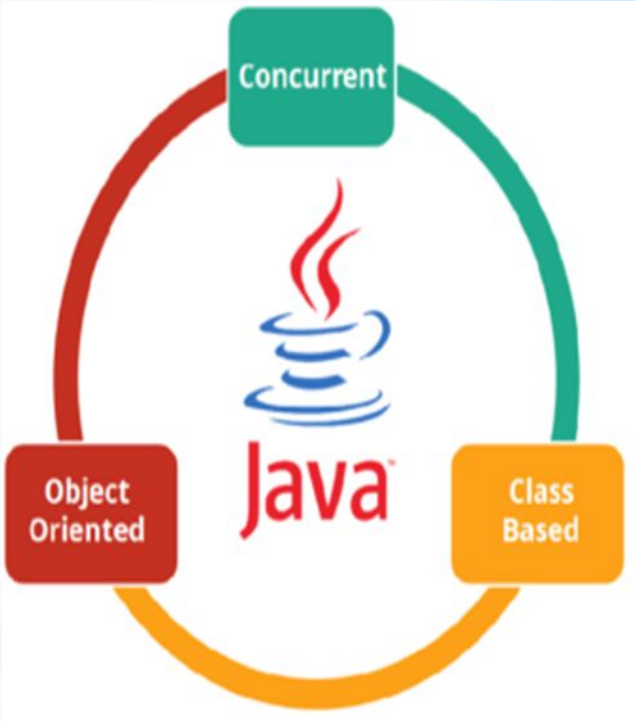
Unit 1
Introduction to java

Introduction To Java

- Java is an **object-oriented, class oriented, concurrent, secured & general-purpose** computer language.

- It is widely used **Robust** Technology.

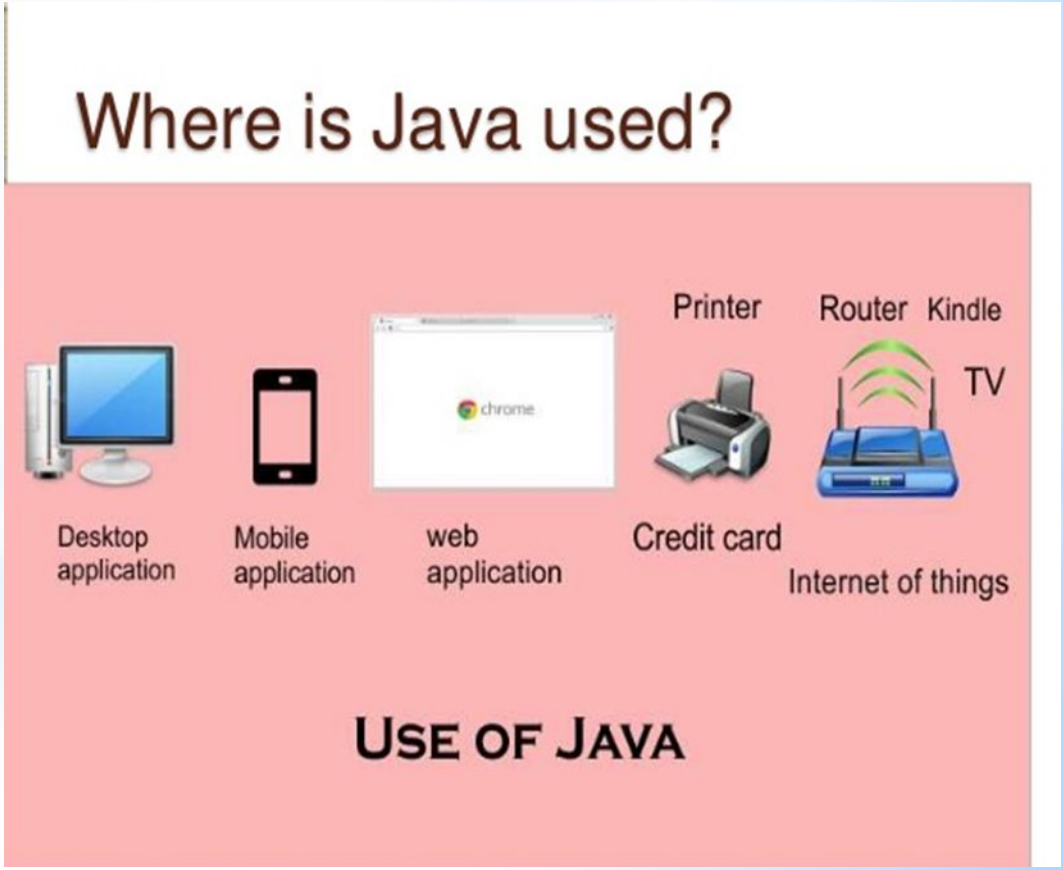


What is java ?

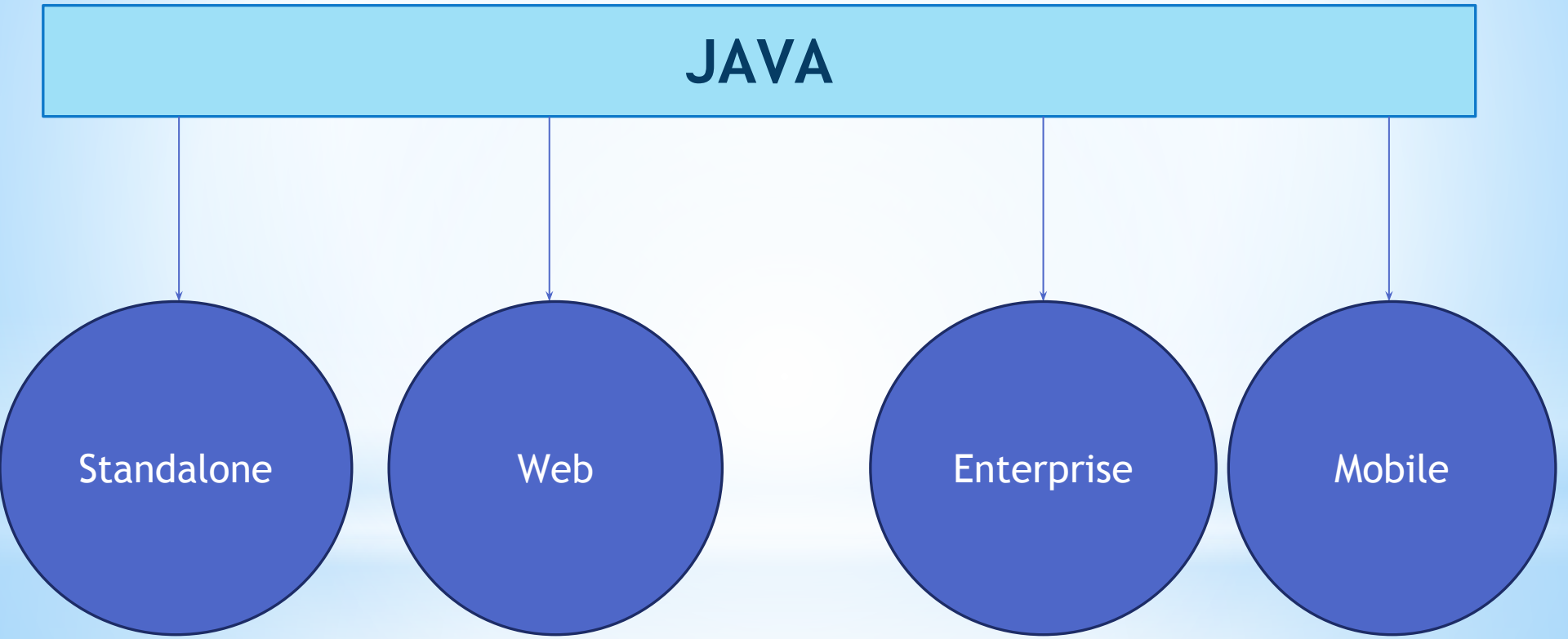
- Java is a **programming language** and a **platform.**
- It is a **high-level** language.
- **Platform:** Any hardware or software environment in which program runs is known as platform
- Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.
- Java is a simple and yet powerful object oriented programming language
- Java code that runs on one platform does not need to be recompiled to run on another platform, it’s called write once, run anywhere(WORA)

Application

1. **Desktop** (Media player, antivirus)
2. **Web** (irctc.co.in, javatpoint.com)
3. **Enterprise** (banking)
4. **Mobile** (word press)
5. **Embedded** system(toaster)
6. **Smart card**(bit coin wallet)
7. **Games** (2048)



Types of java applications



Types of java applications

1. Standalone : known as desktop/window based applications.

We need to install on every machine.

E.g. : media player, antivirus etc.

Swing and AWT are used for creating standalone applications.

2. Web : that runs on server side and creates a dynamic page is called web applications

Spring, struts, hibernate, jsf technologies used for creating web applications.

Types of java applications

3. Enterprise : that is distributed in nature, such as banking applications, etc. is called enterprise application.

It has advantages of the high-level security, load balancing, and clustering.

E.g. : media player, antivirus etc.

EJB is used for creating enterprise applications..

4. Mobile : which is created for mobile devices is called a mobile application.

Currently, Android and Java ME are used for creating mobile applications.

Java platforms/editions

There are 4 platforms or editions of Java:

1. **Java SE (Java Standard Edition):** It is a Java programming platform.
   - It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math.
   - It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.
2. **Java EE (Java Enterprise Edition):**
   - It is an enterprise platform which is mainly used to develop web and enterprise applications.
   - It is built on the top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.

Java platforms/editions

3. **Java ME (Java Micro Edition) :**
   - It is a micro platform which is mainly used to develop mobile applications.
4. **JavaFX :**
   - It is used to develop rich internet applications.
   - It uses a light-weight user interface API.

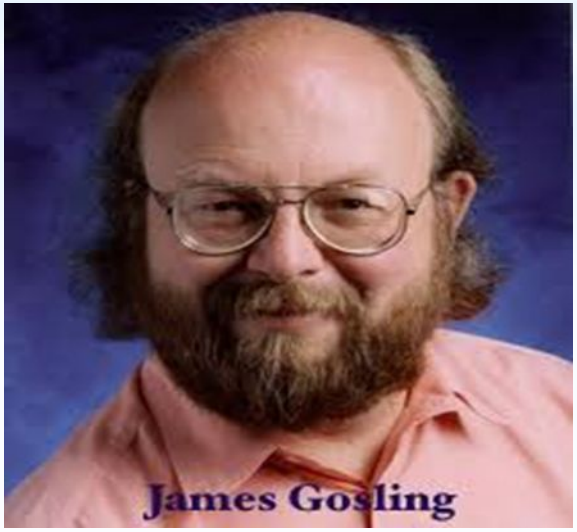
History of java

- Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time.
- The history of java starts with Green Team. Java team members (also known as **Green Team**), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc. However, it was suited for internet programming. Later, Java technology was incorporated by Netscape.
- The principles for creating Java programming were "**Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted and Dynamic**".

History of java

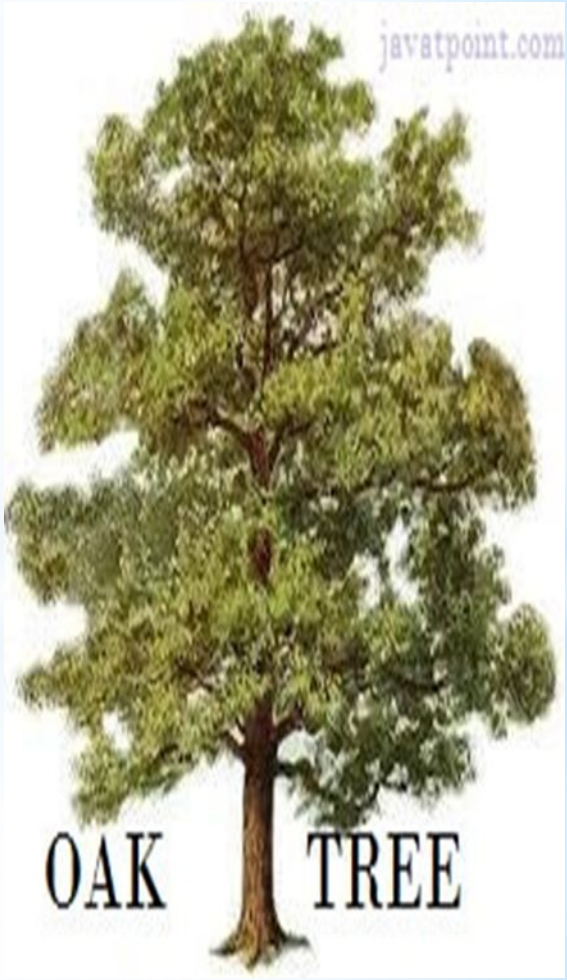
- Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc.
- There are given the significant points that describe the history of Java.

1. **James Gosling**, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.



History of java

2. Originally designed for small, embedded systems in electronic appliances like set-top boxes.
3. Firstly, it was called "**Greentalk**" by James Gosling, and file extension was .gt.
4. After that, it was called **Oak** and was developed as a part of the **Green project**.



History of java

Why Java named "Oak"?

5. **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like U.S.A., France, Germany, Romania, etc
6. In 1995, Oak was renamed as "**Java**" because it was already a trademark by Oak Technologies.

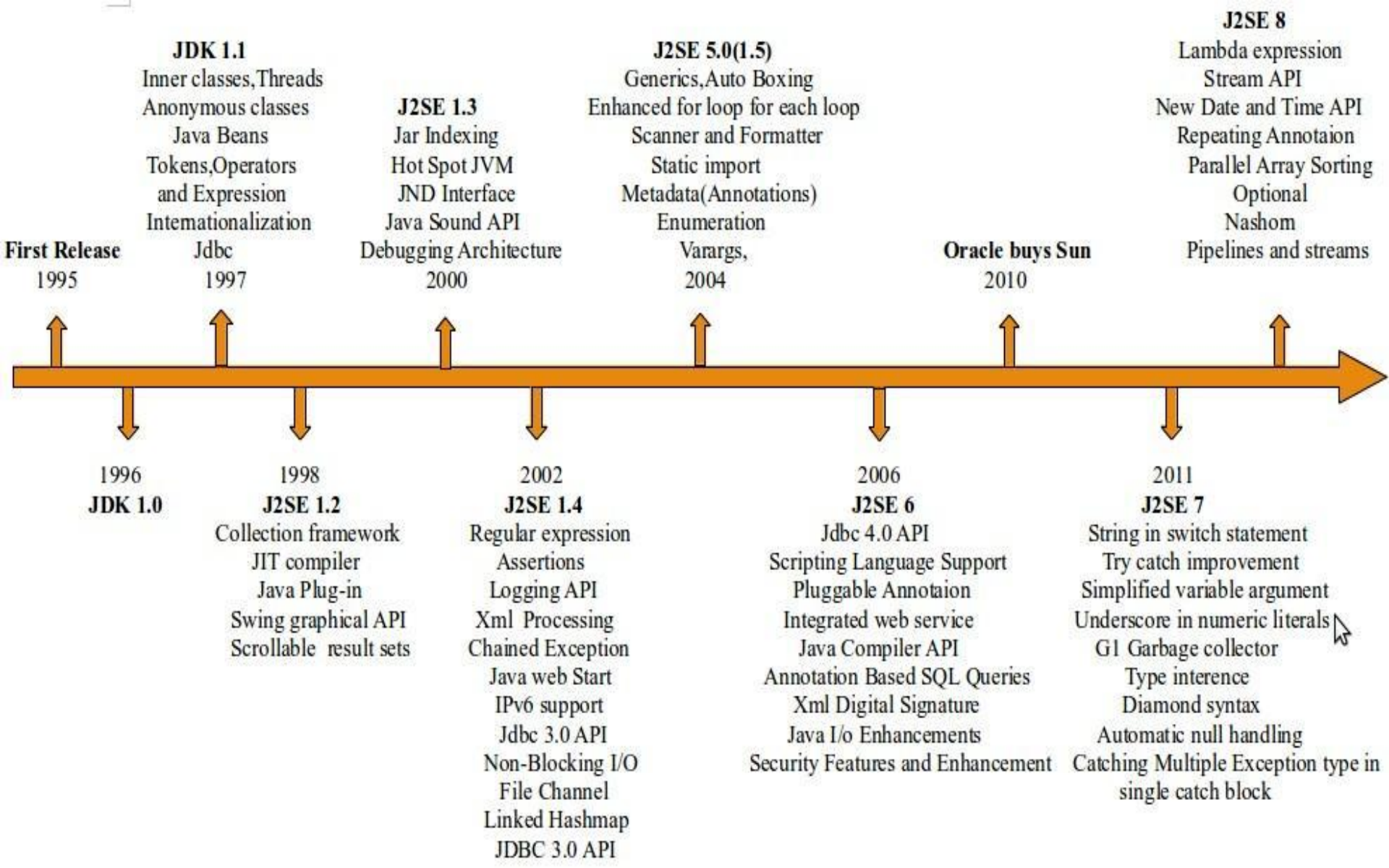
Why programming language named java?

- The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA", etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say.
- According to James Gosling, "Java was one of the top choices along with **Silk**". Since Java was so unique, most of the team members preferred Java than other names.
- Java is an island of Indonesia where first coffee was produced (called java coffee).

Java version history

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan 1996)
3. JDK 1.1 (19th Feb 1997)
4. J2SE 1.2 (8th Dec 1998)
5. J2SE 1.3 (8th May 2000)
6. J2SE 1.4 (6th Feb 2002)
7. J2SE 5.0 (30th Sep 2004)
8. Java SE 6 (11th Dec 2006)
9. Java SE 7 (28th July 2011)
10. Java SE 8 (18th March 2014)
11. Java SE 9 (21st Sep 2017)
12. Java SE 10 (20th March 2018)

Java version history



JDK Editions

- Java Standard Edition (J2SE):

  J2SE can be used to develop client-side standalone applications or applets.

- Java Enterprise Edition (J2EE):

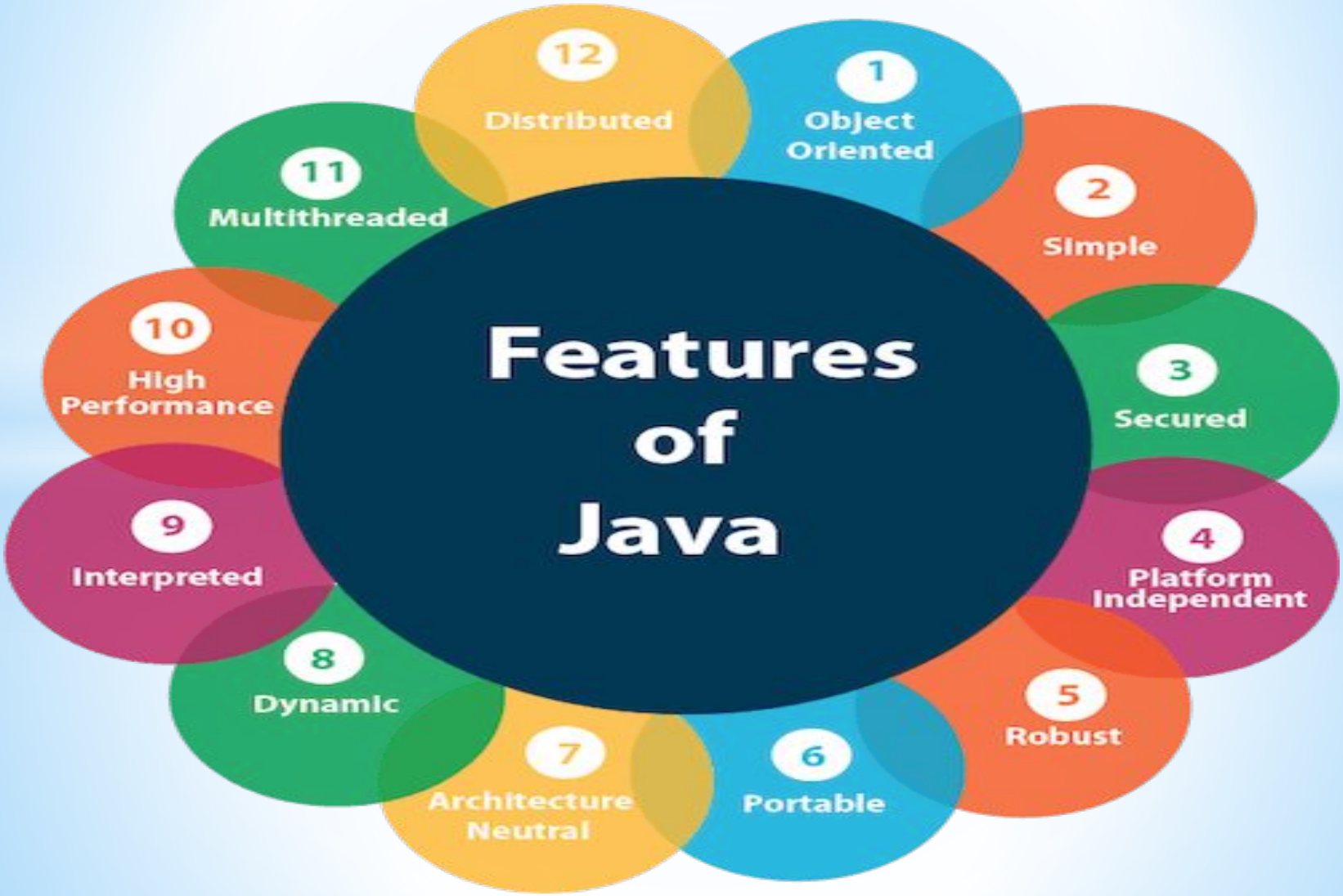
  J2EE can be used to develop server-side applications such as Java servlets and Java ServerPages.

- Java Micro Edition (J2ME):

  J2ME can be used to develop applications for mobile devices such as cell phones.

- This course we uses J2SE to introduce Java programming.

Features of java



Features of java

1. **Simple:**

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun, Java language is a simple programming language because:

- ✔ Java syntax is based on C++ (so easier for programmers to learn it after C++).
- ✔ Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- ✔ There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

Features of java

2. Object-oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.

Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

Basic concepts of OOPs are:

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

Features of java

3. Platform Independent

Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

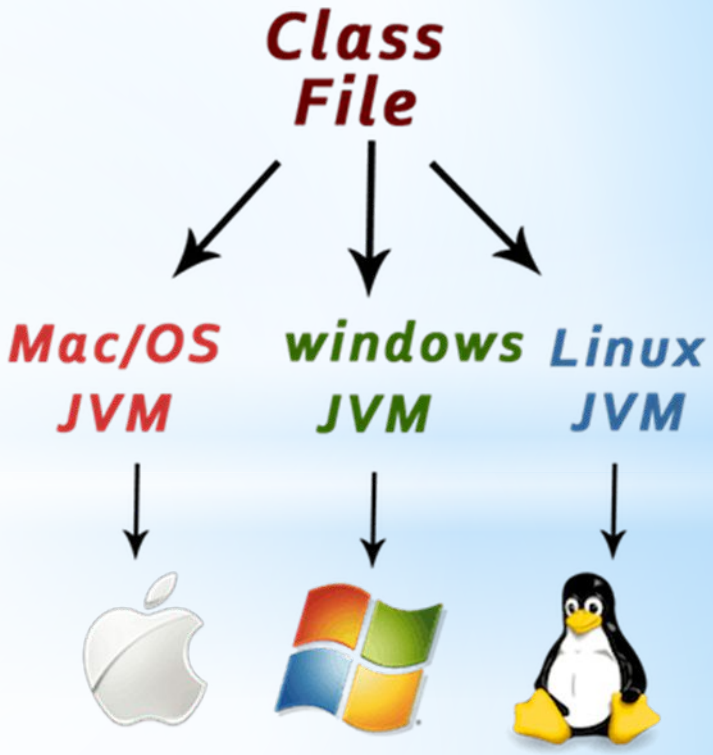
There are two types of platforms software-based and hardware-based. Java provides a software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

- ✔ Runtime Environment
- ✔ API(Application Programming Interface)

Features of java

- Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into byte code. This byte code is a platform-independent code because it can be run on multiple platforms, i.e., **Write Once and Run Anywhere(WORA).**



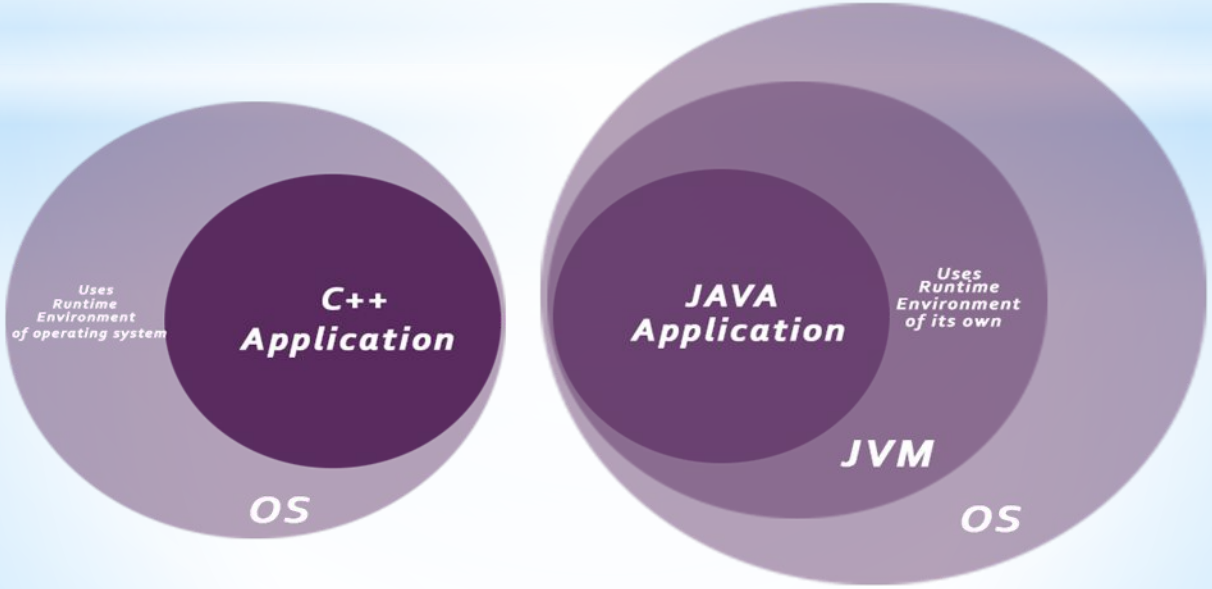
Features of java

4. **Secured :**

✔ Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

a) **No explicit pointer**

b) **Java Programs run inside a virtual machine sandbox**



Features of java

- **Classloader:** Classloader in Java is a part of the Java Runtime Environment(JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.
- **Bytecode Verifier:** It checks the code fragments for illegal code that can violate access right to objects.
- **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.

Features of java

5. **Robust :** Robust simply means strong.

Java is robust because:

1. It uses strong memory management.
2. There is a lack of pointers that avoids security problems.
3. There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
4. There are exception handling and the type checking mechanism in Java. All these points make Java robust.

Features of java

6. **Architecture-neutral**

- ✔ Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.
- ✔ In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture.
- ✔ However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

Features of java

7. **Portable**

- ✔ Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

8. **High-performance**

- ✔ Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.
- ✔ It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

Features of java

9. **Distributed**

- ✔ Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications.
- ✔ This feature of Java makes us able to access files by calling the methods from any machine on the internet.

10. **Multi-threaded**

- ✔ A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads.
- ✔ The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

Features of java

11. **Dynamic**

- ✔ Java is a dynamic language. It supports dynamic loading of classes.
- ✔ It means classes are loaded on demand.
- ✔ It also supports functions from its native languages, i.e., C and C++.
- ✔ Java supports dynamic compilation and automatic memory management (garbage collection).

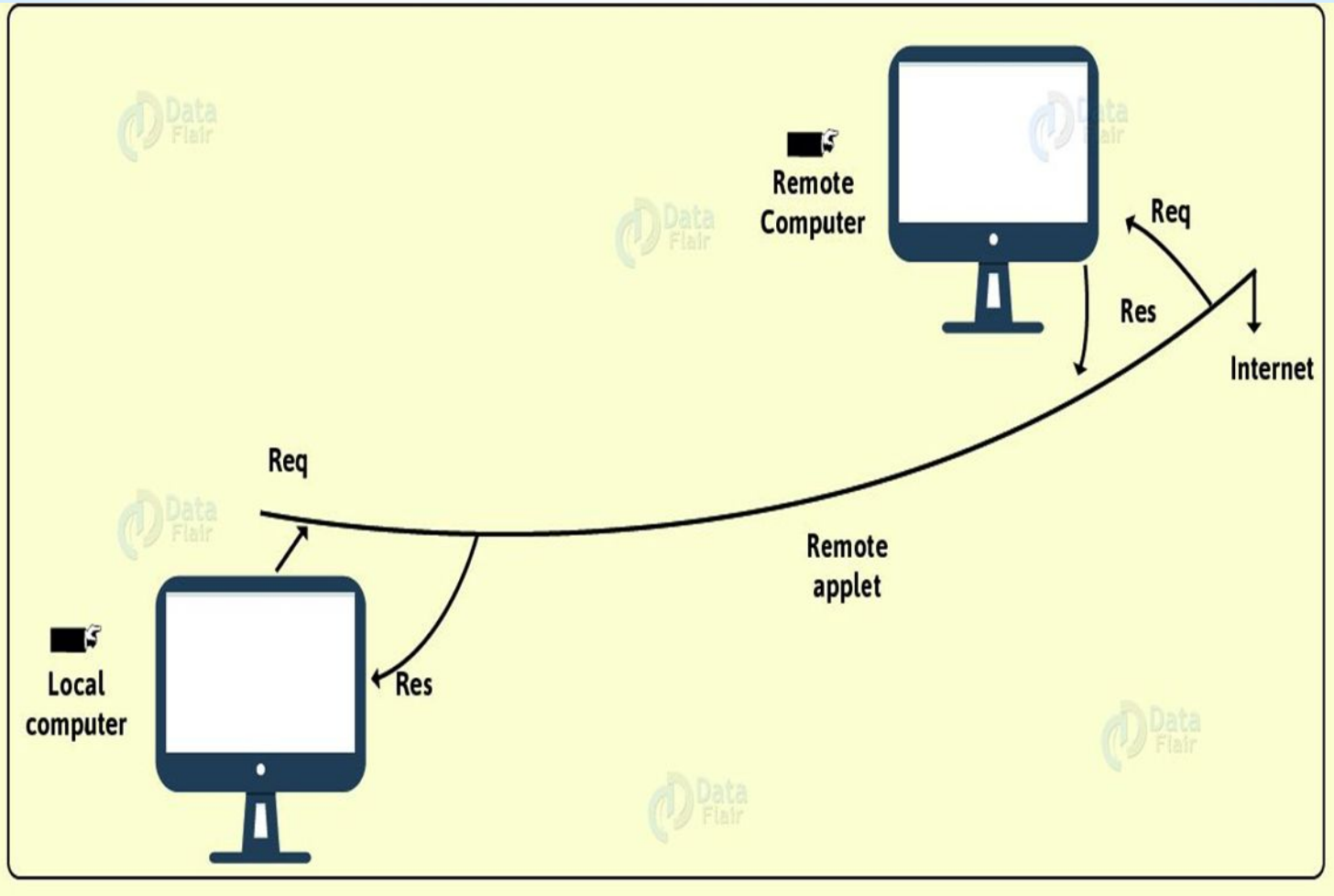
C++ vs JAVA

Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Goto	C++ supports goto statement.	Java doesn't support goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. But you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses compiler only.	Java uses compiler and interpreter both.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.

C++ vs JAVA

Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
Inheritance Tree	C++ creates a new inheritance tree always.	Java uses single inheritance tree always because all classes are the child of Object class in java. Object class is the root of inheritance tree in java.

Java and internet



Java and internet

- The Java Programming is often called **Internet language** because the first application program written in Java was HotJava, a Web browser to run applets on the Internet.
- Internet users can use Java to create applets and run them locally using HotJava.
- A Java-enabled browser to download an applet located anywhere on the Internet can also use.
- Java applets have made the Internet a true extension of the storage system on local computers.
- Internet users can also set up their websites containing Java applets that could use by remote users.

How to set Path

1. There are two ways to set the path in Java:

   Temporary

   Permanent

1) How to set the Temporary Path of JDK in Windows

To set the temporary path of JDK, you need to follow the following steps:

Open the command prompt

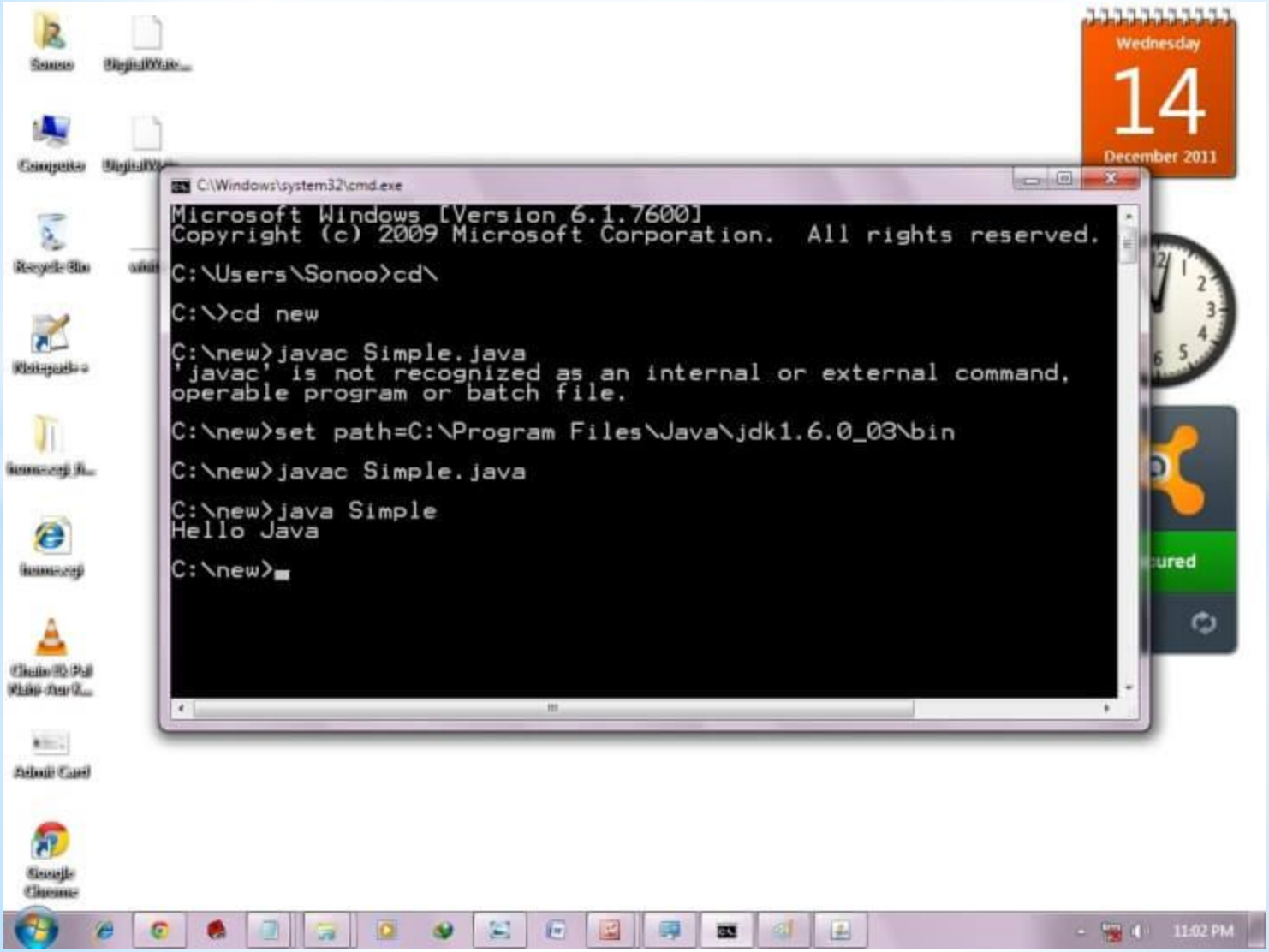
Copy the path of the JDK/bin directory

Write in command prompt: set path=copied_path

For Example:

set path=C:\Program Files\Java\jdk1.6.0_23\bin

How to set Path



How to set Path

2) How to set Permanent Path of JDK in Windows

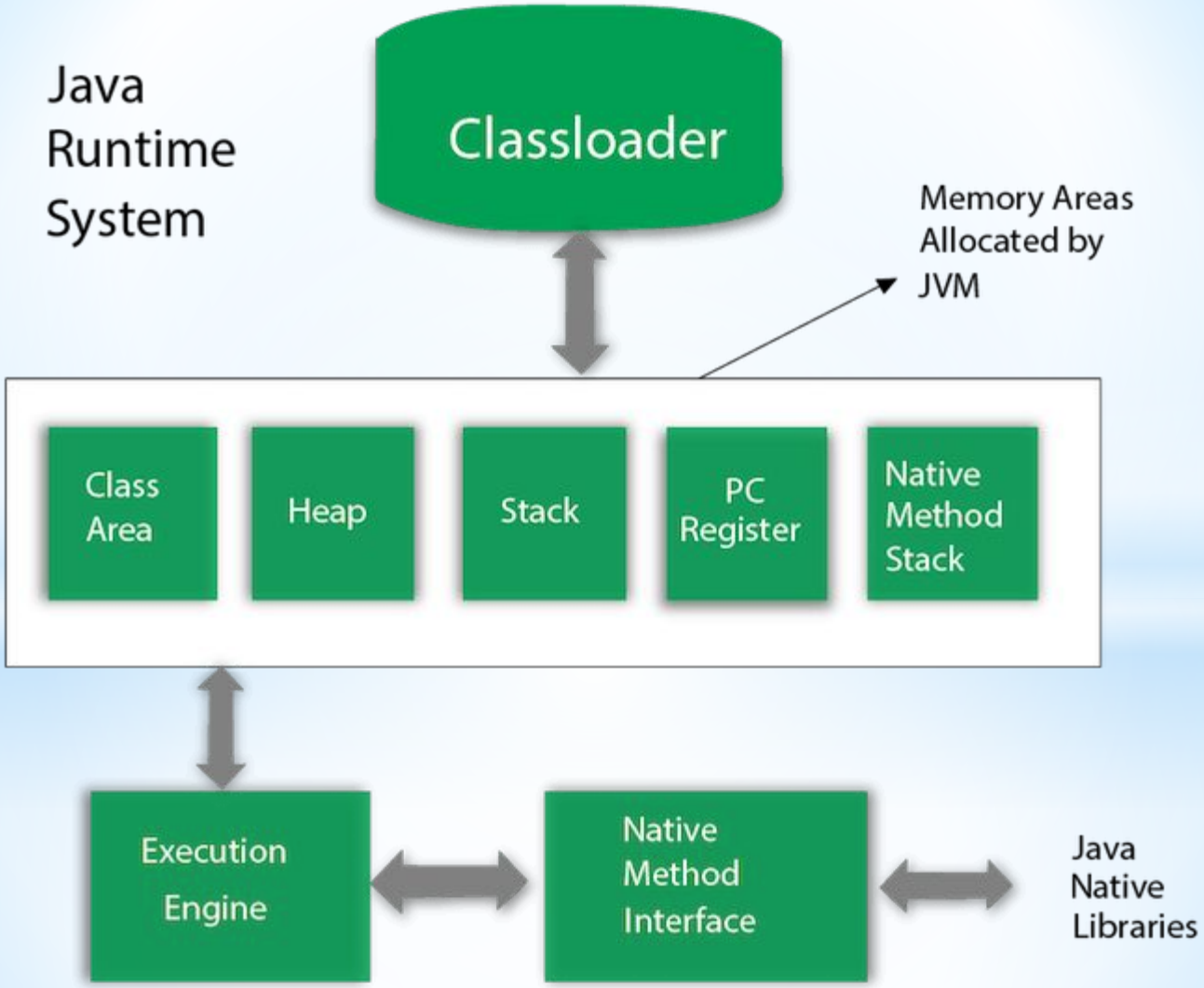
- For setting the permanent path of JDK, you need to follow these steps:

  Go to MyComputer properties -> advanced tab -> environment variables -> new tab of user variable -> write path in variable name -> write path of bin folder in variable value -> ok -> ok -> ok

3) How to set Permanent Path of JDK in Linux

- export PATH=$PATH:/home/jdk1.6.01/bin/

JVM Architecture



JVM Architecture

- JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.
- **It is:**

✔ **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.

✔ **An implementation** Its implementation is known as JRE (Java Runtime Environment).

✔ **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

What it does

The JVM performs following operation:

1. Loads code
2. Verifies code
3. Executes code
4. Provides runtime environment

JVM provides definitions for the:

5. Memory area
6. Class file format
7. Register set
8. Garbage-collected heap
9. Fatal error reporting etc.

JVM Architecture

1. **Class loader:** Class loader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the class loader. There are three built-in class loaders in Java.

   I. **Bootstrap ClassLoader**: This is the first class loader which is the super class of Extension class loader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes etc.

   II. **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loades the jar files located inside *$JAVA_HOME/jre/lib/ext* directory.

   III. **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the classfiles from classpath. By default, classpath is set to current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

JVM Architecture

2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

JVM Architecture

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

It contains:

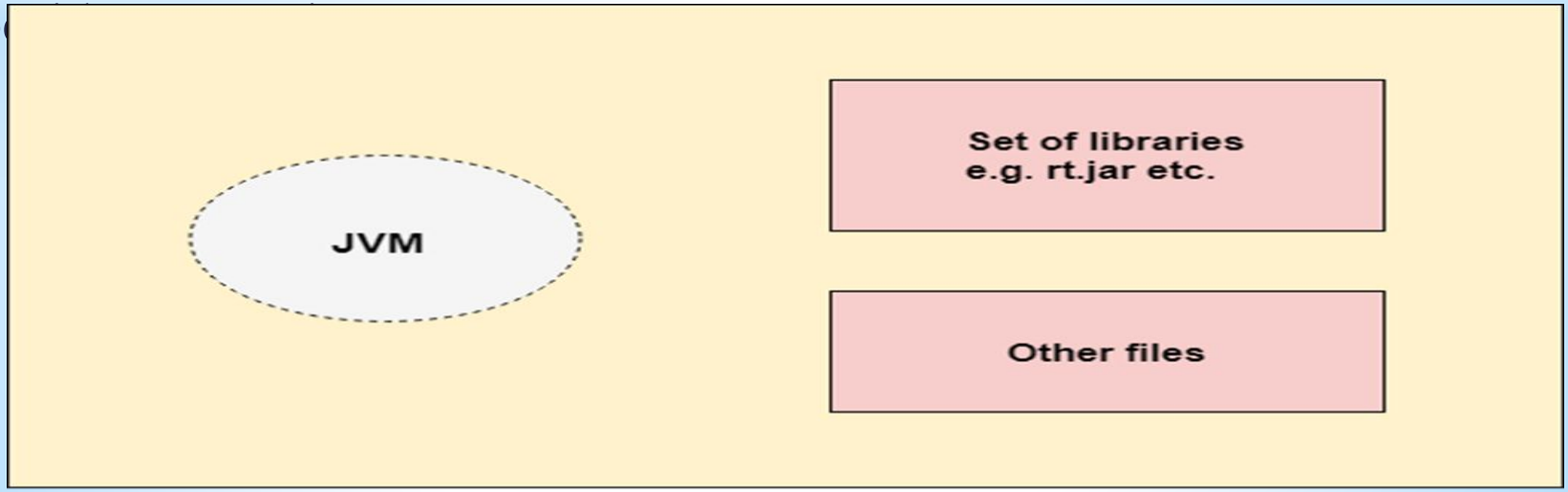
1. **A virtual processor**
2. **Interpreter:** Read bytecode stream then execute the instructions.
3. **Just-In-Time(JIT) compiler:** It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

8) Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS libraries.

JRE

- JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications.
- It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.
- The implementation of JVM is also actively released by other companies b



JRE

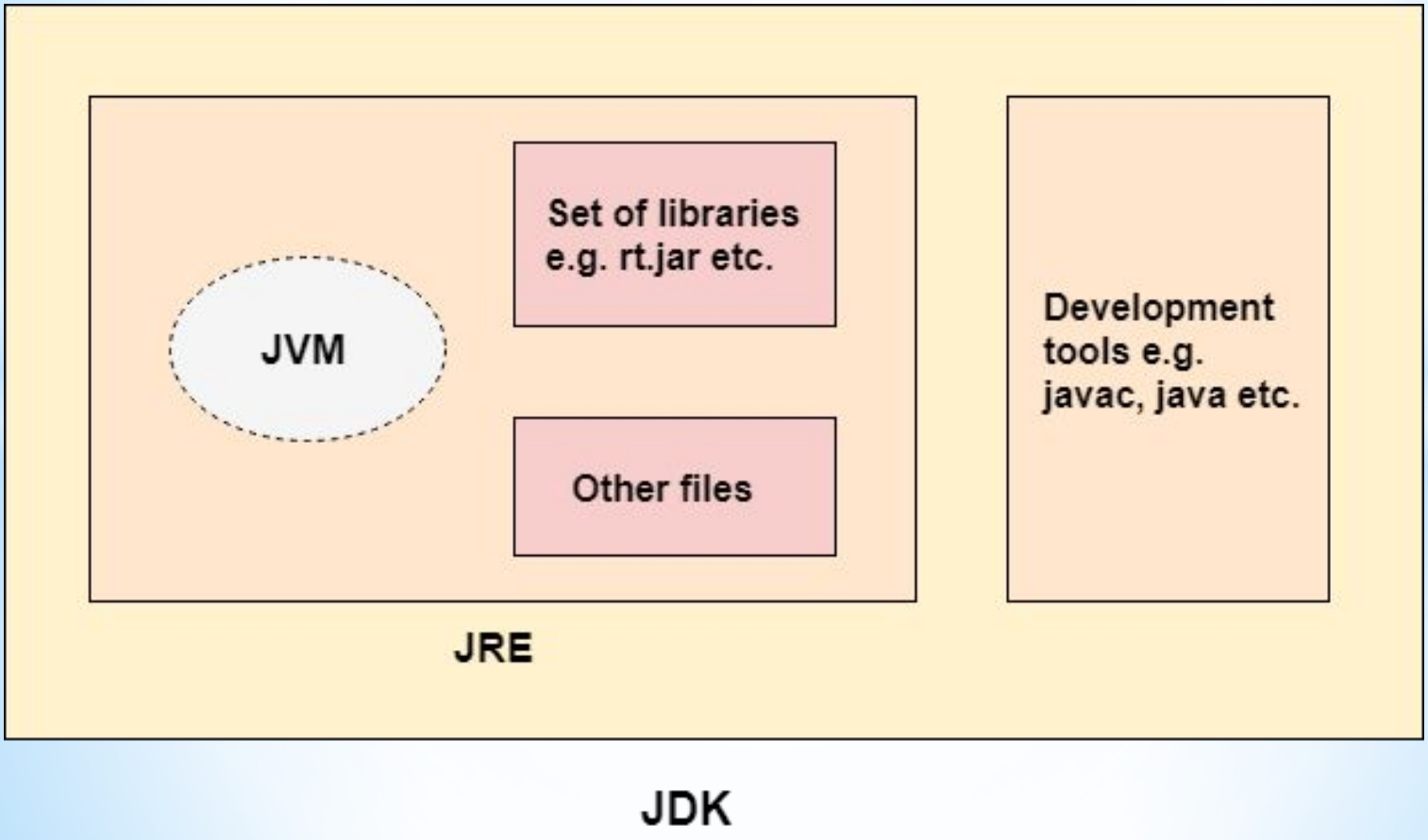
JDK

- JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.
- JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

1. Standard Edition Java Platform
2. Enterprise Edition Java Platform
3. Micro Edition Java Platform

- The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc),

JDK


Set of libraries
e.g. rt.jar etc.
JVM
Other files
Development
tools e.g.
javac, java etc.
JRE
JDK

JAVA Variable

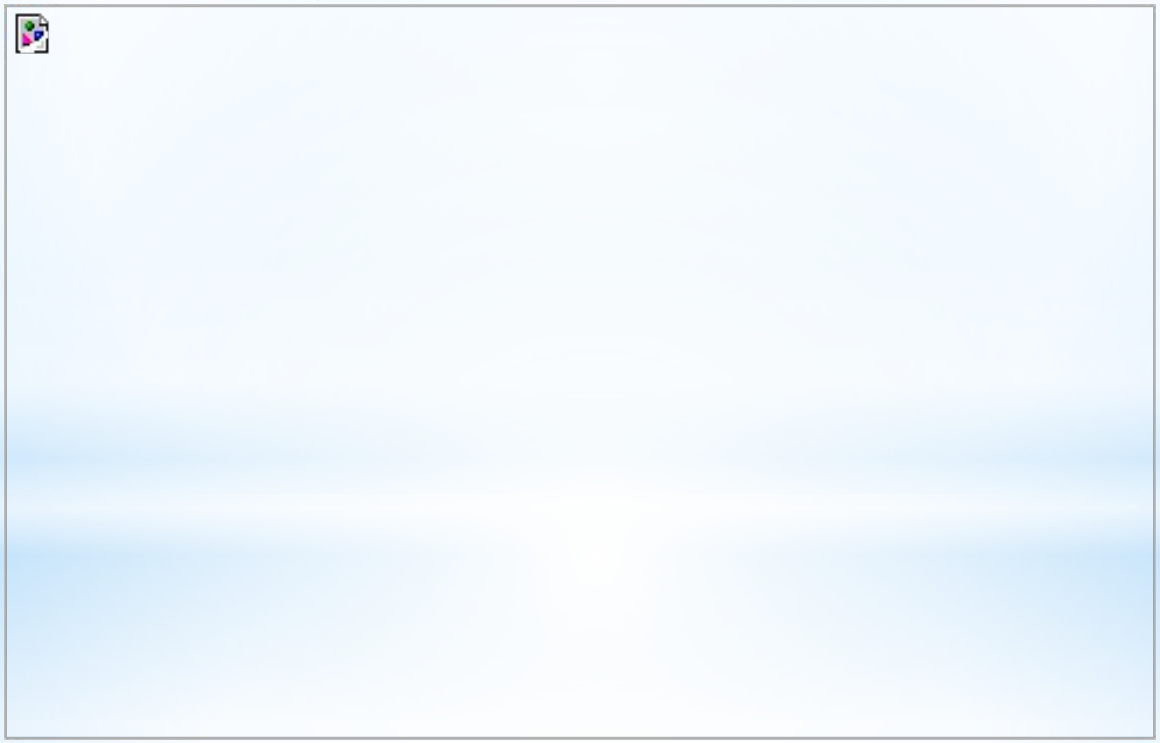
- A variable is a container which holds the value while the java program is executed. A variable is assigned with a datatype.
- Variable is a name of memory location. There are three types of variables in java: local, instance and static.
- There are two types of data types in java: primitive and non-primitive.

Variable

Variable is name of reserved area allocated in memory. In other words, it is a name of memory location. It is a combination of "vary + able" that means its value can be changed.

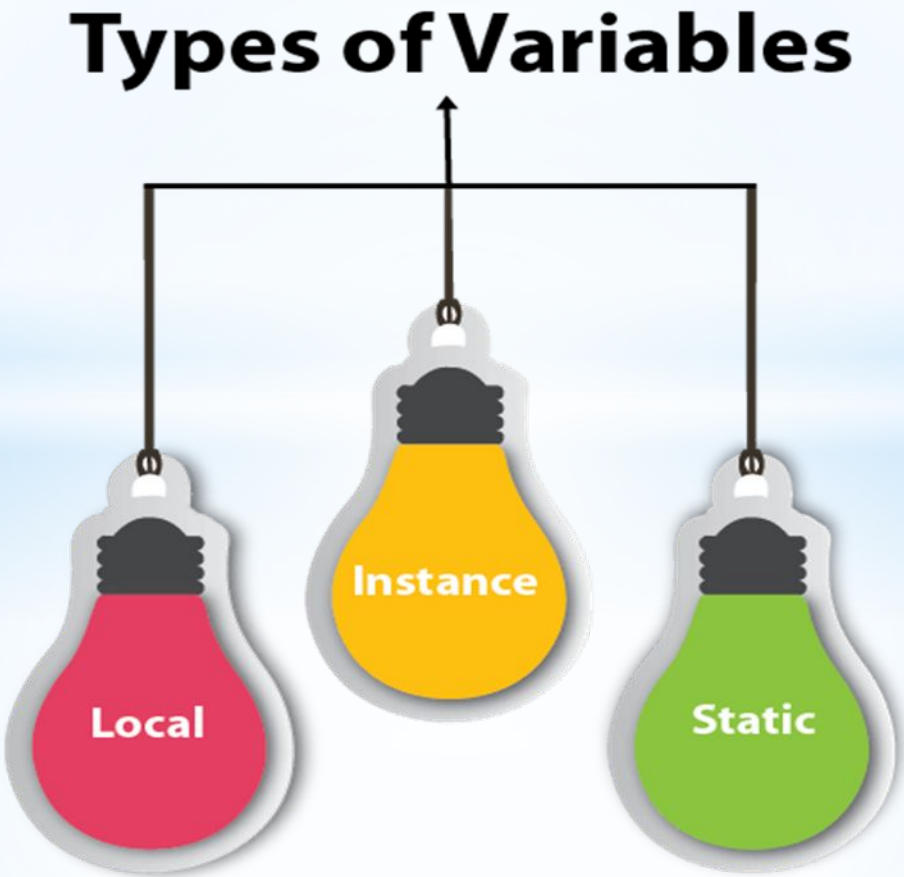
JAVA Variable

Example: int data=10;//Here data is variable



JAVA Variable

Types of Variables :



JAVA Variable

Types of Variables :

1) Local Variable

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

2) Instance Variable

A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.

It is called instance variable because its value is instance specific and is not shared among instances.

3) Static variable

A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

JAVA Variable

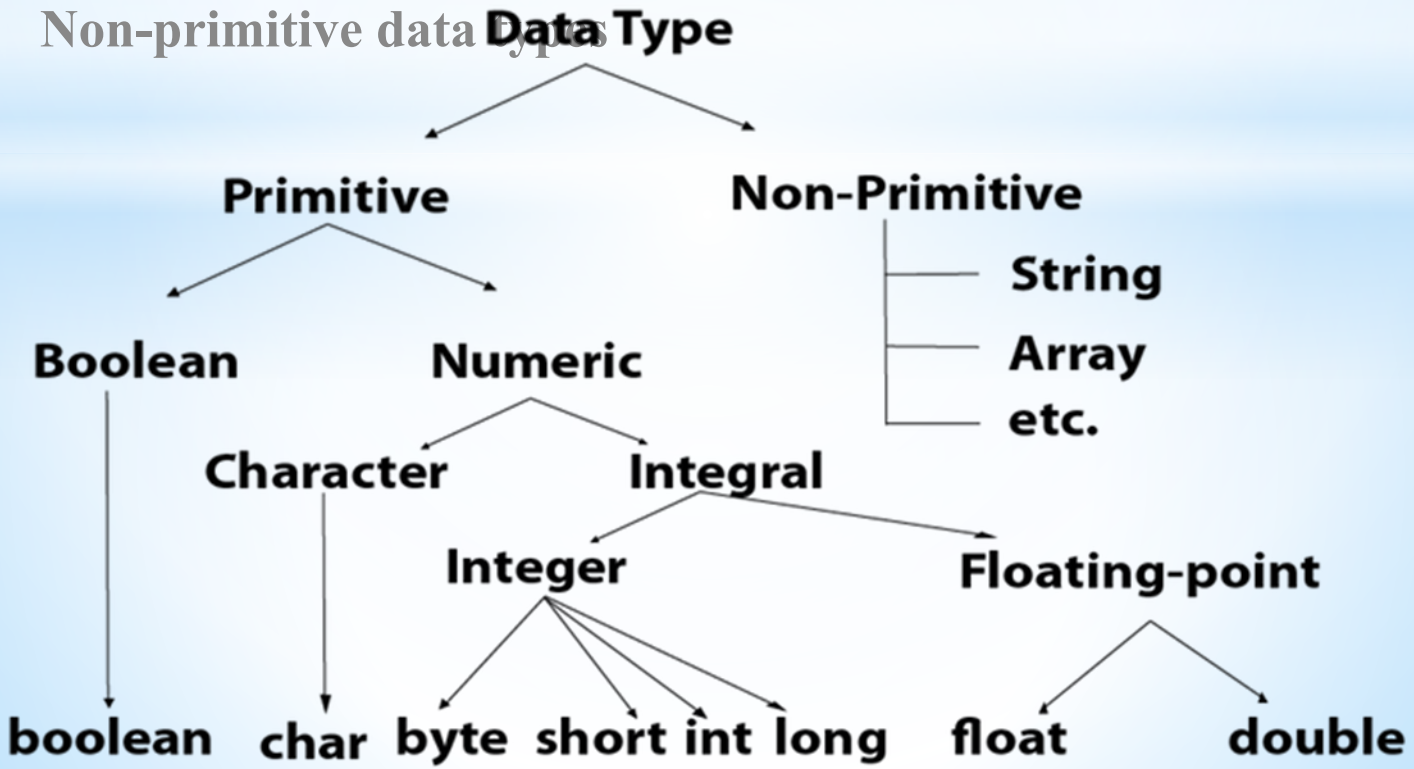
Declaration of Variables:

Syntax: data_type variable_name;

Giving Values to Variables : int a = 10;

Data type

- Data types specify the different sizes and values that can be stored in the variable.
- There are two types of data types in Java:
  1. **Primitive data types**
  2. **Non-primitive data types**



Data type

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Data type

PRIMITIVE DATA TYPE in JAVA

Type	Contains	Default	Size	Range
byte	Signed integer	0	8 bits	-128 to 127
short	Signed integer	0	16 bits	-32768 to 32767
int	Signed integer	0	32 bits	-2147483648 to 2147483647
float	IEEE 754 floating point	0.0f	32 bits	±1.4E-45 to ±3.4028235E+38
long	Signed integer	0L	64 bits	-9223372036854775808 to 9223372036854775807
double	IEEE 754 floating point	0.0d	64 bits	±4.9E-324 to ±1.7976931348623157E+308
boolean	true or false	FALSE	1 bit	NA
char	Unicode character	'\u0000'	16 bits	\u0000 to \uFFFF

Operator

operator Type	Category	Precedence
Unary	postfix	*expr++ expr--*
	prefix	*++expr --expr +expr -expr* ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Java Control Statements

- A group of statements executed in order is written

```
{ stmt1; stmt2; ...; stmtN; }
```

- The statements execute in the order 1, 2, ..., N

Control statements alter this sequential flow of execution.

Java Control Statements

Control Structure	Purpose	Syntax

`if ... else`	Used to write a decision with *conditions* that select the alternative to be executed. Executes the first (second) alternative if the *condition* is true (false).	`if (condition) {`
		`  ...`
		`} else {`
		`  ...`
		`}`

`switch`	Used to write a decision with scalar values (integers, characters) that select the alternative to be executed. Executes the *statements* following the *label* that is the *selector* value. Execution falls through to the next `case` if there is no `return` or `break`. Executes the statements following `default` if the *selector* value does not match any *label*.	`switch (selector) {`
		`  case label : statements; break;`
		`  case label : statements; break;`
		`  ...`
		`  default : statements;`
		`}`

`while`	Used to write a loop that specifies the repetition *condition* in the loop header. The *condition* is tested before each iteration of the loop and, if it is true, the loop body executes; otherwise, the loop is exited.	`while (condition) {`
		`  ...`
		`}`

`for`	Used to write a loop that specifies the *initialization*, repetition *condition*, and *update* steps in the loop header. The *initialization* statements execute before loop repetition begins; the *condition* is tested before each iteration of the loop and, if it is true, the loop body executes; otherwise, the loop is exited. The *update* statements execute after each iteration.	`for (initialization; condition; update) {`
		`  ...`
		`}`

Java Control Statements

Control Structure	Purpose	Syntax

`do ... while`	Used to write a loop that specifies the repetition *condition* after the loop body. The *condition* is tested after each iteration of the loop and, if it is true, the loop body is repeated; otherwise, the loop is exited. The loop body always executes at least one time.	`do {`
		`  ...`
		`while (`*condition*`) ;`