**JAVA PROGRAMMING FORMAT**

1. **COMMENT SECTION 🡪 Optional**
2. **PACKAGE SECTION 🡪 Optional**
3. **IMPORT SECTION 🡪 Optional**
4. **CLASSES/INTERFACES SECTION 🡪 Optional**
5. **MAIN CLASS SECTION 🡪 Mandatory**

MAIN CLASS SECTION

CLASSES/INTERFACES SECTION

IMPORT SECTION

PACKAGE SECTION

COMMENT SECTION

1. **Comment Section:**

In java application development, before starting implementation part, it is convention to provide some details of the application or implementation like Author name, Objective, Client Details, Module details, Team Details,……

To provide this type of description in Java applications, we have to use Comment section, where we have to use Comments.

There are three types of comments in JAVA.

1. Single Line Comment
2. Multi Line Comment
3. Documentation Comment
4. Single Line Comment:

It allows the complete description within a single line.

Syntax:

// ----Description-----

1. Multi Line Comment :

It allows the description in more than one line.

Syntax:

/\*

---description-------

---

\*/

1. Documentation Comment:

It allows the complete description in more than one page.

Syntax:

/\*\*

\*---

\*----

--

---

\*----

\*/

* In general, we will use documentation comments to prepare API Documentations.
* API Documentation: It is a document in form of text doc or .doc file or Html file or a PDF file, it is able to provide declarative information about the programming elements like classes, interfaces, variables, methods, constructors….. which we have used in our java application.

public class Employee implements java.io.Serializable, java.lang.Cloneable {

public String eid;

protected String ename;

public float esal;

String eaddr;

public Employee(){

}

public Employee(String eid){

}

public Employee(String eid, String ename){

}

public Employee(String eid, String ename, float esal){

}

public Employee(String eid, String ename, float esal, String eaddr){

}

public void add(String eid, String ename, float esal, String eaddr){

}

public void search(String eid){

}

public void update(String eid, String ename, float esal, String eaddr){

}

public void delete(String eid){

}

}

Employee.txt:

Class Name: Employee

Access Modifiers: public

Super Class: java.lang.Object

Interfaces:java.io.Serializable, java.lang.Cloneable

Variables: eid, ename, esal, eaddr

Constructors: Employee, Employee, Employee, Employee, Employee

Methods: add, search, update, delete

---

**Variables**:

eid:

ename:

esal:

**Methods**:

add:

Name: add

Access Modifiers: public

Return type: void

Parameter: eid, ename, esal, eaddr

Throws Exception: No Exception

Using documentation comments for API documents is not suggestible.

* To simplify the preparation of API documentation, JAVA has provided a predefined internal command in the form of “javadoc” at “C:\Java\jdk1.8.o\bin”.
* **NOTE:** javadoc command is able to provide API documentation for our program in the form of “HTML” files.
* Create a java file and write code in it at the below location in system
* **D:\BackUpFolder\JAVA\**
* In cmd, type ‘javadoc Employee.java’
* Load in browser

To provide description/Metadata in JAVA applications, JDK1.5 version has provided a new feature called as “Annotations”.

**Q) In java applications, to provide description we have already comments then what is the requirement to use Annotations?**

A) If we provide description in java applications in the form of comments, then the comments and its description will be removed by “Lexical Analysis” Phase as part of the compilation, it may not be available to the generated .class files and up to runtime of the applications.

As per the requirement, if we want to bring description or metadata up to .JAVA file, up to .class file and up to Runtime of our application, we need “Annotations”.

**Q) In java applications, to bring metadata up to Runtime we have already XML documents then What is the requirement to use Annotations?**

A) To provide metadata at Runtime of our Java applications if we use XML documents then we are able to get the following problems.

* 1. We must learn XML technology
  2. Every time we have to check whether XML documents are stored at proper locations or not, if not, there may be a chance to get java.io.FileNotFoundException.
  3. Every time we have to check whether XML document is well formed document or not.
  4. Every time we have to check whether we are using Right Parsing mechanisms or not to read data from XML document.

To overcome these problems we need JAVA alternative, that is, “Annotations”.

**NOTE:** In JAVA/J2EE Applications, we are able to use Annotations as an alternative to XML documents.

Xml Based Technologies Annotation Based Technologies [XML Docs are optional]

1. Up to JDK1.4 -----------------------------🡪 JDK1.5 and above
2. JDBC3.0 -----------------------------------🡪 JDBC4.0 and above
3. Servlets2.x--------------------------------🡪 Servlets3.x
4. Struts1.x-----------------------------------🡪 Struts2.x
5. JSF1.X--------------------------------------🡪 JSF2.X
6. EJBs2.x------------------------------------🡪 EJBs3.x
7. Spring2.4---------------------------------🡪 Spring2.5
8. Hibernet3.2.4---------------------------🡪 Hibernet3.2.5
9. **Package Section:**
10. Package is a collection of related classes and interfaces as a single unit.
11. Package is a folder contains .class files representing related classes and interfaces.

EX: java.io

java.awt.event

In Java Applications, Packages are able to provide the following advantages.

1. Modularity
2. Abstraction
3. Security
4. Sharability
5. Reusability

In Java, there are two types of packages:

1. Predefined packages:

These packages are defined by JAVA programming Language and these packages are provided along with JAVA software.

EX: java.io

java.util

java.awt

1. User defined packages:

These packages are defined by the developers as per their application requirements.

To declare user defined packages, we have to use the following syntax.

Package package\_name;

Where ‘package\_name’ may be a single name or the combination of parent package name and child package name.

EX: package p;

Package p1.p2.p3;

To use package declaration statements we have to follow two conditions.

1. Package declaration statement must be first statement in java files.
2. Package names must be unique, they must not be sharable and they must not be duplicated.

**Q) Is it possible to provide more than one package declaration statement in single java file?**

A) No, it is not possible to declare more than one package statement in single java file, because in JAVA files, package declaration statement must be first statement. Every java file is able to allow at most one package declaration statement.

EX: abc.java

Package p1; 🡪 valid

Package p2; 🡪 Invalid

Package p3; 🡪 Invalid

To provide package names, JAVA has provided a convention like to include our company domain name in reverse in package name.

[www.durgasoft.com](http://www.durgasoft.com)

durgasoft.com

com.durgasoft

package com.durgasoft.icici.transactions.deopsit;

com.durgasoft 🡪 company domain name in reverse.

icici 🡪 client / project name

transactions 🡪 module name

deposit 🡪 sub module name

1. **Import Section:**

If we want to access classes and interfaces of a particular package into the present java file then we have to import the respective package to the present JAVA file.

To import packages into present java file, we have two types of syntaxes.

* 1. import package\_Name.\*;
     + It is able to import all classes and interfaces of a particular package to the present JAVA file.

EX: import java.io.\*;

import java.util.\*;

* 1. import package\_Name.member\_Name;
     + It is able to import only the specified class or interface from the specified package to the present java file.

EX: import java.io.BufferedReader;

import java.util.ArrayList;

**Q) Is it possible to provide more than one import statement in a single JAVA File?**

A) Yes, it is possible to provide more than one import statement in a single java file, but we are able to provide only one package declaration statement.

EX: abc.java

Package p1; 🡪 valid

Packagep2; 🡪 Invalid

Package p3; 🡪 Invalid

Import java.io .\*; 🡪 valid

Import java.util.\*; 🡪 valid

Import java.sql.\*; 🡪 valid

**Q) Is it possible to use classes and interfaces of a particular package in the present JAVA file without importing the respective package?**

A) Yes, it is possible to use classes and interfaces of a particular package into the present java file without importing the respective package, but, we have to use “”Fully Qualified Names” to the classes and interfaces.

NOTE: Specifying class names and interfaces names along with their respective package names is called as Fully Qualified Name.

EX: java.io.BufferedReader

Java.util.ArrayList

Java.sql.connection

A java program with import statement:

Import java.io.\*

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

A java program without import statement:

Java.io.BufferedReader br = new java.io.BufferedReader(new

Java.io.InputStreamReader(System.in));

NOTE: In JAVA Applications, always, import statements are suggestible instead of Fully Qualified Names.

1. **Classes/Interfaces Section:**

The main intention of classes and interfaces of Java Applications is to represent all real world entities.

EX: Student, Employee, Customer, Account, Product, Transaction, ………..

In JAVA applications, there are no limitations for writing number of classes and interfaces; we are able to write any number of classes and interfaces in java applications as per the application requirement.

NOTE: In Real Time Application development, we will prepare a single class per java file, but it is possible to write any number of classes and interfaces in a single java file. But, writing more than one class or more than one interface is not suggestible.

Class Employee{

-----Variables and methods---

}

Interface EmployeeService{

---variables and methods----

}

1. **Main Class Section:**

If any Java class contains main() method then that java class is called as Main Class.

**Q) What is the requirement to write main() method in java applications?**

A)

* 1. To define application logic in Java applications, we need main() method.
  2. To define starting point and ending point to the application execution, we need main() method.