1. List two main keywords used in java which are not used in c++

super: The super keyword in Java is a reference variable which is used to refer immediate parent class object.

extends: The extends keyword extends a class (indicates that a class is inherited from another class).

2. Distinguish between boxing and unboxing

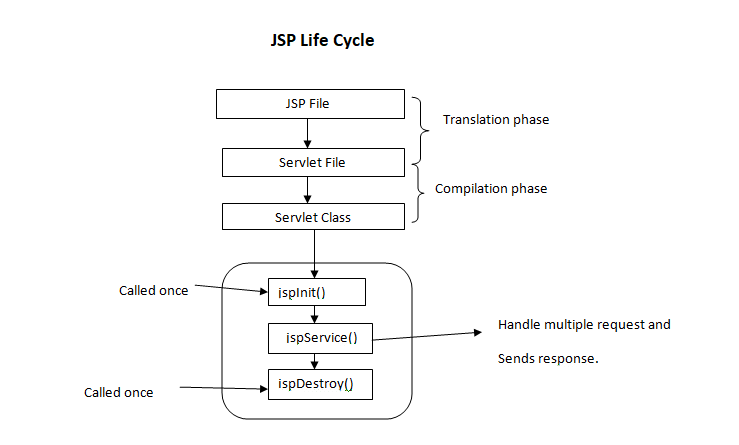
The basic difference between Boxing and Unboxing is that Boxing is the conversion of the value type to an object type whereas, on other hands, the term Unboxing refers to the conversion of the object type to the value type.

3. Recall the advantages of servlet over cgi

Servlets are server side components that provides a powerful mechanism for developing server web applications for server side. Earlier CGI was developed to provide server side capabilities to the web applications. Although CGI played a major role in the explosion of the Internet, its performance, scalability and reusability issues make it less than optimal solutions. Java Servlets changes all that. Built from ground up using Sun’s write once run anywhere technology java servlets provide excellent framework for server side processing.

Using servlets web developers can create fast and efficient server side applications and can run it on any servlet enabled web server. Servlet runs entirely inside the Java Virtual Machine. Since the servlet runs on server side so it does not depend on browser compatibility.

4. Identify the life cycle methods for a JSP



5. Differentiate ServletContext and PageContext.

In JSP, the ServletContext shares information about the container in which the servlet is running in. This can be set up in the web application descriptor. There can be only one v ServletContext per web application. Compared to this, the PageContext gives the servlet information about the request that it is currently managing and contains information about the request, the response object, the session, and also a reference to the ServletContext of the web application.

6. Describe multiple inheritance and write a program to show how it can be achieved in java

Multiple Inheritance is a feature of an object-oriented concept, where a class can inherit properties of more than one parent class. The problem occurs when there exist methods with the same signature in both the superclasses and subclass. On calling the method, the compiler cannot determine which class method to be called and even on calling which class method gets the priority.

When the child class extends from more than one superclass, it is known as multiple inheritance. However, Java does not support multiple inheritance.

To achieve multiple inheritance in Java, we must use the interface.

interface Backend {

// abstract class

public void connectServer();

}

class Frontend {

public void responsive(String str) {

System.out.println(str + " can also be used as frontend.");

}

}

// Language extends Frontend class

// Language implements Backend interface

class Language extends Frontend implements Backend {

String language = "Java";

// implement method of interface

public void connectServer() {

System.out.println(language + " can be used as backend language.");

}

public static void main(String[] args) {

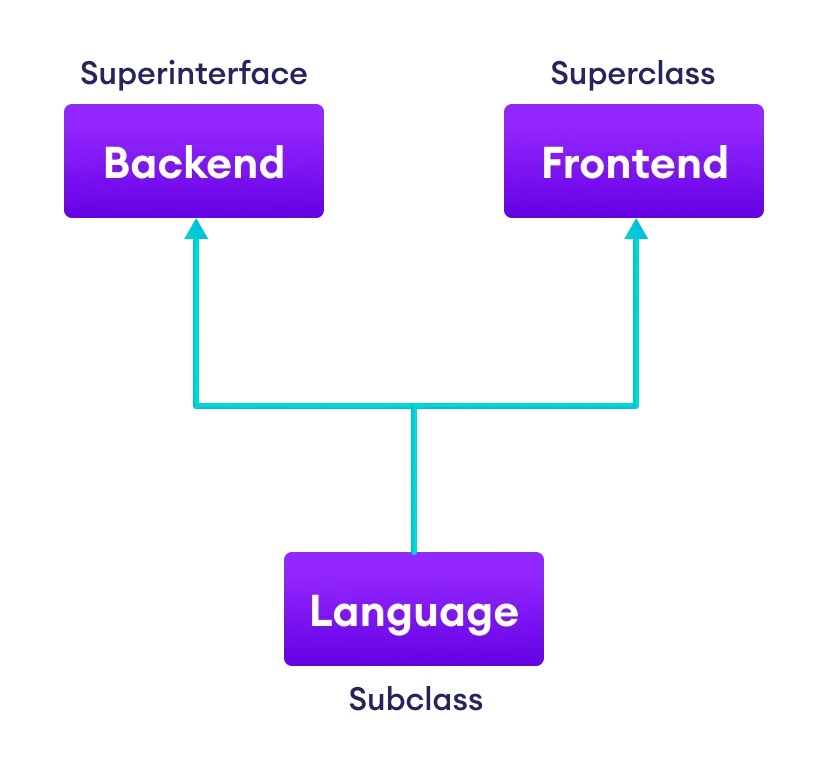
// create object of Language class

Language java = new Language();

Output

Java can be used as backend language.

Java can also be used as frontend.



7. Explain the various linkedlist class methods with the aid of a program

import java.util.\*;

class Book {

int id;

String name,author,publisher;

int quantity;

public Book(int id, String name, String author, String publisher, int quantity) {

    this.id = id;

    this.name = name;

    this.author = author;

    this.publisher = publisher;

    this.quantity = quantity;

}

}

public class LinkedListExample {

public static void main(String[] args) {

    //Creating list of Books

    List<Book> list=new LinkedList<Book>();

    //Creating Books

    Book b1=new Book(101,"Let us C","Yashwant Kanetkar","BPB",8);

    Book b2=new Book(102,"Data Communications & Networking","Forouzan","Mc Graw Hill",4);

    Book b3=new Book(103,"Operating System","Galvin","Wiley",6);

    //Adding Books to list

    list.add(b1);

    list.add(b2);

    list.add(b3);

    //Traversing list

    for(Book b:list){

    System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);

    }

}

}

Output:

101 Let us C Yashwant Kanetkar BPB 8

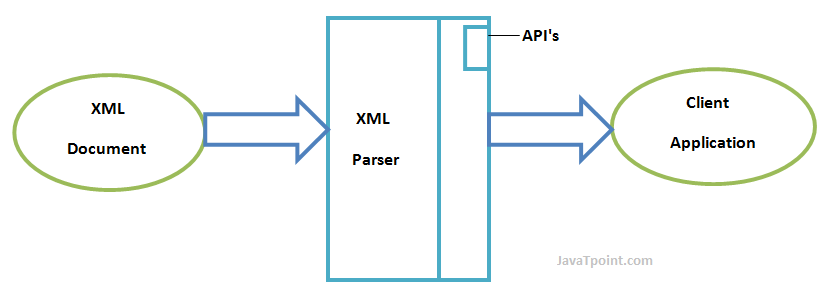
102 Data Communications & Networking Forouzan Mc Graw Hill 4

103 Operating System Galvin Wiley 6

8. Elaborate why XML parsing is important? Explain different types of XML parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.



These are the two main types of XML Parsers:

1. DOM
2. SAX

## DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

### Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

### Advantages

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

## SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

### Features of SAX Parser

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

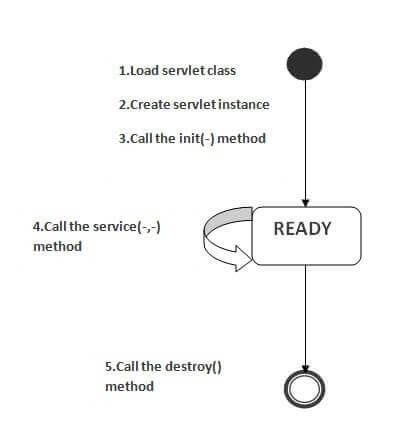
It is an event based parser, it works like an event handler in Java.

### Advantages

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

9. Illustrate steps involved in lifecycle of a Servlet with the help of diagram



### 1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

### 2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

### 3) init method is invoked

|  |
| --- |
| The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below: |

1. public void init(ServletConfig config) throws ServletException

### 4) service method is invoked

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

1. public void service(ServletRequest request, ServletResponse response)
2. throws ServletException, IOException

### 5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

1. public void destroy()

10. Demonstrate with the help of program the usage of try, catch, throw, throws and finally keyword in java

1. [**try**](https://www.geeksforgeeks.org/flow-control-in-try-catch-finally-in-java/): The try block contains a set of statements where an exception can occur.

try

{

// statement(s) that might cause exception

}

2. catch: The catch block is used to handle the uncertain condition of a try block. A try block is always followed by a catch block, which handles the exception that occurs in the associated try block.

catch

{

// statement(s) that handle an exception

// examples, closing a connection, closing

// file, exiting the process after writing

// details to a log file.

}

3. throw: The throw keyword is used to transfer control from the try block to the catch block.

4. throws: The throws keyword is used for exception handling without try & catch block. It specifies the exceptions that a method can throw to the caller and does not handle itself.

5. finally: It is executed after the catch block. We use it to put some common code (to be executed irrespective of whether an exception has occurred or not ) when there are multiple catch blocks.

class ThrowsExecp {

    // This method throws an exception

    // to be handled

    // by caller or caller

    // of caller and so on.

    static void fun() throws IllegalAccessException

    {

        System.out.println("Inside fun(). ");

        throw new IllegalAccessException("demo");

    }

    // This is a caller function

    public static void main(String args[])

    {

        try {

            fun();

        }

        catch (IllegalAccessException e) {

            System.out.println("caught in main.");

        }

finally {

            System.out.println("I am in final block");

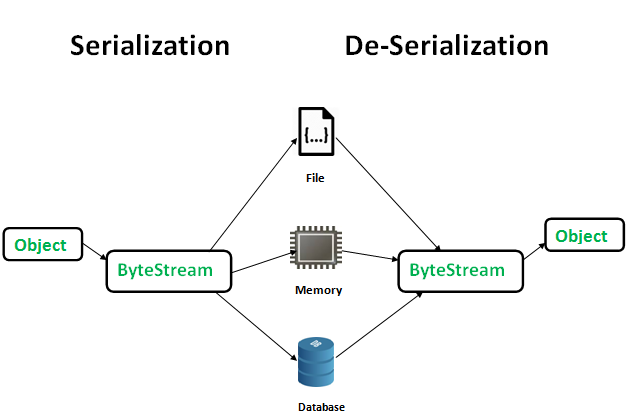
        }

    }

}

11. Describe serialization, deserialization and the significance of transient variables. Write code to justify your answer

Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.



The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.

To make a Java object serializable we implement the java.io.Serializable interface.  
The ObjectOutputStream class contains writeObject() method for serializing an Object.

public final void writeObject(Object obj)

throws IOException

The ObjectInputStream class contains readObject() method for deserializing an object.

public final Object readObject()

throws IOException,

ClassNotFoundException

|  |
| --- |
| // Java code for serialization and deserialization  // of a Java object  import java.io.\*;    class Demo implements java.io.Serializable  {      public int a;      public String b;        // Default constructor      public Demo(int a, String b)      {          this.a = a;          this.b = b;      }    }    class Test  {      public static void main(String[] args)      {          Demo object = new Demo(1, "geeksforgeeks");          String filename = "file.ser";            // Serialization          try          {              //Saving of object in a file              FileOutputStream file = new FileOutputStream(filename);              ObjectOutputStream out = new ObjectOutputStream(file);                // Method for serialization of object              out.writeObject(object);                out.close();              file.close();                System.out.println("Object has been serialized");            }            catch(IOException ex)          {              System.out.println("IOException is caught");          }              Demo object1 = null;            // Deserialization          try          {              // Reading the object from a file              FileInputStream file = new FileInputStream(filename);              ObjectInputStream in = new ObjectInputStream(file);                // Method for deserialization of object              object1 = (Demo)in.readObject();                in.close();              file.close();                System.out.println("Object has been deserialized ");              System.out.println("a = " + object1.a);              System.out.println("b = " + object1.b);          }            catch(IOException ex)          {              System.out.println("IOException is caught");          }            catch(ClassNotFoundException ex)          {              System.out.println("ClassNotFoundException is caught");          }        }  } |

Output :

Object has been serialized

Object has been deserialized

a = 1

b = geeksforgeeks

A transient variable is a special type of variable which we create by using the transient keyword. It is a special type of variable which have a non-serialized value at the time of serialization. A variable that is initialized by its default value during de-serialization is known as a transient variable.

A transient variable plays an important role in preventing an object from being serialized. We can make any variable transient by using the transient keyword.

12. Explain JSP declaration, JSP directive and JSP scriptlets tags using suitable examples

## JSP Declaration

* A declaration tag is a piece of [Java](https://www.guru99.com/java-tutorial.html) code for declaring variables, methods and classes. If we declare a variable or method inside declaration tag it means that the declaration is made inside the servlet class but outside the service method.
* We can declare a static member, an instance variable (can declare a number or string) and methods inside the declaration tag.

Syntax of declaration tag:

<%! Dec var %>

Here Dec var is the method or a variable inside the declaration tag.

Example:

In this example, we are going to use the declaration tags

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Guru Declaration Tag</title>

</head>

<body>

<%! int count =10; %>

<% out.println("The Number is " +count); %>

</body>

</html>

## JSP Scriptlet

* Scriptlet tag allows to write Java code into JSP file.
* JSP container moves statements in \_jspservice() method while generating servlet from jsp.
* For each request of the client, service method of the JSP gets invoked hence the code inside the Scriptlet executes for every request.
* A Scriptlet contains java code that is executed every time JSP is invoked.

Syntax of Scriptlet tag:

<% java code %>

Here <%%> tags are scriplets tag and within it, we can place java code.

Example:

In this example, we are taking Scriptlet tags which enclose java code.

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Guru Scriplet</title>

</head>

<body>

<% int num1=10;

int num2=40;

int num3 = num1+num2;

out.println("Scriplet Number is " +num3);

%>

</body>

</html>

The **jsp directives** are messages that tells the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives:

* page directive
* include directive
* taglib directive

### Syntax of JSP Directive

1. <%@ directive attribute="value" %>

### JSP page directive

The page directive defines attributes that apply to an entire JSP page.

### Syntax of JSP page directive

1. <%@ page attribute="value" %>