**Question 1**

Give names of two sorting algorithms?

**Answer :** Bubble sort, Insertion sort, Shell sort,Merge sort, Heap sort, Quick sort,Radix sort, Bucket sort, Distribution sort, Shuffle sort.

**Question 2**

What is priority queue?

**Answer :** An abstract data type to efficiently support finding the item with the highest priority across a series of operations. The basic operations are: insert, find-minimum (or maximum), and delete-minimum (or maximum). Some implementations also efficiently support join two priority queues (meld), delete an arbitrary item, and increase the priority of a item (decrease-key).

**Question 3**

What is the basic difference between a stack and a queue?

**Answer :** In a stack, items are added to the stack and removed from the stack on the same end (called the "top" of the stack). In a queue, items are added at one end (the "back") and removed at the other end (the "front"). Because of this difference, a queue is a FIFO structure (items are removed in the same order in which they were added), and a stack is a LIFO structure (the item that is popped from a stack is the one that was added most recently).

**Question 4**

Consider the following code snippet.

int i = 10;

int n = i++%5;

1. What are the values of i and n after the code is executed? **Answer :**  i is 11, and n is 0.
2. What are the final values of i and n if instead of using the postfix increment operator (i++), you use the prefix version (++i))? **Answer :** i is 11, and n is 1.

**Question 5**

What is the result of trying to compile and run this program.

public class Test{

public static void main(String[] args){

int[] a = {1};

Test t = new Test();

t.increment(a);

System.out.println(a[a.length - 1]);

}

void increment(int[] i){

i[i.length - 1]++;

}

}

**Answer :** Compiles and runs printing out 2. You are passing a reference to an array as the argument to the method. The method may not modify the passed object reference but it can modify the object itself.

**Question 6**

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

public class Test {

public static void main(String[] args){

String s = "HelloWorld".substring(5,10);

System.out.println(s);

}

}

**Answer :** Compile and run printing out "World"

**Question 7**

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

public class Test {

public static void main(String[] args){

byte a = 10;

Byte b = new Byte(a);

Byte c = new Byte(11);

System.out.println(b.compareTo(c));

}

}

**Answer :** Compiler error

**Question 8**

What is the fundamental property that distinguishes Sets from other types of Collections?

**Answer :** A Set cannot contain repeated elements. (The equals() method is used to test whether two objects should be considered the same.)

**Question 9**

Is there anything wrong with this exception handler as written? Will this code compile?

try {

} catch (Exception e) {

} catch (ArithmeticException a) {

}

**Answer :** The first handler catches exceptions of type Exception; therefore, it catches any exception, including ArithmeticException. The second handler could never be reached. This code will not compile.

**Question 10**

What would be the output of the following program:

**public** **class** Overloader{

**public** **static** **void** read(String s){

System.out.println("read(String)");

}

**public** **static** **void** read(Integer i){

System.out.println("read(Integer)");

}

**public static void** read(Object o){

System.out.println("read(Object)");

}

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

Object s = **new** String("Java");

Integer i = new Integer(10);

Overloader.read(s);

Overloader.read(i);

}

}

**Answer :** read(s) prints “read(Object)”  
        read(i) prints “read(Integer)”

The method signature to be invoked is determined at **compile time**. Therefore, it depends on the declared type and not the runtime type of the argument/s.

**Question 11**

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);

HashSet hs = new HashSet();

hs.add(a);

hs.add(b);

hs.add(c);

System.out.println(hs);

}

}

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

**Question 12**

Comsider the following class hierarchy. What will be the outcome on attempting to compile and run this ?

interface A {

public void method1();

}

class One implements A{

public void method1(){

System.out.println("hello");

}

}

class Two extends One{}

public class Test extends Two{

public static void main(String[] args)

{

A a;

Two t = new Two();

a = t;

a.method1();

}

}

**Answer :** Compiles and runs printing out "hello". Object reference conversion is possible here. The old type which is class can be assigned to an interface type as long as the class implements that interface.

**Question 13**

What will this program print out ?

class Base{

int value = 0;

Base(){

addValue();

}

void addValue(){

value += 10;

}

int getValue(){

return value;

}

}

class Derived extends Base{

Derived(){

addValue();

}

void addValue(){

value += 20;

}

}

public class Test {

public static void main(String[] args){

Base b = new Derived();

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

}

}

**Answer : 40**

**Question 14**

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

interface ITest{

public void setVal();

}

public class Test {

private String a;

void aMethod(){

final String b = " World";

ITest it = new ITest() {

public void setVal(){

a = "Hello" + b;

}};

it.setVal();

System.out.println(a);

}

public static void main(String[] args) {

Test t = new Test();

t.aMethod();

}

}

**Answer :** Will compile and run printing "Hello World"

**Question 15**

What is the advantage of using a PreparedStatement?

**Answer :** For SQL statements that are executed repeatedly, using a PreparedStatement object would almost always be faster than using a Statement object. This is because creating a PreparedStatement object by explicitly giving the SQL statement causes the statement to be precompiled within the database immediately. Thus, when the PreparedStatement is later executed, the DBMS does not have to recompile the SQL statement and prepared an execution plan - it simply runs the statement. Typically, PreparedStatement objects are used for SQL statements that take parameters. However, they can also be used with repeatedly executed SQL statements that do not accept parameters.

**Question 16**

Explain about servletConfig and servletContext in applications?

**Answer :** Servletcontext is used to obtain application level information and only one servletcontext can be present in one application. ServletConfig object is present for every servlet and it provides initialization parameters for every servlet.

**Question 17**

How does JSP handle runtime exceptions?

**Answer :** Using errorPage attribute of page directive and also we need to specify isErrorPage=true if the current page is intended to URL redirecting of a JSP.

**Question 18**

**What is the difference between using getSession(true) and getSession(false) methods?**

**Answer :** getSession(true) – This method will check whether already a session is existing for the user. If a session is existing, it will return that session object. Otherwise, it will create new session object and return that object. getSession(false) – This method will check existence of session. If session exists, then it returns the reference of that session object. If not, this methods will return null.

**Question 19**

How many JSP scripting elements and what are they?

**Answer :** There are three scripting language elements:  
--declarations   
--scriptlets   
--expressions

**Question 20**

How do I perform browser redirection from a JSP page?

**Answer :** You can use the response implicit object to redirect the browser to a different resource, as:  
response.sendRedirect("http://www.exforsys.com/path/error.html");  
You can also physically alter the Location HTTP header attribute.