**Question 1**

Give two names of typical data structure. What are the three operations on a *stack?*

**Answer :**  Stack, Queue, Linked list, Doubly Linked List. The three stack operations are *push, pop,* and *isEmpty.* The definitions of these operations are: push(item) adds the specified item to the top of the stack; pop() removes the top item of the stack and returns it; and isEmpty() is a boolean-valued function that returns true if there are no items on the stack.

**Question 2**

**What are differences between tree and graph?**

**Answer :** -Tree has no loop but graph has loops.  -Tree always has direction but graph do not have direction.

**Question 3**

Explaining what is meant by parsing a computer program.

**Answer :** To parse a computer program means to determine its syntactic structure, that is, to figure out how it can be constructed using the rules of a grammar (such as a BNF grammar).

**Question 4**

How long is the string returned by the following expression? What is the string?

"Was it a car or a cat I saw?".substring(9, 15)

**Answer :**  It's 3 characters in length: “car or”. It does not include the space after car.

**Question 5**

What is the result that will be printed out ?

void aMethod()

{

float f = (1 / 4) \* 10;

int i = Math.round(f);

System.out.println(i);

}

**Answer : 0.** The result of 1/4 will be zero because integer division is carried out on the operands. If you need to obtain a fractional value you need to use either a float or double literal as in 1F / 4F.

**Question 6**

What is the result of trying to compile and run the following code.

public static void main(String[] args){

double d = 10 / 0;

if (d == Double.POSITIVE\_INFINITY)

System.out.println("Positive infinity");

else

System.out.println("Negative infinity");

}

**Answer :** Runtime exception. Division by zero on integer literals will throw

a runtime error.

**Question 7**

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

public class Test {

public static void main(String[] args)

{

Float f = new Float(32D);

System.out.println(f);

}

}

**Answer :** Compiles and runs printing out "32.0"

**Question 8**

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

public class Test {

public static void main(String[] args){

Integer i = new Integer(256);

System.out.println(i.byteValue());

}

}

**Answer :** Runs and prints "0"

**Question 9**

What is an Object's Hash Code in Java?

**Answer :** Objects in Java have hash codes associated with them. An object's hash code is a signed number that identifies the object (for example, an instance of the parent class). An object's hash code may be obtained by using the object's hashCode() method as follows:

int hashCode = SomeObject.hashCode();

**Question 10**

Can ClassCastException be thrown without the need to declare it in the method header? Why?

Answer: Yes. ClassCastException is an unchecked exception, so it does not need to be declared.

**Question 11**

What is the output of the following code?

**class** Parent{

**private** **void** f(){

System.out.println("Parent.f()");

}

**public** **void** g(){

System.out.println("Parent.g()");

f();

}

}

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

**public** **void** f(){

System.out.println("Child.f()");

}

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

Parent p = **new** Child();

p.g();

}

}

**Answer :**  Parent.g()  
 Child.f()

**Question 12**

What will this print out ?

public class Test {

public static void main(String[] args){

Integer a = new Integer(8);

Integer b = new Integer(4);

Integer c = new Integer(4);

Vector vec = new Vector();

Iterator itr;

vec.add(a);

vec.add(b);

vec.add(c);

itr = vec.iterator();

while (itr.hasNext()) {

ystem.out.println("" + itr.next());

}

}

}

**Answer :** 8 , 4 and 4

**Question 13**

What is the result of compiling and running this program?

class Mammal{

void eat(Mammal m){

System.out.println("Mammal eats food");

}

}

class Cattle extends Mammal{

void eat(Cattle c){

System.out.println("Cattle eats hay");

}

}

class Horse extends Cattle{

void eat(Horse h){

System.out.println("Horse eats hay");

}

}

public class Test{

public static void main(String[] args){

Mammal h = new Horse();

Cattle c = new Horse();

c.eat(h);

}

}

**Answer :** prints "Mammal eats food". The method that will be called is the one

from class Mammal. The reasons are quite obvious.

**Question 14**

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

class Base{

String s = "Base";

String show(){

return s;

}

}

class Derived extends Base{

String s = "Derived";

}

public class Test {

void print(Base b){

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

}

void print(Derived d){

System.out.println(d.show());

}

public static void main(String[] args){

Test t = new Test();

Base b = new Derived();

t.print(b);

}

}

**Answer :** Will compile and run printing "Base".

**Question 15**

How do you call a Stored Procedure from JDBC?

**Answer :**  The first step is to create a CallableStatement object. As with Statement and PreparedStatement objects, this is done with an open Connection object. A CallableStatement object contains a call to a stored

procedure.

CallableStatement cs =

con.prepareCall("{call SHOW\_SUPPLIERS}");

ResultSet rs = cs.executeQuery();

**Question 16**

When a servlet accepts a call from a client, it receives two objects. What are they?

**Answer :**  ServeltRequest: which encapsulates the communication from the client to the server.  
ServletResponse: which encapsulates the communication from the servlet back to the client.  
ServletRequest and ServletResponse are interfaces defined by the javax.servlet package.

**Question 17**

**How you can destroy the session in Servlet? Write the Java code that can be used to** destroy the session.

**Answer :**  You can call invalidate() method on the session object to destroy the session. e.g. **session.invalidate();**

**Question 18**

Why use RequestDispatcher to forward a request to another resource, instead of using a sendRedirect?

**Answer :**  The RequestDispatcher does not require a round trip to the client, and thus is more efficient and allows the server to maintain request state.

**Question 19**

What is the difference in using request.getRequestDispatcher() and context.getRequestDispatcher()?

**Answer :**  request.getRequestDispatcher(path): In order to create it we need to give the relative path of the resource context.getRequestDispatcher(path): In order to create it we need to give the absolute path of the resource.

**Question 20**

What is the difference between ServletContext and PageContext?

**Answer :** ServletContext: Gives the information about the container  
PageContext: Gives the information about the Request