JAVA INTRODUCTION

JAVA

- Java is a general purpose, high level, object oriented programming language
- Java is an example of OOPS language
- Write Once Run Anywhere (WORA) language. This means that, the compiled java code can be run on any platforms such as windows, linux, mac, etc, …
- Change of the platform does not affect the original java program
- It is mainly designed for web / internet applications
- Pure object oriented language
    - ♦ In java, it is not possible to write a code without using class and object

Java Language

Old Name	:	OAK
New Name	:	Java
Inventor	:	James Gosling
Organization	:	Sun Micro systems / Oracle
File Extension	:	.java
Compiler	:	.javac (java compiler)
Interpreter	:	.java (java interpreter)
IDE	:	Netbeans IDE, Eclipse IDE, IntelliJ IDEA, etc, …

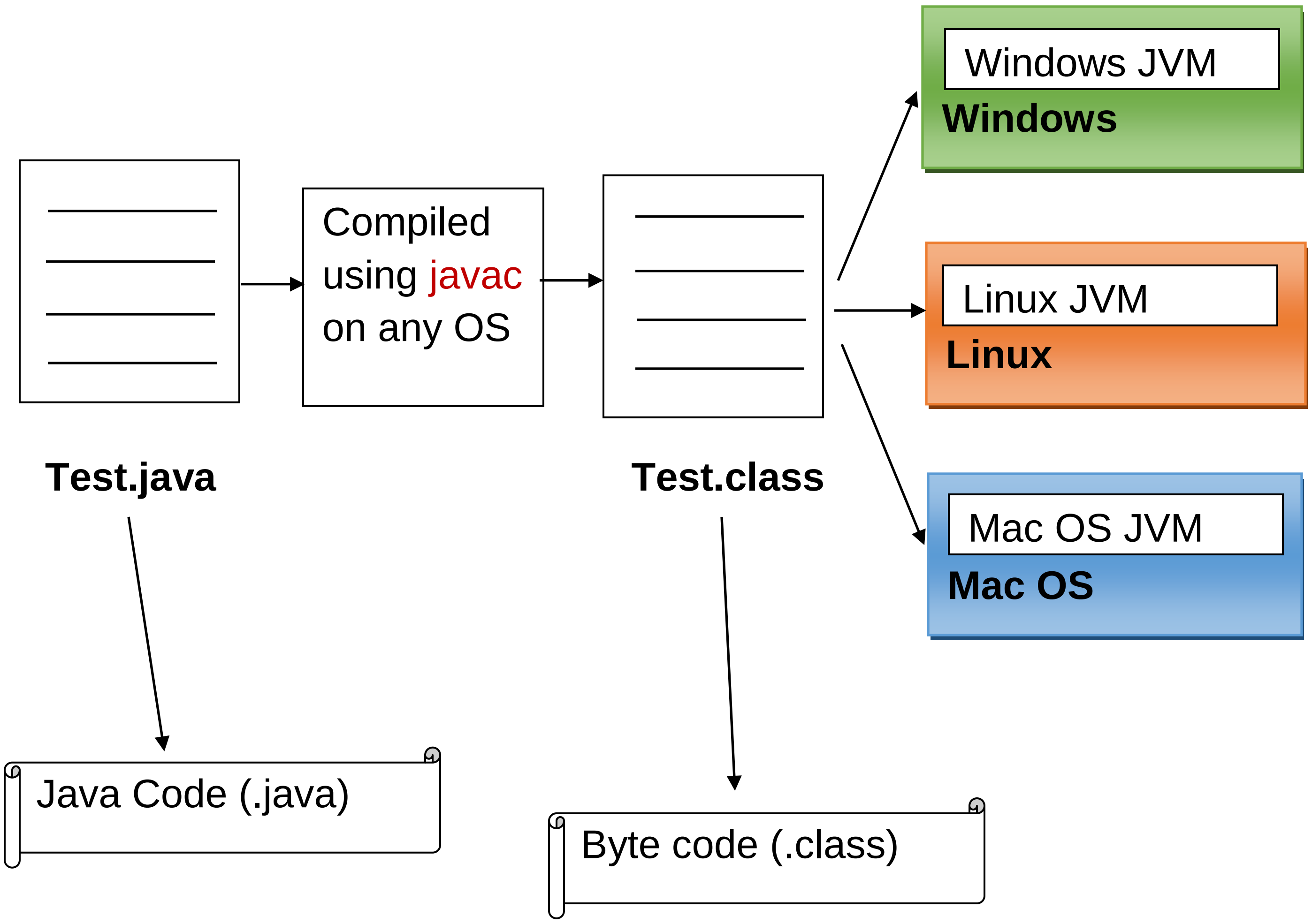
FEATURES

1. Simple

- Java is a simple and easy to learn, easy to understand
- Because
  - ♦ Its syntax is based on c++
  - ♦ It removed many complicated features for example: operator overloading, explicit pointers, etc, …

2. Platform Independent

- Java is a platform independent language that means, java is not dependent on any platform or OS
- It supports multiple platforms such as windows, linux, mac, etc, ..



3. Object Oriented

- Java is an object oriented programming language
- It supports the oops features such as class and objects, inheritance, polymorphism, encapsulation, etc, …
- Everything in java is an object (except java base data types: int, float, double, etc)

4. Interpreted and Compiled Language

- Unlike c/c++, java is a two-way language
- It supports both compilation and interpretation

5. Strongly Typed Language (Static Typed System)

- Java supports static typed system. Data types must be mentioned in the variable / method creation
- Ex. int k=10;

6. High Performance

- Java program is compiled into java byte code (.class file) which is highly optimized by the java compiler, so that the java virtual machine (JVM) can execute java applications / programs at full speed. (Byte codes are highly optimized)

7. Multithreading support

- Java supports multithreading concept. That means we can build applications with concurrent threads of activity.

8. Security

- Java gives best choice for security
- We can develop virus free systems with help of java secure features

9. Exception Handling

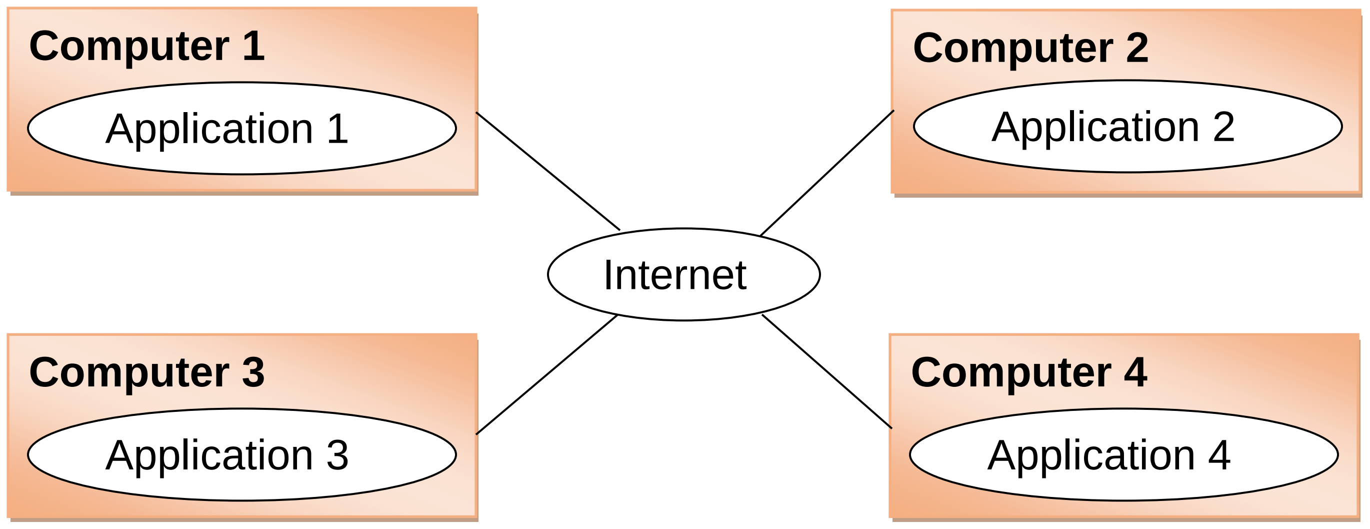
- Java handles run time errors with help of exception handling concept

10. Architecture Neutral

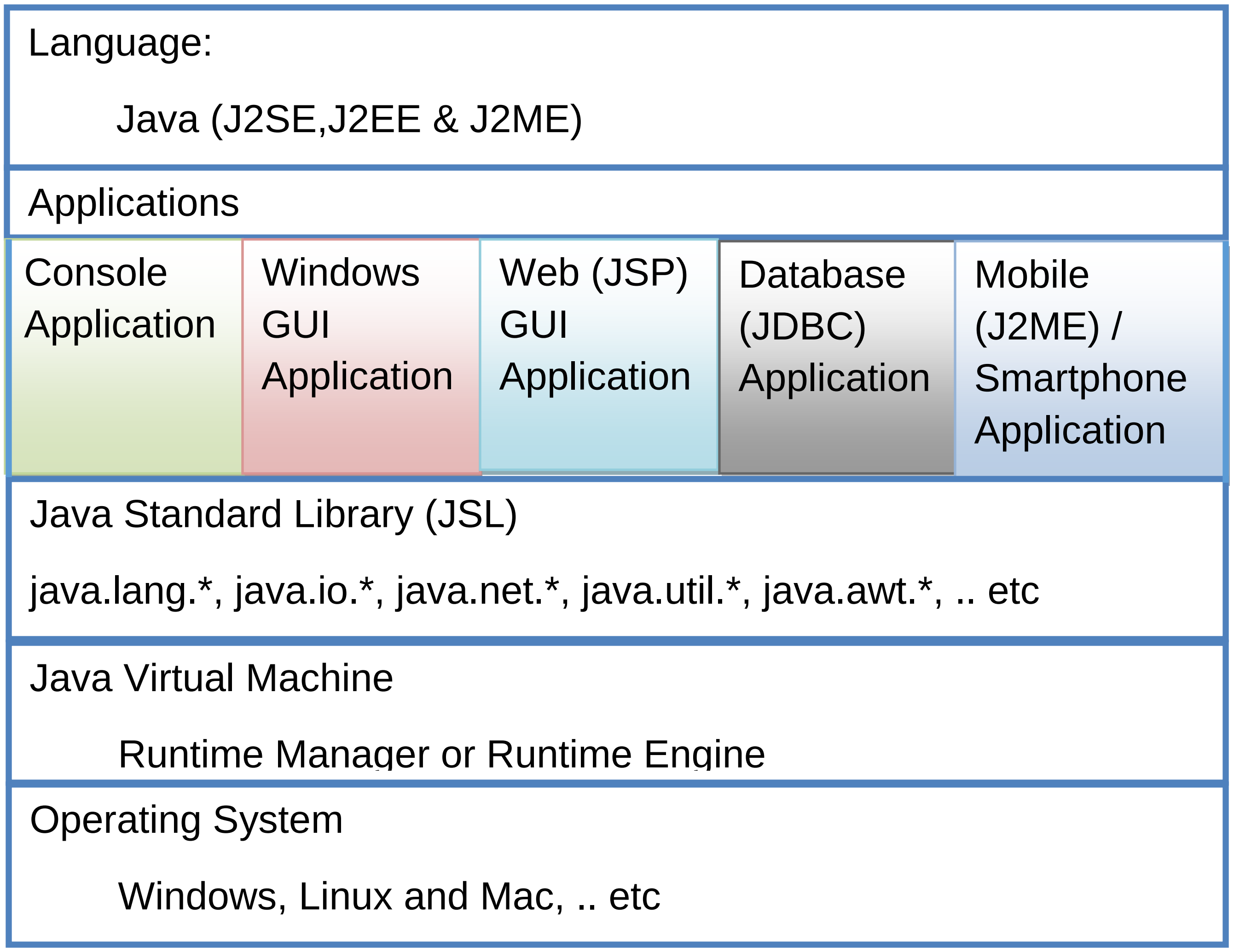
- Java is an architecture neutral language. Because it does not depend on architecture (organization of processor, memory, CPU, input/output) of a computer.
- Hence, java code can be run on any computer architecture.

11. Distributed

- Java is a distributed language. This means that, java programs running on one machine can easily access the resources (files, java objects, etc, …) of other machine on internet.
- It provides class libraries for high-level support of networking
- Remote method invocation(RMI) API's allow java programs to call methods of remote java objects, as if they were local objects
- RMI and EJB are mostly used for distributed programming.



JAVA ARCHITECTURE


Language:
Java (J2SE,J2EE & J2ME)
Applications
Console Application
Windows GUI Application
Web (JSP) GUI Application
Database (JDBC) Application
Mobile (J2ME) / Smartphone Application
Java Standard Library (JSL)
java.lang.*, java.io.*, java.net.*, java.util.*, java.awt.*, .. etc
Java Virtual Machine
Runtime Manager or Runtime Engine
Operating System
Windows, Linux and Mac, .. etc

Operating System

- Java is a platform independent OS
- Ex. Windows, Linux, Mac, etc, …

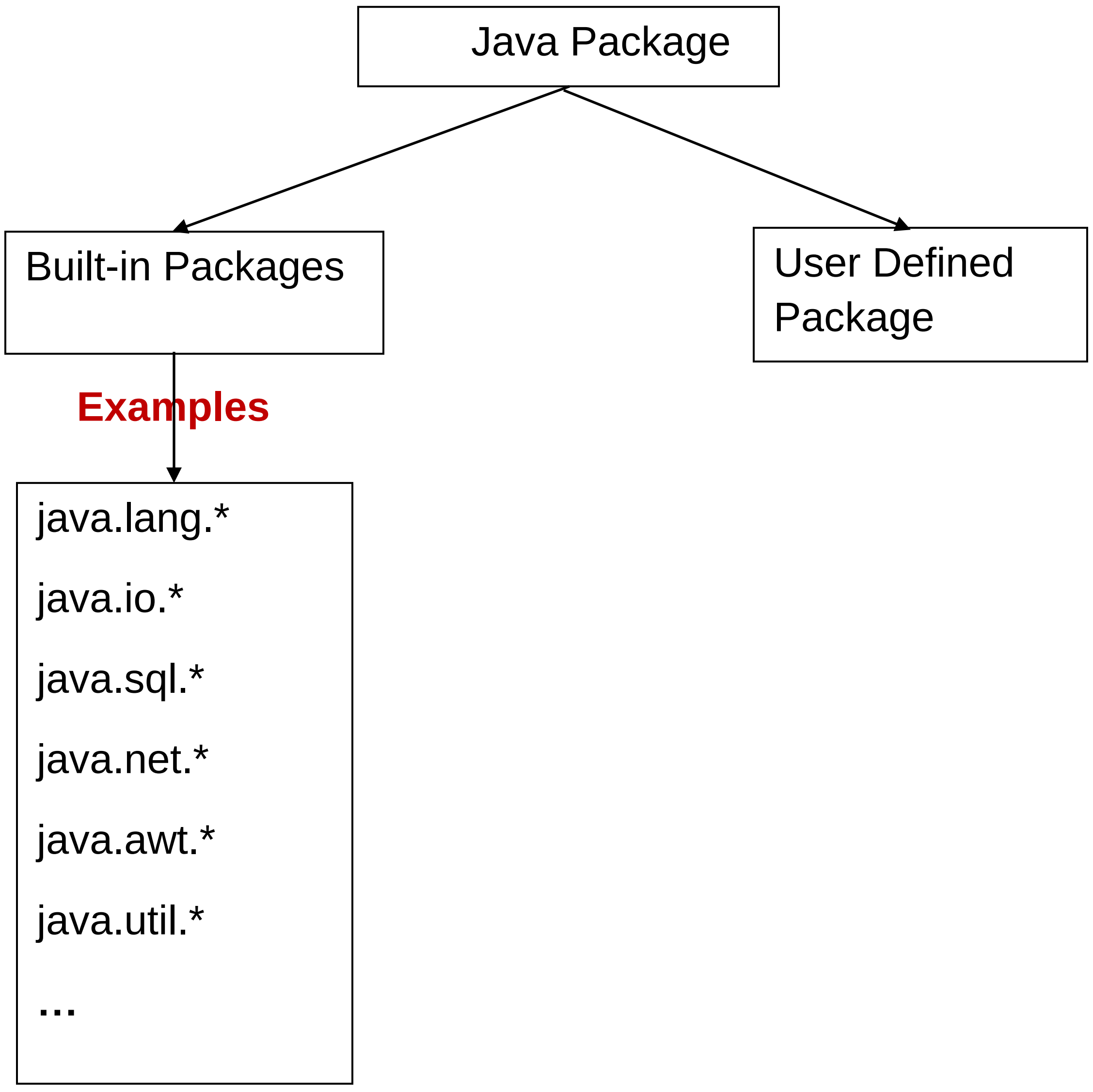
JVM (JAVA VIRTUAL MACHINE)

- Runtime manager or Runtime engine for Java
- Used to convert the byte code(.class) or object file or class file or machine code to target output

Java System Packages

- It is a part of java API (Application Programming Interface)
- Package is a group of classes, interfaces and sub packages
- Package is classified as two types in java. They are

  1. System packages / Built-in packages
     - Standard packages which comes as a part of Java Runtime Environment (JRE)
  2. User defined packages
     - Package defined by the user / programmer to bundle group of related classes



Note

- The operator ‘*’ is used load all classes and interfaces in a specified package

1. **java.lang.***
   - It is used for basic operations, mathematical operations, string features (String, StringBuffer, StringBuilder) thread operations in java
2. **java.io.***
   - This package is used for input & output related operations, File and Directory related operations
3. **java.util.***
   - This package provides the support for ArrayList, Vector, advanced data structures such as list, set, map interfaces
4. **java.sql.***
   - It is used for database related operations
     - Storage & retrieval
     - CRUD operations
5. **java.net.***
   - This package contains the predefined classes for supporting network related operations like TCP and UDP based applications
6. **java.awt.***
   - This package contains the predefined classes for implementing Windows GUI application / Desktop based applications
7. **java.applet.***
   - This package is used for creating applet (GUI) based applications

Java Applications

- Java supports 4+ different types of applications. They are:
    1. Console Application
    2. Windows GUI Application
    3. Web Application
    4. Database Application
    5. Smart Phone Application

1. Console Application

- Creating an application without using any graphical components, that is called as console application.

Input	: Java code (.java)
Output	: DOS Window / IDE based Window

2. Windows / Desktop Application (offline)

- Creating an application, with use of graphical components that is called as windows GUI Application (windows forms application)
- It is an offline application: possible to run the windows GUI application in a particular machine at a time
- Technologies: Applets, AWT, Swing, JavaFX, etc, …

Input	: Java code (.java)
Technology	: **Applet or AWT or Swing**
Output	: Java GUI Application (Applet | Frame | JFrame)

3. Web / Internet Application (online)

- Creating an application, with use of web graphical components that is called as Web GUI Application
- It is an online application: possible to run the web GUI application in all machine at a time via internet access
- Technologies: JSP, Servlet

Input	: Java code (.java) and JSP code (.jsp, .html, …)
Technology	: **JSP and Servlet**
Output	: HTML with GUI (Web browser window)

4. Database Application

- Creating a database related application using any type of java applications such as console, windows, web, etc, …
- Technology: JDBC

Input	: Java code (.java)
Technology	: **JDBC (Java Database Connectivity)**
Output	: DOS Window / Windows GUI / Web GUI

5. Mobile Application

- Java provides two types of mobile applications. They are entry level application, smart phone applications
- Technology: J2ME, Android

Input	: Java code (.java)
Technology	: **J2ME, Android**
Output	: Simulator or Emulator, Real Mobile Device

IDE Support

- Java programs can be created using either manual or automatic approaches such as IDE
- Popular IDEs
  - Netbeans IDE, Eclipse IDE
  - IntelliJ IDEA, etc, …

SOFTWARE REQUIREMENTS

• Language	:	Java
• JDK	:	JDK 1.0 – 11. 0.1
• Tool	:	Editor or IDE

Java Version History

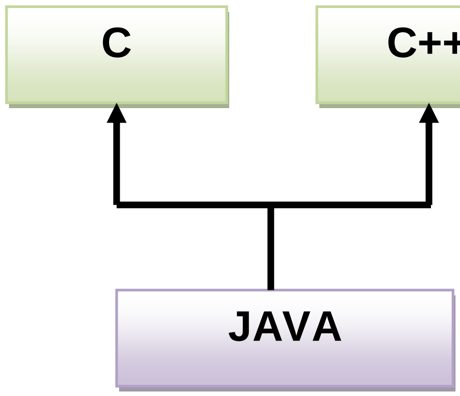
- Java includes many JDK versions. They are
  - JDK Alpha and Beta (1995)
  - JDK 1.0 (23rd Jan 1996)
  - JDK 1.1 (19th Feb 1997)
  - J2SE 1.2 (8th Dec 1998)
  - J2SE 1.3 (8th May 2000)

- J2SE 1.4 (6th Feb 2002)
- J2SE 5.0 (30th Sep 2004)
- Java SE 6 (11th Dec 2006)
- Java SE 7 (28th July 2011)
- Java SE 8 (18th March 2014)
- Java SE 9 (21st Sep 2017)
- Java SE 10 (20th March 2018)
- Java SE 11 (25th September 2018)

JAVA PLATFORM

- JVM and APIs (Application Programming Interface) together are referred as the "Java Platform"
- APIs
    - A collection of compiled code that can be included in a source program
    - used to give several ready-made solutions for our tasks that need to be frequently performed

FAMILY OF JAVA



- Java is derived from C and C++
- C gives the syntax and C++ gives the oops concepts

JAVA EDITIONS

- Java supports three different editions
  1. **J2SE** (Java to Standard Edition)
     - Supports core java features
  2. **J2EE** (Java to Enterprise Edition)
     - Supports enterprise technology
  3. **J2ME** (Java to Micro Edition)
     - Supports mobile applications

Flowchart

- Sequence of graphical symbols

Algorithm

- Set of steps

Program

- Set of instructions (implementation of algorithms)

DIFFERENCES BETWEEN C AND JAVA

S.N	C	JAVA
1.	File Name: (.c)	File Name: (.java)
2.	Pure structured oriented language (Procedural Language)	Pure object oriented language
3.	It supports pointer concept	- No pointer support - Java indirectly supports pointer via references
4.	It has keywords like **goto**, **typedef** and **sizeof()**	It does not support the keywords like **goto**, **typedef** and **sizeof()**
5.	The data types structure, union & enum are present	It does not support the user defined data types like structure, union & enum
6.	It has preprocessor	It does not support preprocessor such as #ifdef, #define, #include
7.	It follows Top down approach	It follows Bottom up approach
8.	It supports type modifiers like **auto, static, register & extern**	It does not support type modifiers
9.	It supports both call by value and call by reference	It supports only call by value concept
10.	IDEs: Code Blocks, DevCPP, NetBeans, Eclipse CDT	IDEs: NetBeans, Eclipse, IntelliJ IDEA
11.	C is a one way language system (it supports only compilation)	Java is a two way language system (It supports both compilation and interpretation)

DIFFERENCES BETWEEN C++ AND JAVA

S.N	C++	JAVA
1.	File Name: (.cpp)	File Name: (.java)
2.	**Partially object oriented language** 1. We can write a C++ code with or without using class 2. Because C++ is C with OOPs concepts	Pure object oriented language 1. In Java, it is not possible to write a code **without using class concept**
3.	It supports both function overloading and operator overloading	Java supports only function overloading
4.	It supports pointer concepts	It does not support pointer concepts
5.	C++ is derived from C (structures) and Simula67 (oops concepts)	Java is derived from C(syntax) and C++ (oops concept)
6.	It supports header file and global variable	It does not support header file & global variable
7.	C++ directly supports multiple inheritance	Java does not directly support multiple inheritance but indirectly it supports multiple inheritance by using interface concept
8.	It has destructor() function to destroy the memory occupied by the objects	It has finalize() function to remove the memory occupied by the objects
9.	It has template classes	It does not support template classes

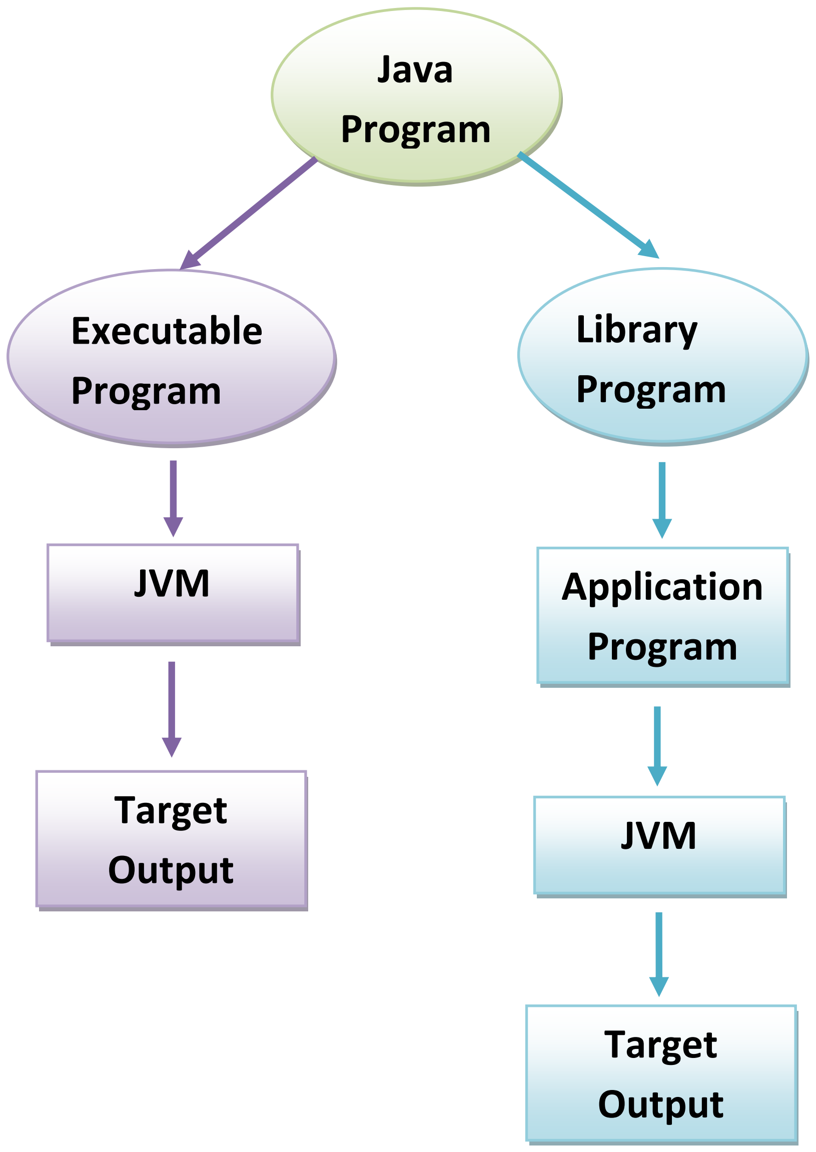
10.	It supports both call by value & call by reference	Java supports only call by value
11.	C++ is a one way language system (it supports only compilation)	Java is a two way language system (It supports both compilation and interpretation)

12.	Semicolon must be needed at the end of class program
	
	Ex.
	
	class Hello
	{
	   // code
	};
		Semicolon is not needed at the end of class program
		
		Ex.
		
		class Hello
		{
		   // code
		}

13.	In c++, object is a value type. So new modifier is not needed when creating an object in c++ class program **Ex.** class Test{…}; … Test obj; **// value type**	In java, object is a reference type. So **new modifier must be needed** when creating an object in java class program. **Ex.** class Test{…} … Test obj=new Test(); **// ref type**
14.	C++ is mainly used for system programming. It is lagging in Windows, Web & Mobile applications	Java is mainly used for Windows, Web & Mobile applications
15.	IDEs: Code Blocks, DevCPP, NetBeans, Eclipse CDT	IDEs: NetBeans, Eclipse, IntelliJ IDEA

OVERVIEW OF JAVA PROGRAMS

- Java can be used to develop two categories of programs, namely
  - (i) Executable Program (Direct Access)
  - (ii) Library Program (Indirect Access)



Executable Program

- It must have main method(). So no need to create end user or other application program to get the output
- It can be accessed directly
- Output: **.class, .jar**

Library Program

- It does not have main method(). So we should create an application to access this library program.
- It is very similar to c/c++ header files
- It can be accessed indirectly with help of end user application
- Output: **.dll**

S.N	EXECUTABLE PROGRAM	LIBRARY PROGRAM
1.	It must have Main()	It does not have main() method
2.	It can be accessed by directly	It can be accessed by indirectly with help of end user application
3.	Output: **.class, .jar**	Output: **.dll**

PRIMITIVE DATA TYPES

- Data types plays an important role in the definition of variables
- Java supports 8 standard data types

S.N	DATA TYPE IN JAVA	SIZE	SYMBOLS	DEFAULT VALUS
1.	boolean	1 bit		false
2.	byte	1 byte		0
3.	char	2 bytes		\u0000
4.	short	2 bytes		0
5.	int	4 bytes		0
6.	long	8 bytes	l or L	0
7.	float	8 bytes	f or F	0.0
8.	double	8 bytes	d or D	0.0

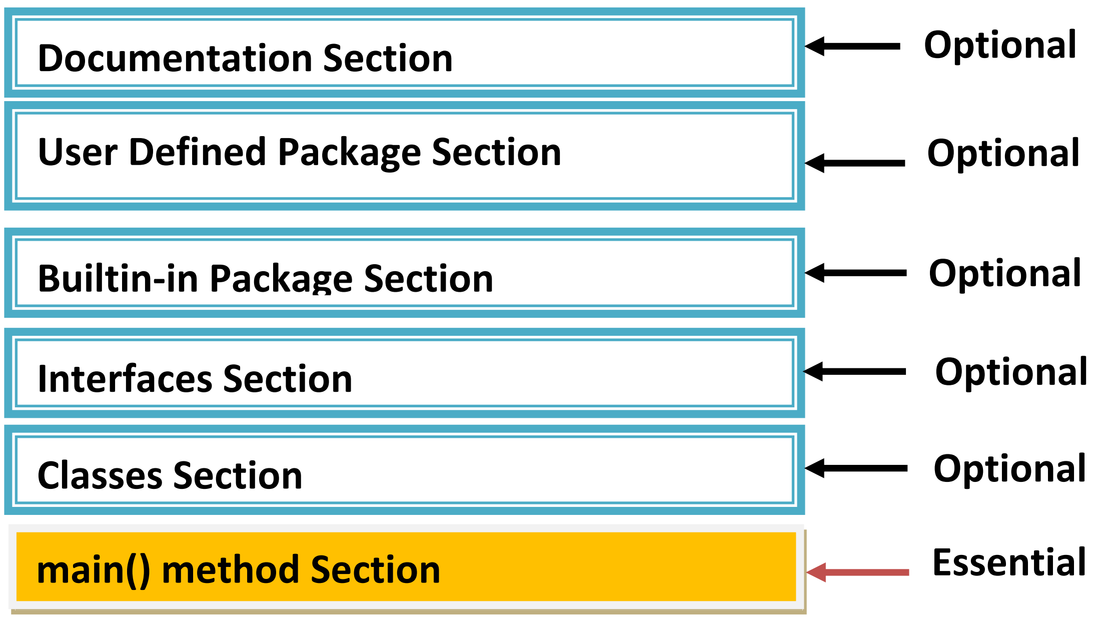
NOTE

- In java, character takes 2 bytes of memory. Because java uses Unicode character system not ASCII system

DEFAULT VALUES OF VARIABLES

S.N	TYPE	DEFAULT VALUE
1.	All integer types **e.g:** short, int, long	0
2.	char	'\x000'
3.	float	0.0f
4.	double	0.0d
5.	boolean	false
6.	All reference types e.g: **string, object**	null

JAVA PROGRAM STRUCTURE



(1) DOCUMENTATION SECTION

- It is an optional
- Gives the complete information about program like name of the program, author name, date, version & other details, etc
- Like c/c++, java supports two comment type statements

1. **Single Line Comment**
   - It is used to ignore only one statement at a time
   - It starts with operator like //

Example

```
// Good Morning
```

2. **Multiple Line Comments**
   - It is used to ignore more than one statements at a time
   - This is done by using the operator like /* … */

Example

```
/*
        Good Morning
        Welcome to chennai
        Hello
*/
```

(2) USER DEFINED PACKAGE SECTION

- Here we can create our own package
- Package is a set of classes / interfaces
- It is just like a container

Syntax

<package> <user-defined name>;

Example

IT.java

```
package MIT;
public class IT
{
        // code
}
class CC
{
        // code
}
…
class N
{
        // code
}
```

NOTE

- It is important to note that, class name and file name must be same in java

(3) PREDEFINED PACKAGE SECTION

- It is an optional
- It is used to load the predefined java libraries using "**import**" keyword
- It is used to create different types of applications, algorithms, tools & OS, etc
- **Ex.** java.lang.*, java.io.*, java.util.*, etc

Syntax

<import> <package.subpackage.*>;

<import> <package.subpackage.Class>;

Example

```
import java.sql.*;              // load all classes from sql package
import java.io.*;               // load all classes from io package
import java.io.DataInputStream;          // load only specific class
```

import

- It is a **reserved keyword** in java
- It is mainly used to **load the system packages** in java

(4) INTERFACES & CLASSES SECTION

Interfaces

- It has only declaration, no body (no definition)
- All the methods in the interfaces are public by default
- All the variables in the interfaces are const by default
- Interfaces are defined using interface (reserved word)
- It has no constructor

Syntax

```
< interface> <user-defined name>
   {
          // variable initialization
          // method declaration, no definition
   }
```

Example

```
interface ATM
{
        double balance = 5300;          // declaration + definition
        void details();                 // method declaration only
}
```

NOTE

- If the sub class inherits the interface, then the sub class must implement the methods of interface, else sub class also considered as an interface type.

Classes

- Collection of variables & methods are grouped together into a single entity (called class)
- classes are default or friendly modifier by default
- All the variables and methods in the class is default modifier by default
- Classes are defined using class (reserved word)
- It supports constructors (parameter & parameter less)

Syntax

```
<modifier >< class> <user-defined name>
{
        // variable initializations
        // method definitions
}
```

Example

```
class atm
{
        void get_balance()   { … }
        void deposit()       { … }
        void withdrawl()     { … }
        void pinchange()     { … }
}
```

(5) main() METHOD SECTION

- It is essential part in program
- It is an entry point of any java program
- It starts the program execution
- main() method accepts only void and int type. Other types are not allowed.

Example

```
public static void main(String[] arr)
{
      System.out.println("Good Morning...");
}
```

NOTE

- It is an important to note that, main() method does not return any type except void (It does not return any value)
- Main() method must be void type