**Question 1**

Explain what is a recursive subroutine.

**Answer :** A recursive subroutine is simply one that calls itself either directly or through a chain of calls involving other subroutines.

**Question 2**

**What is a binary tree?**

**Answer :** A binary tree is a tree in which each node has at most two subtrees.   
  -The left of a node contains only nodes with keys less than the node's key. -The right subtree of a node contains only nodes with keys greater than node's key. Both the left and right subtrees must also be binary search trees.

**Question 3**

What is an *activation record?* What role does a stack of activation records play in a computer?

**Answer :** When a subroutine is called, an activation record is created to hold the information that is needed for the execution of the subroutine, such as the values of the parameters and local variables. This activation record is stored on a stack of activation records. A stack is used since one subroutine can call another, which can then call a third, and so on. Because of this, many activation records can be in use at the same time. The data structure is a stack because an activation record has to continue to exist while all the subroutines that are called by the subroutine are executed. While they are being executed, the stack of activation records can grow and shrink as subroutines are called and return.

**Question 4**

Explain the following code sample: result = someCondition ? value1 : value2;

**Question 5**

You plan to write a program that uses several basic collection interfaces: Set, List, Queue, and Map. You're not sure which implementations will work best, so you decide to use general-purpose implementations. Which implementations are these?

**Answer :**   
Set: HashSet   
List: ArrayList   
Queue: LinkedList   
Map: HashMap

**Question 6**

What are collection pools? What are their advantages?

**Answer :** A connection pool is a cache of database connections that is maintained in memory, so that the connections may be reused.

**Question 7**

What is the output of the following code?

public class JavaTest {

public static void main(String[] args) {

String s1 = new String("Test");

String s2 = "Test";

if (s1==s2)

System.out.println("Same");

if (s1.equals(s2))

System.out.println("Equals");

}

}

**Answer :** Print **“**Equals”

**Question 8**

What is the result of attempting to compile and run this ?

**public** **class** JavaTest {

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

Thread.*sleep*(2000);

System.*out*.println("Java Quiz");

}

}

**Answer :** Print “Java Quiz”

**Question 9**

What is the result of attempting to compile and run this ?

public class Test {

public static void main(String[] args){

Float f = new Float(16/0);

System.out.println(f.isNaN());

}

}

**Answer :** Runtime error

**Question 10**

What is the essential difference in functionality between a TreeMap and a HashMap?

**Answer :** The key/value pairs in a TreeMap are sorted so that the keys are in ascending order. (For this reason, it must be possible to compare the keys in a TreeMap, either by using the compareTo() method or a Comparator.)

**Question 11**

**What will be the output of the following code?**

**class** Base {

**protected** **int** i = 10;

**public** **int** get(){

**return** i;

}

}

**public** **class** Derived **extends** Base{

**protected** **int** i = 20;

**public** **int** get(){

**return** i;

}

**public** **static** **void** main(String argv[])

{

Base b = **new** Derived();

System.out.println(b.i);

System.out.println(b.get());

}

}

**Answer :** 10 and 20. Methods are invoked virtually – the exact method to be executed is determined at runtime according to the actual class of the object on which it was called. Data members, however, do not really override each other. Each class has its member i, and the one to be retrieved is determined at compile time, according to the declared class of the reference (in this case Base).

**Question 12**

When compiling this code, what will happen? Why?

**class** Parent{

**protected** **void** x(){}

**public** **void** y(){}

}

**public class** Child **extends** Parent{

**public** **void** x(){}

**protected** **void** y(){}

}

**Answer :** Compilation error – y can not have its visibility reduced. Reducing visibility of methods when overriding is not permitted.

**Question 13**

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

public class Test {

public static void main(String[] args){

Integer a = new Integer(4);

Integer b = new Integer(8);

Integer c = new Integer(4);

TreeSet ts = new TreeSet();

ts.add(a);

ts.add(b);

ts.add(c);

System.out.println(ts);

}

}

**Answer :** Will print [4, 8]

**Question 14**

What will this program print out ?

class Base{

static int value = 0;

Base(){

addValue();

}

static void addValue(){

value += 10;

}

int getValue(){

return value;

}

}

class Derived extends Base{

Derived(){

addValue();

}

static void addValue(){

value += 20;

}

}

public class Test {

public static void main(String[] args){

Base b = new Derived();

System.out.println(b.getValue());

}

}

**Answer : 30**

**Question 15**

Is there any problem with this exception handler? Will this code compile? What message will be displayed when the code is executed?

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

**try**{

String s = **null**;

s=s.substring(10);

}

**catch**(Exception e){

System.out.println("Caught exception!");

}

**catch**(RuntimeException e){

System.out.println("Caught a runtime exception!");

}

}

**Answer :** The code doesn’t compile. RuntimeException derives from Exception, so the ordering of the catch blocks makes the second unreachable.

**Question 16**

What is the difference between RequestDispatcher and sendRedirect?

**Answer :** RequestDispatcher: server-side redirect with request and response objects. sendRedirect : Client-side redirect with new request and response objects.

**Question 17**

**What do you understand by servlet mapping?**

**Answer :** Servlet mapping defines an association between a URL pattern and a servlet. You can use one servlet to process a number of url pattern (request pattern). For example in case of Struts \*.do url patterns are processed by Struts Controller Servlet.

**Question 18**

Describes the relationship between JSP and servlets.

**Answer :** JSPs are built on servlet semantics and all JSPs are compiled to servlets for runtime usage.

**Question 19**

What are the implicit objects in JSP technology?

**Answer :** Implicit objects are objects that are created by the web container and contain information related to a particular request, page, or application. They are:  
--request   
--response   
--pageContext   
--session   
--application   
--out   
--config   
--page   
--exception

**Question 20**

How can I declare methods within my JSP page?

**Answer :** You can declare methods for use within your JSP page as declarations. The methods can then be invoked within any other methods you declare, or within JSP scriptlets and expressions.