Introduction

DURGA SOFTWARE SOLUTIONS®

Java History

Java Details

Home : SUN Mc Systems (Oracle Corporation)
Author : James Gosling
Objective : To prepare simple electronic consumer goods.
Project : Green
First Version : JDK 1.0 (1996, Jan-23rd)
Used Version : Some org JDK5.0, Some other JAVA 6, JAVA 7
Latest Version : JAVA7, JAVA8, This April – JAVA 9
Type of Software : Open Source Software
Strong Features : Object-oriented, Platform Independent, Robust, Portable, Dynamic, Secure.......

Version	Code Name	Enhancements
1) JDK1.0[Jan,23,1996]	OAK	Language Introduction
2) JDK1.1[Feb,19,1997]	----	RMI, JDBC, Reflection API, Java Beans, Inner classes
3) JDK1.2[Dec,8,1998]	Playground	Strictfp, Swing, CORBA, Collection, Framework
4) JDK1.3[May,8,2000]	Kestrel	Updations on RMI, JNDI
5) JDK1.4[Feb,6,2002]	Merlin	Regular Expression, NIO, assert, Keyword, JAXP, ...
6) JDK5.0[Sep,30,2004]	Tiger	Autoboxing, var-arg method,static import, Annotations ,..
7) JAVA SE6[Dec,11,2006]	Mustang	JDBC4.0,GUI updations, Console
8) JAVA SE7[Jul,28,2011]	Dolphin	Strings in switch,'_' symbol in literals,try-with- resources
9) JAVA SE8[Mar,18,2014]	Spider	Interface improvements, Lambda Expression, Date-Time API, Updations on Collections

10)JAVA SE9[Sep,20017]	----	JSHELL, JPMS,Private Methods in Interfaces, ……
11)JAVA SE10[March,2018]	----	Local Variables Type Inference, GarbageCollector interface, Application Class Data Sharing,….
12)JAVA SE11[Sep,2018]	----	HttpClient,Local Variables Syntax for Lambda Parameter,….
13)JAVA SE12[March,2019]	----	Switch Expressions, JVM Constants,….
14)JAVA SE13[Sep,2019]	----	Text Blocks, Switch Expressions Updations, Dynamic CDS Archieves…..
15)JAVA SE14[March,2020]	----	Pattern Matching For instanceof, Records, Text Blocks Updations,……
16)JAVA SE15[Sep,2020]	-----	Updations on Text Blocks, Pattern Matching for instanceof operator, ….

Differences between Java and Others [C and C++]

1)C and C++ are static programming languages but JAVA is dynamic programming language:

If any programming language allows memory allocation for primitive data types at compilation time [Static Time] then that programming language is called as Static Programming language.
EX: C and C++.

In C and C++ applications, memory will be allocated for primitive data types at compilation time only, not at runtime.

If any programming language allows memory allocation for primitive data types at runtime, not at compilation time then that programming language is called as Dynamic Programming Language.
EX: JAVA

In java applications, memory will be allocated for primitive data types at runtime only, not at compilation time.

Note: In Java applications, memory will be allocated for primitive data types at the time of creating objects only, in java applications, objects are created at runtime only.

2)Pre-Processor is required in C and C++, but, Pre-Processor is not required in Java:

In case of C and C++, the complete predefined library is provided in the form of header files

EX:
stdio.h
conio.h
math.h

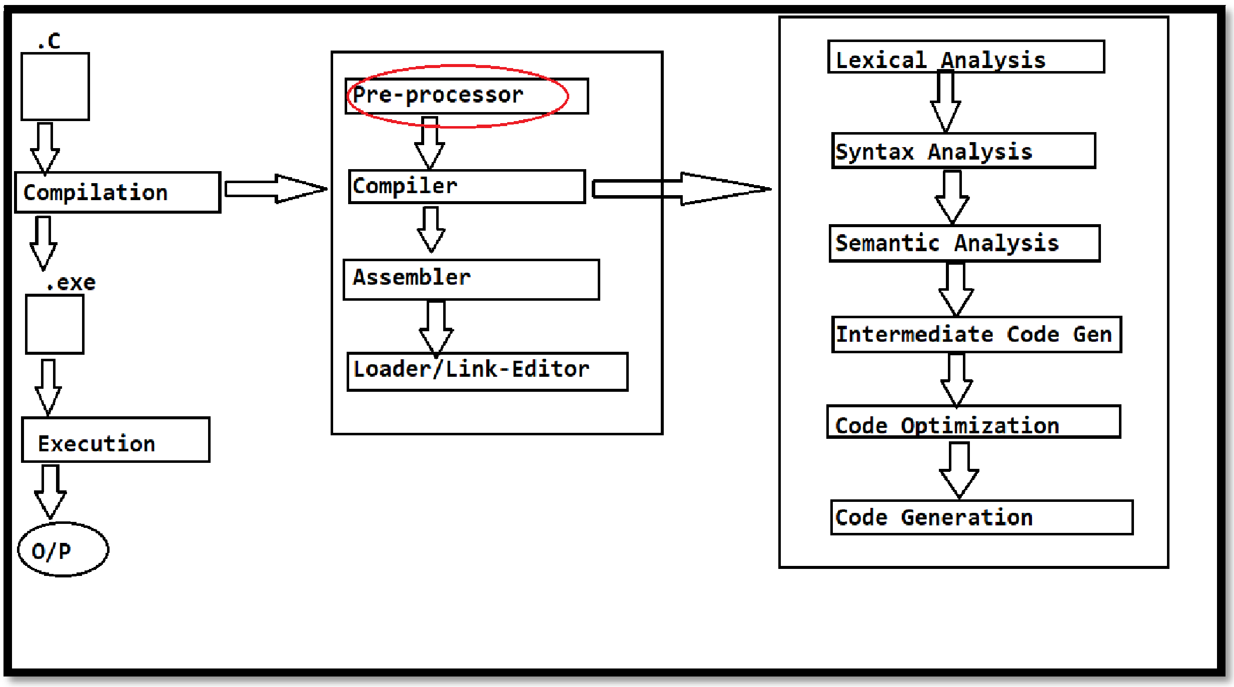
If we want to use predefined library in C and C++ applications, we have to include header files in C and C++ applications, for this, we have to use #include<> statement.

EX:
#include<stdio.h>
#include<conio.h>
#include<math.h>

If we compile C and C++ applications then Pre-Processor will perform the following actions.

1) Pre-Processor will recognize all #include<> statement.
2) Pre-Processor will take all the specified header files from #include<> statements.
3) Pre-Processor will check whether the specified header files are existed or not in C and C++ softwares.
4) If the specified header files are not existed the Pre-Processor will generate some error messages.
5) If the specified header files are existed then Pre-Processor will load the specified header files to the memory, this type of loading predefined library at compilation time is called as "Static Loading".

In C and C++ applications, Pre-Processor is required to recognize #include<> statements inorder to load header files to the memory.



In java , the complete predefined library is provided in the form of classes and interfaces in packages

<u>EX:</u>
java.io
java.util
java.sql

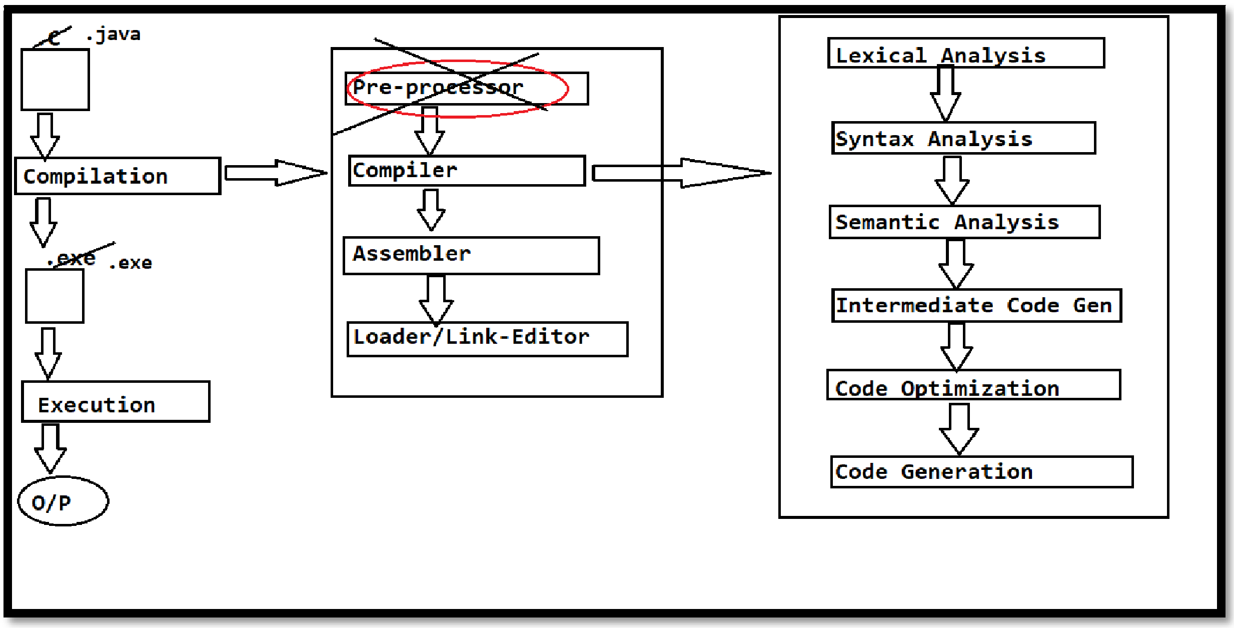
If we want to use predefined library in java applications then we have to include packages in java application, for this we have to use "import" statements
<u>EX:</u>
import java.io.*;
import java.util.*;
import java.sql.*;

If we compile java program then compiler will perform the following actions.

1.Compiler will recognize all the import statements.
2.Compiler will take the specified package names from import statements.
3.Compiler will check whether the specified packages are existed or not in java software.
4.If the specified packages are not existed in java predefined library then compiler will rise an error "package xxx does not exist".
5.If the specified packages are existed in java predefined library then Compielr will not rise any error and compiler will not load any package content to the memory.

While executing java program, when JVM[Java Virtual Machine] encounter any class or interface from the specified package then only JVM will load the required classes and interfaces to the memory at runtime, loading predefined library at runtime is called as "Dynamic Loading".

Pre-Processor is not required in JAVA , because, java does not include header files and #include<> statements, alternativily, JAVA has clases and interfaces in the form of packages and import statements.



Q)What are the differences between #include<> statement and import statement?

1. #include<> statement is available upto C and C++.
   import statement is available upto JAVA.

2. #include<> statements are used to include the predefined library which is available in the form of header files.
   import statements are used to include the predefined library which are available in the form of packages.

3. #include<> statement is providing static loading.
   import statement is providing dynamic loading.

4. #include<> statements are recognized by Pre-Processor.
   import statements are recognized by both Compiler and JVM.

5. By using Single #include<> statement we are able to include only one header file.

EX:
```
#include<stdio.h>
#include<conio.h>
#include<math.h>
```

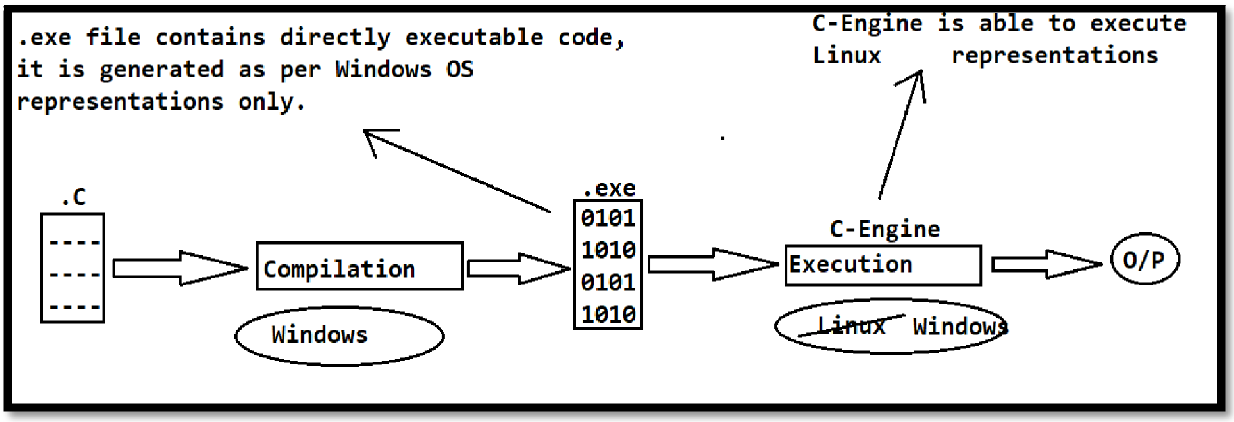
By using single import statement we are able to include more than one class or more than one interface of the same package.

EX: import java.io.*;

3)C and C++ are platform dependent programming languages, but, JAVA is platform Independent programming language.

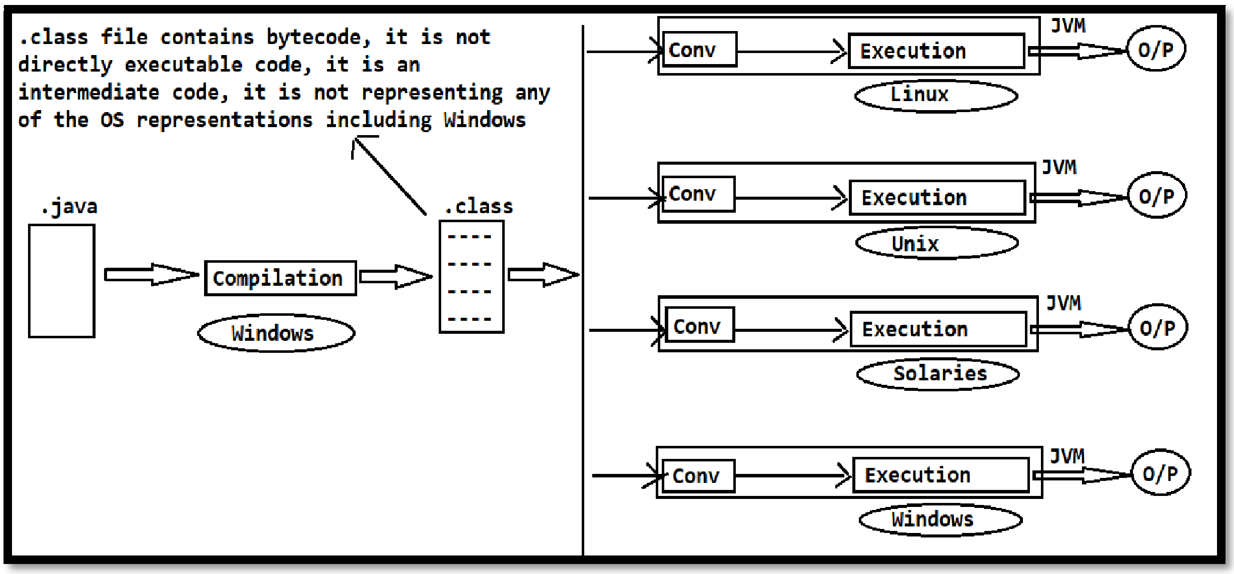
If any PL allows its applications to perform compilation and execution on the same Operating System then that PL is called as Platform Dependent PL.

<u>EX:</u> C and C++


.exe file contains directly executable code, it is generated as per Windows OS representations only.
C-Engine is able to execute Linux representations
.C
Compilation
Windows
.exe
0101
1010
0101
1010
C-Engine
Execution
~~Linux~~ Windows
O/P

If any PL allows its applications to perform Compilation is on one OS and execution is on another OS then that PL is called as Platform independent PL.

<u>EX:</u> JAVA


.class file contains bytecode, it is not directly executable code, it is an intermediate code, it is not representing any of the OS representations including Windows
.java
Compilation
Windows
.class
Conv
Execution
JVM
O/P
Linux
Conv
Execution
JVM
O/P
Unix
Conv
Execution
JVM
O/P
Solaries
Conv
Execution
JVM
O/P
Windows

Q)What are the differences between .exe file and .class file?

1. .exe file is available upto C and C++ only.
   .class file is available upto Java.

2. .exe file contains directly executable code.
   .class file contains bytecode, it is not executable code directly, it is an intermediate code.

3. .exe file is platform dependent file.
   .class file is platform independent file.

4. .exe file is less secured file.
   .class file is more secured file.

4)Pointers are existed in C and C++, but, Pointers are not existed in Java:

In general, in Progaming languages, Variables are able to store data.
EX: int eno = 111;

In C and C++, to manipulate data through memory locations , C and C++ have provided a type of varable called as "Pointer" variable.

Pointer is a variable in C and C++, it able to store address locations of the data structers, where Data Structer may be a variable, an array, a struct, or may be another pointer variable.

Q)What are the differences between pointer variables and referece variables?

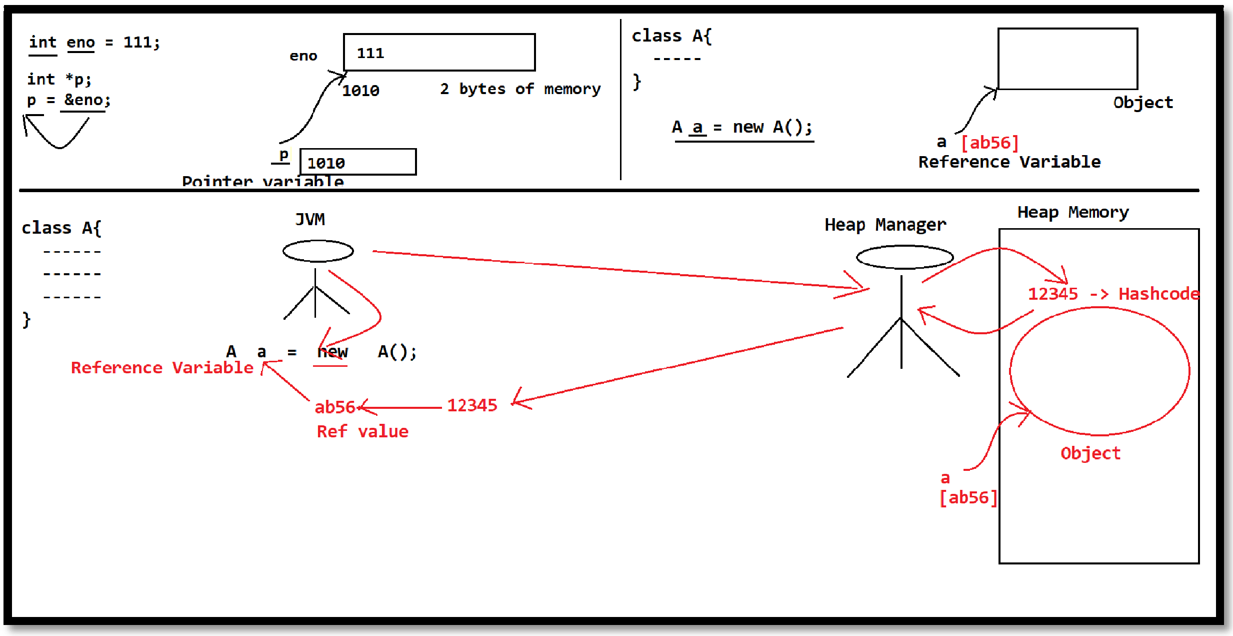
1. Pointer variables are available upto C and C++.
   Reference variables are available upto JAVA mainly.

2. Pointer variables are able to refer a block of memory by storing its address locations.

   Reference variables are able to refer a block of memory (Object) by storing object reference values, where Object reference value is hexa decimal form of hashcode, where hashcode is an unique identity provided by Heap manager.

3. Pointer variables are recognized and initialized at compilation time.

4. Reference variables are recognized and initialized at runtime.



5)Multiple inheritance is not possible in Java:

If any PL allows to represent data in the form of Objects as per Object Oriented principles[Features] then that PL is called as Object Oriented PL.

In general, there are 7 Object Oriented principles or Features.
1. Class
2. Object
3. Encapsulation
4. Abstraction
5. Inheritance
6. Polymorphism
7. Message Passing

From the above 7 Object Oriented Features, the following features are most powerfull features.

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

Inheritance: inheritance is relation[Parent-Child] between classes,it will bring variables and methods from one class[Super class / Base Class / Parent Class] to another class[Sub class / Derived Class / Child Class] inorder to reuse.

The main advantage of Inheritance is "Code Reusability".
Note: In Java, Inheritance is represented in the form of "extends" keyword.

EX:
```
class Employee{ // Super class
        int eno;
        String ename;
        float esal;
        String eaddr;
        ---
        ----
}

class Manager extends Employee{// Sub Class
        ---Reuse Employee class members here-----
        ----
        ----
}

class Accountent extends Employee{// Sub class
        ---Reuse Employee class members here-----
        ------
        ------
}
```

Initially, there are two types of Inheritances.

1. Single Inheritance
2. Multiple Inheritance

On the basis of the above two types of inheritances, three more Inheritances are defined.

3. Multi Level inheritance.
4. Hierarchical Inheritance.
5. Hybrid Inheritance.

1. Single Inheritance: It is a relation between classes, where it will bring variables and methods from only one super class to one or more no of sub classes.

Java does support Single Inheritance.

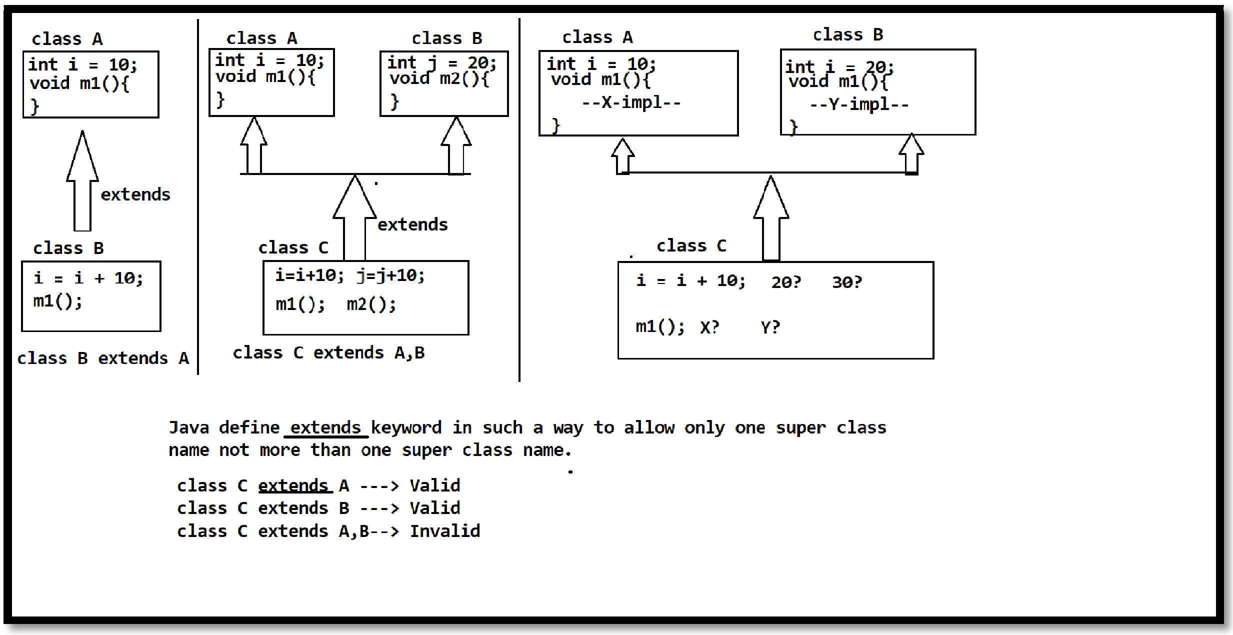
2. Multiple Inheritance: It is arelation between classes, where it will bring variables and methods from more than one super class to one or more no of sub classes.

Java does not support Multiple Inheritance.

If we declare same variable with different values and same method with different implementatiuon in both the super classes and if we access that common variable and method at sub class then which super class variable and method will be accessed is big confusion for the compiler and JVM, JAVA is simple pl, it will not allow any confusion oriented features, so JAVA does not allow Multiple Inheritances.

Note: Java define "extends" keyword in such a way to allow only ne super class name , not to allow more than one super class name.

class A extends B{ } ---> Valid
class A extends C{ } --> Valid.
class A extends B, C { } ---> invalid.



```
Java define extends keyword in such a way to allow only one super class
name not more than one super class name.

 class C extends A ---> Valid
 class C extends B ---> Valid
 class C extends A,B--> Invalid
```

6)Destructors are required in C++, but, Destructors are not required in JAVA:

If any PL allows to represent data in the form of Objects as per Object oriented Features then that PL is called as "Object Oriented Programming Language".

In Object oriented Programming Languages, it is minimum to represent data in the form of objects , to repersent data in the form of objects , first, we have to create objects and at end of the program we have to destroy that objects.

In Object Oriented PLs, two operations we have to perform frequently.
1. Creating Objects
2. Destroying Objects

To create Objects in Object Oriented Programming Languages, Object oriented Programming Languages have given a feature called as "Constructor".

To Destroy objects in Object Oriented Programming Languages, Object Oriented Programming Languages have a given a seperate feature called as "Destructor".

In Java, to destroy objects automatically , JAVA has provided an internal component called as "Garbage Collector", it will destroy unused objects in Java applications.

In Java, Garbage Collector is respnsible for Destroying objects, Developer is not responsible to destroy objects, so Developers are not required to use Destructors", so that, Destructors are not reqwuired in JAVA.

In C++, Developers are responsible to destroy Objects, because, in C++ Garbage Collector kind of component is not existed, Therefore Developers must use "Destructors" todestroy objects.

7) Operator Overloading is not supported in Java:

When a PL represents data in the form of Objects then that PL is called as an Object Oriented Programming Language.

In Object Oriented Programming Languages , while rpresenting data in the forom of Objects we have to follow a set of conventions called as "Object oriented Features".

1. Class
2. Object
3. Encapsulation
4. Abstraction
5. Inheritance
6. Polymorphism
7. Message Passing

From the above list of Object oriented Features the following features are the most powerfull features.

1. Encapsulation
2. Abstration
3. Inheritance
4. Polymorphism

Polymorphism:

--> "Polymorphism" is a Greak word, where Poly means Many and Morphism means Structers / Forms.
--> If one thing is existed in more than one form then it is called as "Polymorphism".
--> The main advantage of Polymorphism is "Flexibility".
--> There are two types of Polymorphisms.

1. Static Polymorphism
2. Dynamic Polymorphism

1. Static Polymorphism:

--> If the Polymorphism is existed at compilation time then it is called as Static Polymorphism.
EX: Overlooading

2. Dynamic Polymorphism:

--> If the Polymorphism is existed at runtime then that Polymorphism is called as Dynamic Polymorphism.
EX: Overridding

Overloading:

--> There are two types of overloadings.

1. Method Overloading
2. Operator Overloading

1. Method Overloading:

--> If we declare more than one method / Function with the same name and with different parameter list then it is
called as Method Overloading.
EX:

```
class A{
	void add(int i, int j){
		---implementation integer addition----
	}
	void add(float f1, float f2){
		---implementation for Float Addition----
	}
	void add(String str1, String str2){
		---Concatination on two String values----
	}
}
```

2. Operator Overloading:

--> If we declare any operator with more than one functionality then it is called as Operator Overloading.

EX:

```
int a = 10;
int b = 20;
int c = a + b; // + is for Arithmetic Addition.
System.out.println(c); //30

String str1 = "abc";
String str2 = "def";
String str3 = str1 + str2;// + is for String concatination
System.out.println(str3);//abcdef
```

In Java Operator overloading is not possible , because,

1. Operator overloading is a rarely used feature in Java application development.
2. It is a bit confusion oriented feature when we define more no of operations for more no of operators.

Note: As per JAVA internal requirement, JAVA has defined some of the predefined operators as overloaded Operators with fixed functionalities, but, JAVA has not given any env to define operator overloading explicitly at developer level.

EX: +, *, %,..

8)C and C++ are following Call By Value and Call by reference parameter passing mechanisms, but, JAVA is following only call by value parameter passing mechanism:

In any PL, if we pass primitive data[int, long, float, double, boolean, char,...] as parameter to methods or functions then the parameter passing mechanism is "Call By Value".

In any PL, if we pass address locations as parameters to the methods then the parameter passing mechanism is "Call By Reference" Parameter Passing Mechanism.

In C and C++, if we pass pointer variables as parameters to the methods or functions then the parameter passing mechanism is "Call By Reference", because, Pointer variables are able to store address locations.

In Java, if we pass reference variable as parameter to the methods or functions then the parameter passing mechanism is "Call By Value" only, because, in JAVA, reference variables are not storing address locations, reference variables are able to store Object reference value, where Object reference value is hexa decimal form of Hashcode, where Hashcode is an integer value provided by Heap manager as an unique identity for each and every object.

9.In C and C++ , integers will take only 2 bytes of memory and characters will take 1 byte of memory, but, in JAVA integers will take 4 bytes of memory and characters will take 2 bytes of memory:

In C and C++, memory allocation for primitive data types is not fixed, it is variable and it is depending on the Operating System which we used.

In Java, memory allocation for the primitive data types is fixed irrespective of the Operating System which we used.

1. byte ----------> 1 byte
2. short ----------> 2 bytes
3. int ------------> 4 bytes
4. long -----------> 8 bytes
5. float ----------> 4 bytes
6. double ---------> 8 bytes
7. char -----------> 2 bytes
8. boolean --------> 1 bit

Q) In C and C++, to store character value one byte of memory is sufficient, then , what is the requirement for java to assign 2 bytes of memory for character data?

In C and C++, all characters are repesented in the form of ASCII values, where to store any ASCII value 1 byte of memory is sufficient.

In Java, all characters are represented in the form of UNICODE values, where to store UNICODE values one byte of memory is not sufficient, we must provide 2 bytes of memroy for characters.

Q) What is UNICODE and what is the adv of UNICODE rerpesentation in Java?

Ans:

UNICODE is one of the character reprsentation in Programming Languages, It able to rerpesent all the alphabet from all the natural languages like English, Hindi, Italian, Chinees,......and it able to provide very good Internationalization support in Java applications.

Q) What is Internationlization in Java?

Ans:

Internationalization is a service in Java, it enables Java applications to take input data in a particulr local language and to provide output data in the same local language.

--> I have a software product, it has customers all over India, All Indians are able to understasnd English, so i provided all the product Services in the form of English. One fine day, i found Customers from Japan, Germany, Itly,..... and these customers are not good in English and they are not undestanding my product services .

Solutions

1. Can i prepare a seperate Product for each and every customer language ---> Not Suggestible.
2. Single product, but, it has to understand Customer Locality and it has to give services as per customer locality
in customer understandable language.

To achieve 2 requirement we have to use "Internationalization".

Dersigning Java applications as per Local Conventions is called as "Internationalization".

<u>Note:</u> Java isa able to provide very good Internationalization support because of UNICODE representation only.

Java Features

To show the nature of java programming language, JAVA has provided the following features.

1) Simple
2) Object Oriented
3) Platform independent
4) Arch Nuetral
5) Portable
6) Robust
7) Secure
8) Dynamic
9) Distributed
10) Multi Threadded
11) Interpretive
12) High Performance

1) Simple:

Java is simple programming language, because,

1) Java applications will take less memory and less execution time.
2) Java has removed all most all the confusion oriented features like pointers, multiple inheritance,.....
3) Java is using all the simplified syntaxes from C and C++.

2) Object Oriented:

Java is an object oriented programming language, because, JAVA is able to store data in the form of Objects only.

3) Platform Independent:

Java is platform independent programming Language, because, Java allows its applications to compile on one operating system and to execute on another operating system.

4) Arch Nuetral:

Java is an Arch Nuetral Programming language, because, Java allows its applications to compile on one H/W Arch and to execute on another H/W Arch.

5) Portable:

Java is a portable programming language, because, JAVA is able to run its applications under all the operating systems and under all the H/W Systems.

6) Robust:

Java is Robust programming language, because,

1) Java is having very good memory management system in the form of heap memory Management SYstem, it is a dynamic memory management system, it allocates and deallocates memory for the objects at runtime.

2) JAVA is having very good Exception Handling mechanisms, because, Java has provided very good predefined library to represent and handle almost all the frequently generated exceptions in java applications.

7) Secure:

Java is very good Secure programming language, because,

1) JAVA has provided an implicit component inside JVM in the form of "Security Manager" to provide implicit security.

2) JAVA has provided a seperate middleware service in the form of JAAS [Java Authetication And Autherization Service] inorder to provide web security.

3) Java has provided very good predefined implementations for almost all well known network security alg.

8) Dynamic:

If any programming language allows memory allocation for primitive data types at RUNTIME then that programming language is called as Dynamic Programming Language.

JAVA is a dynamic programming language, because, JAVA allows memory allocation for primitive data types at RUNTIME.

9) Distributed:

By using JAVA we are able to prepare two types of applications

1) Standalone Applications
2) Distributed Applications

1) <u>Standalone Applications:</u>

If we design any java application with out using client-Server arch then that java application is called as Standalone application.

2) **Distributed Applications:**

If we design any java application on the basis of client-server arch then that java application is called as Distributed application.
To prepare Distributed applications, JAVA has provided a seperate module that is "J2EE/JAVA EE".

10) Multi Threadded:

Thread is a flow of execution to perform a particular task.

There are 2 thread models

1) Single Thread Model
2) Multi Thread Model

1) **Single Thread Model:**

It able to allow only one thread to execute the complete application,it follows sequential execution, it will take more execution time, it will reduce application performance.

2) **Multi Thread Model:**

It able to allow more than one thread to execute application, It follows parallel execution, it will reduce execution time, it will improve application performance.

JAVA is following Multi Thread Model, JAVA is able to provide very good environment to create and execute more than one thread at a time, due to this reason, JAVA is Multi threaded Programming Language.

11) Interpretive:

JAVA is both compilative programming language and Interpretive programming language.

1) To check developers mistakes in java applications and to translate java program from High level representations to low level representation we need to compile java programs
2) To execute java programs , we need an interpretor inside JVM.

12) High Performance:

JAVA is high performance programming language due to its rich set of features like Platform independent, Arch Nuetral, Portable, Robust, Dynamic,......

JAVA Naming Conventions

Java is a case sensitive programming language, where in java applications, there is a seperate recognization for lower case letters and for upper case letters.

To use lower case letters and Upper case letters seperatly in java applications, JAVA has provided the following conventions.

All class names , abstract class names, interface names and enum names must be started with upper case letter and the sub sequent symbols must also be upper case letters.

EX:
String
StringBuffer
InputStreamReader

All java variables must be started with lower case letters, but, the subsequent symbols must be upper case letters.
EX:
in, out, err
pageContext, bodyContent

All java methods must start with lower case letter , but, the sub sequent symbols must be upper case letters
EX:
concat(--)
forName(--)
getInputStream()

4.All java Constant variables must be provided in Upper Case letters.
EX:
MIN_PRIORITY
NORM_PRIORITY
MAX_PRIOITY

5.All java package names must be provided in lower case letters
EX:
java.util
java.lang.reflect
javax.servlet.jsp.tagext

Note: All the above conventions are mandatory for predefined library, they are optional form User defined library, but, suggestible.

EX:
```
1)String str=new String("abc"); ----> Valid
2)string str=new string("abc");-----> Invalid
3)class Employee{ ----> Valid and Suggestible
  ---
 }

4)class student{ ---> valid, but, not suggestible
  ---
 }
```

Java Programming Format

To prepare basic Java application we have to use the following Structer.

1) Comment Section
2) Package Section
3) Import Section
4) Classes/Interfaces Section
5) Main Class Section

1.Comment Section:

Before starting implementation part, it is convention to provide some description about our implementation, here to provide description about our implementation we have to use Comment Section. Description includes author name, Objective, project details, module details, client details,......

To provide the above specified description in comment section, we will use comments.

There are 3 types of comments.

1) Single Line Comments
2) Multi Line Comments
3) Documentation Comments.

1.Single Line Comment:

It allows the description with in a single line.

Syntax:
```
// --- description------
```

2.Multi Line Comment:

It allows description in more than one line

Syntax:

```
/*
---
--description-----
----
*/
```

3.Documentation Comment.

It allows description in more than one page.

Syntax:

```
/*
*----
*----
---
---
*----
*/
```

Note: We will use documentation comments to prepare API kind of documentations, but, it is not suggestible.

API kind of documentation:

It is a document in the form of .txt file or .doc file or .pdf file or .html it includes the complete declarative information about our programming elements like variables, methods, classes,..... which we have used in our java file.

EX: Employee.java

```
public class Employee extends Person implements Serializable, Cloneable {
                puiblc String eid;
                public String ename;
                public float esal;
                public String eaddr;
                public Employee (String eid, String ename, float esal, String eaddr) {
                }
                public Employee (String eid, String ename, float esal) {
                }
                public Employee (String eid, String ename) {
                }
                public void add(String eid, String ename, float esal, String eaddr) {
                        ----
                }
                public String search(String eid) {
                        return "success";
```

```
            }
            publc void delete(String eid) {
                ---
            }
}
```

Employee.txt[html/pdf]

```
Class : Name: Employee
        Super Class: Person
        Interfaces: Serializable, Cloneable
        Variables: 1) Name: eid
                      DataType: String
                      Access Mod: public

                   2)Name: ename
                     Data Type: String
                     Access Mod: public
                     ----
                     ----
        Methods:  1)Name: add
                      Return type: void
                      access mod: public
                      parameters: eid, ename, esal, eaddr
                      ------
        Constructors: 1)Employee
          ----
```

To simpify API documentation for JAVA applications, JAVA has provided an implicit command , that is, "javadoc".

EX: D:\javaapps\ Employee.java

```
public class Employee implements java.io.Serializable, Cloneable {
   public String eid;
   public String ename;
   public float esal;
   public String eaddr;
   public Employee(String eid, String ename, float esal, String eaddr) {
   }
   public Employee(String eid, String ename, float esal) {
   }
   public Employee(String eid, String ename) {
   }
          public void add(String eid, String ename, float esal, String eaddr) {
          }
          public String search(String eid) {
```

```
                   return "success";
            }
            public void delete(String eid) {
            }
}
```

On Command Prompt:
D:\javaapps>javadoc Employee.java
 --- Generating xxx.html files----

To provide description[Metadata] in java programs, JDK5.0 version has provided a new feature that is "Annotations".

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

If we provide description along with comments in java program then "Lexical Analysis" phase will remove comments and their descriotion which we provided in java program as part of Compilation.
As per the requirement,if we want to make available our description upto .java file, upto .class file and upto RUNTIME of our applications there we have to use "Annotations".

<u>Note:</u> If we provide metadata with comments then we are unable to access that metadata programatically, but, if we provide metadata with Annotations then we are able to access that metadata through java program.

Q)In java applications, to provide metadata at RUNTIME we are able to use XML documents then what is the requirement to use "Annotations".

If we use XMl documents to provide description then we are able to get the following problems.

1) We have to learn XML tech.
2) Every time we have to check whether XML documents are located properly or   not.
3) Every time, we have to check whether XML documents are formatted properly or   not.
4) Every time we have to check whether we are using right parsing mechanisms   or not to read data from XML documents.

To overcome all the above problems we need a java alternative , that is, Annotations.

<u>Note:</u> IN JAVA/J2EE applicatinos, we can utilize Annotations as an alternative to XML documents.

XML Based Tech	Annotation Based Tech [XML documents Optional]
1) 1.Upto JDK1.4	1) JDK5.0 and above
2) 2.Upto JDBC3.0	2) JDBC4.0
3) 3.Servlets2.5	3) Servlets3.0
4) 4.Struts1.x	4) Struts2.x
5) 5.JSF1.x	5) JSF2.x
6) 6.EJBs2.x	6) EJBs3.x
7) 7.Spring2.x	7) Spring3.x
-----	-----

2)Package Section:

1.Package is the collection of related classes and interfaces as a single unit.
2.Package is a folder contains .class files representing related classes and interfaces.

Packages are able to provide some advantages in java applications

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

There are two types of packages in java

1) Predefined Packages
2) User defined Packages

1.Predefined Packages:

These packages are provided by Java programming language along with java software.

EX: java.io
java.util
java.sql

2.User defined Packages:

These packages are defined by the developers as per their application requirements.
Syntax:
package package_Name;
where package name may be
1.directly a single name
2.sub package names with . operator

EX:
package p1;
package p1.p2.p3;

If we want to use package declaration statement in java files then we have to use the following two condition.
1.Package declaration statement must be the first statement in java file after the comment section.
2.Package name must be unique, it must not be sharable and it must not be duplicated.

Q)Is it possible to declare more than one package statement with in a single Java file?

No, it is not possible to declare more than one package declaration statement with in a single java file, because, package declaration statement must be first statement, in java files only one package declaration statement must be provided as first statement.

abc.java

```
package p1;---> Valid
package p2;---> Invalid
package p3;---> INvalid
--
---
```

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

EX: www.durgasoft.com
durgasoft.com
com.durgasoft
package com.durgasoft.icici.transactions.deposit;
com.durgasoft---> company domain name in reverse.
icici ----------> project name/ client name
transactions----> module name
deposit---------> sub module

3.Import Section:

The main intention of "import" statement is to make available classes and interfaces of a particular package into the present JAVA file inorder to use in present java file.

Syntax 1:
import package_Name.*;
--> It able to import all the classes and interfaces of the specified package into the present java file.

EX: import java.io.*;

Syntax 2:
import package_Name.Member_Name;
--> It able to import only the specified member from the specified package into the present java file.

EX: import java.io.BufferedReader;

Q)Is it possible to write more than one "import" statement with in a single Java file?

Yes, In a single java file, we are able to provide atmost one package declaration statement, but, we are able to provide any no of import statements.

abc.java

```
package p1;
//package p2;-----> Error
//package p3;-----> Error
import java.io.*;
import java.util.*;--> No Error
import java.sql.*;---> No Error
---
----
---
```

Q)Is it possible to use classes and interfaces of a particular package with out importing that package?

Yes, it is possible to use classes and interfaces of a particular package in the present java file with out importing the respective package, but, just by using fully qualified names of the classes.

Note: Specifying class names or interface names along with their respective package names is called as "Fully Qualified Names".

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 with out import statement:
java.io.BufferedReader br=new java.io.BufferedReader(new java.io.InputStreamReader(System.in));

4.Classes/Interfaces Section:

The main intention of classes and interfaces is to repersent all real world entities in the form of coding part.
EX: Account, Employee, Product, Customer, Student,......

Note: No restrictions for no of classes in a java file or in a java application, depending on the application requirement, we are able to write any no of classes and interfaces in java applications.

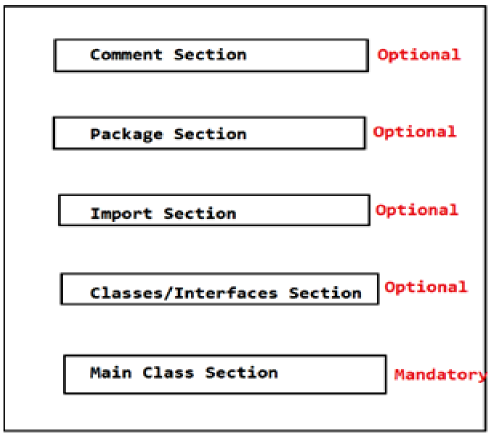
5.Main Class Section:

Main Class is a java class ,it includes main() method.

The main intention of main() method is,
1.To manage application logic which we want to execute by JVM directly we have to use main() method.
2.To define starting point and ending point to the application execution we have to use main() method.

Syntax:
```
public static void main(String[] args){
----instructions----
}
```

Note:main() method is a conventional method with fixed prototype and with user defined implementation part.


Comment Section
Optional
Package Section
Optional
Import Section
Optional
Classes/Interfaces Section
Optional
Main Class Section
Mandatory