

\* request

\* response

\* pageContext

\* session

\* application

\* out

\* config

\* page

\* exception

**Question 15**

How can I enable session tracking for JSP pages if the browser has disabled cookies?

**Answer:**

If the browser does not support cookies, or if cookies are disabled, you can still enable session tracking using URL rewriting and use the following page directive in each jsp.

<%@ page session="true" %>

**Question 16**

Write an algorithm to illustrate Merge Sort?

void mergesort(int lo, int hi)

{

if (lo<hi)

{

int m=(lo+hi)/2;

mergesort(lo, m);

mergesort(m+1, hi);

merge(lo, m, hi);

}

}

// Straightforward variant

void merge(int lo, int m, int hi)

{

int i, j, k;

// copy both halves of a to auxiliary array b

for (i=lo; i<=hi; i++)

b[i]=a[i];

i=lo; j=m+1; k=lo;

// copy back next-greatest element at each time

while (i<=m && j<=hi)

if (b[i]<=b[j])

a[k++]=b[i++];

else

a[k++]=b[j++];

// copy back remaining elements of first half (if any)

while (i<=m)

a[k++]=b[i++];

}

**Question 17**

Write an algorithm to find the number of occurrences of an element in a given array?

/\*\* Counts the number of times an integer appears in an array. \*/

public static int findCount(int[] a, int k) {

int count = 0;

for (int i=0; i < a.length;i++) {

if (a[i]== k) // check if the current element equals k

count++;

}

return count;

}

**Question 18**

Write an algorithm to find whether the given string is a palindrome?

Public class palindrome  
{  
Boolean isPalindrome(String testString){  
    return(testString.Equals(reverse(testString)));  
}  
  
public static String reverse(String s){  
    String and;  
      
    if (s.Length() <= 1)  
        return s;  
      
    else{     
        char lastC = s.CharAt(s.Length()-1);          
        String stringLeft = s.Substring(0, s.Length() -1);  
        return and = lastC + reverse(stringLeft);     
        }  
    }  
}

**Question 19**

Write the iterative linear algorithm to get the Fibonacci series?

**public** **class** FibonacciIterative **{**

**public** **static** **int** fib**(int** n**)** **{**

**int** prev1=0, prev2=1;

**for(int** i=0; i<n; i++**)** **{**

**int** savePrev1 = prev1;

prev1 = prev2;

prev2 = savePrev1 + prev2;

**}**

**return** prev1;

**}**

**}**

**Question 20**

Write a recursive function to find the position of the element in an array using binary search?

**int binarySearch(int[] array, int value, int left, int right) {**

**if** (left > right)

**return** -1;

**int** middle = (left + right) / 2;

**if** (array[middle] == value)

**return** middle;

**else** **if** (array[middle] > value)

**return** binarySearch(array, value, left, middle - 1);

**else**

**return** binarySearch(array, value, middle + 1, right);

}