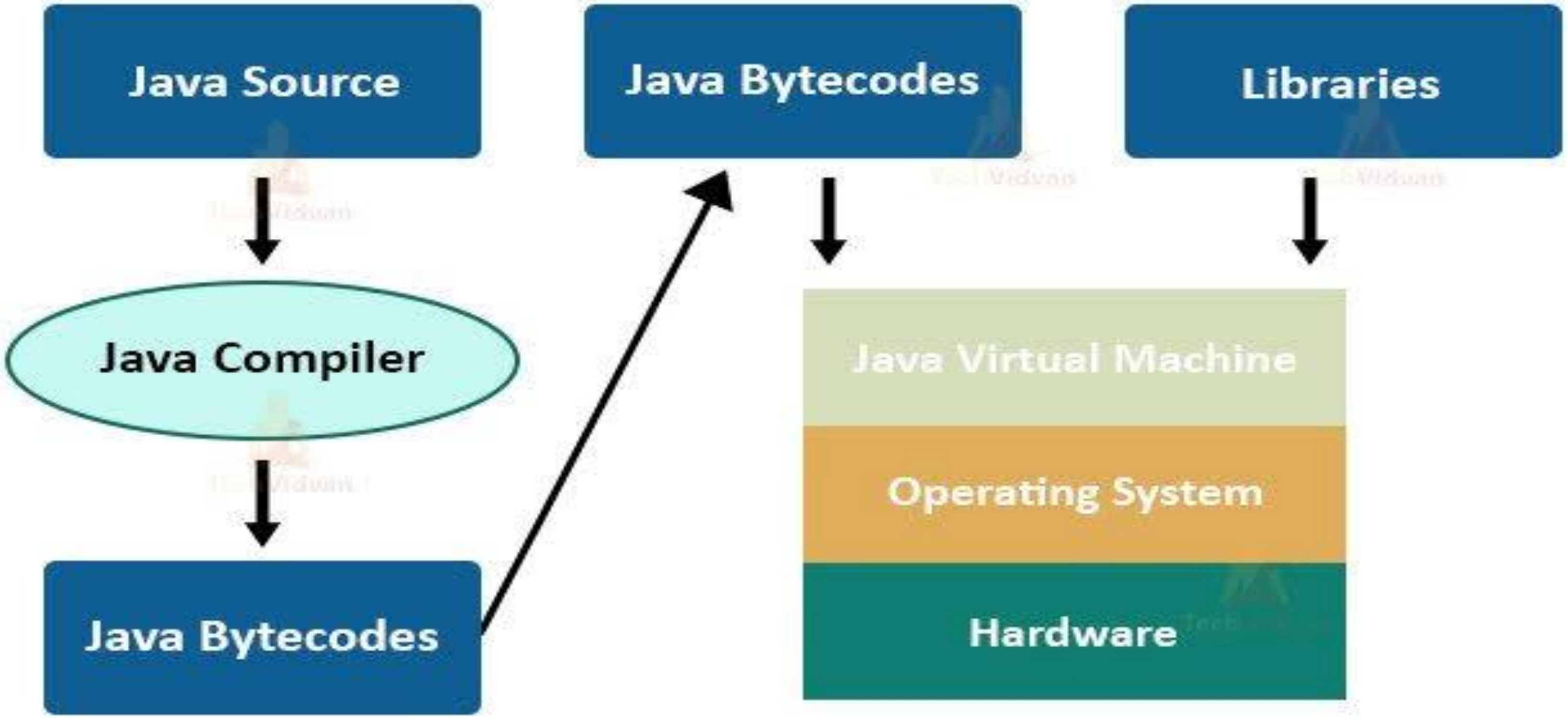
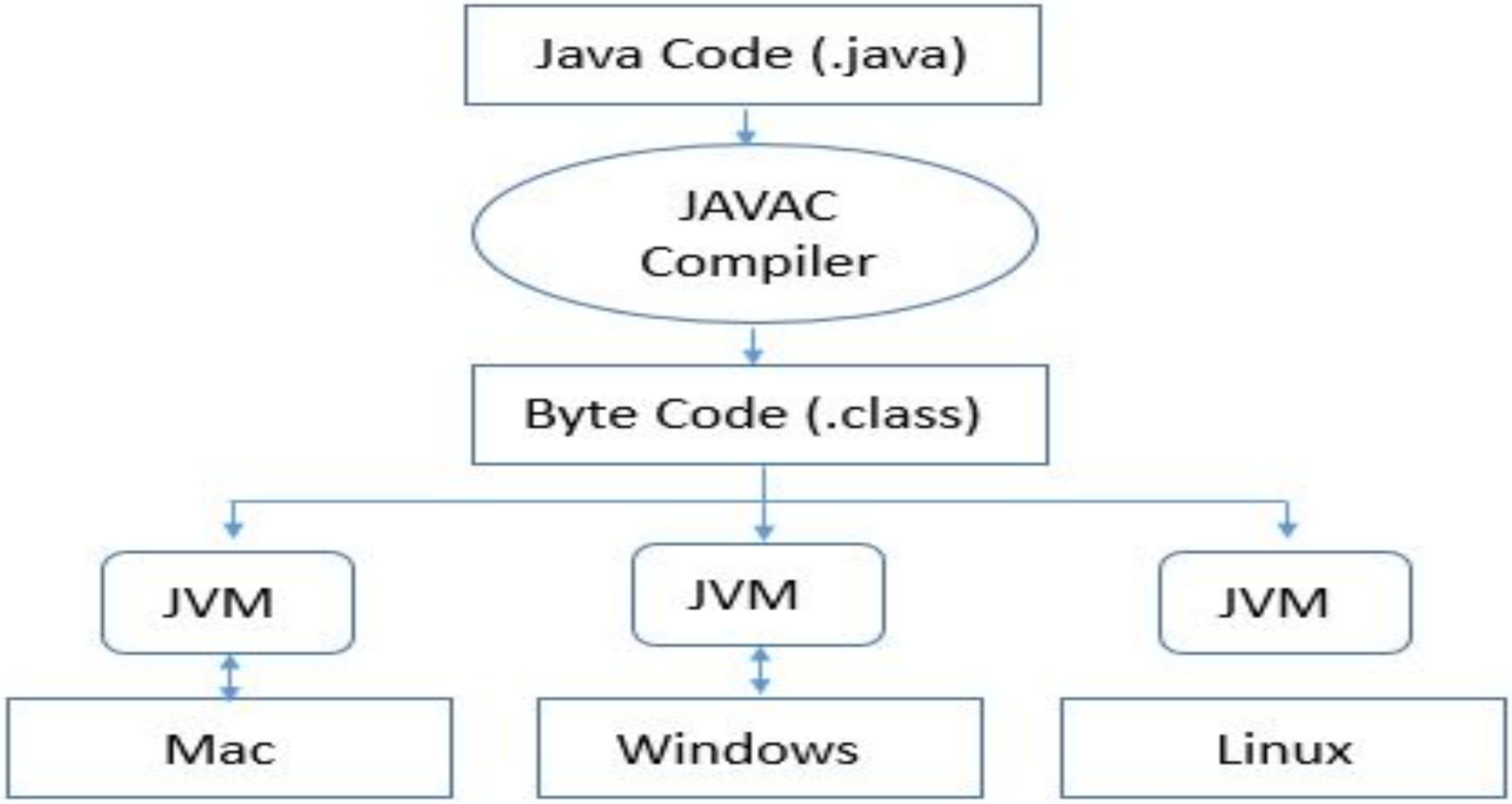
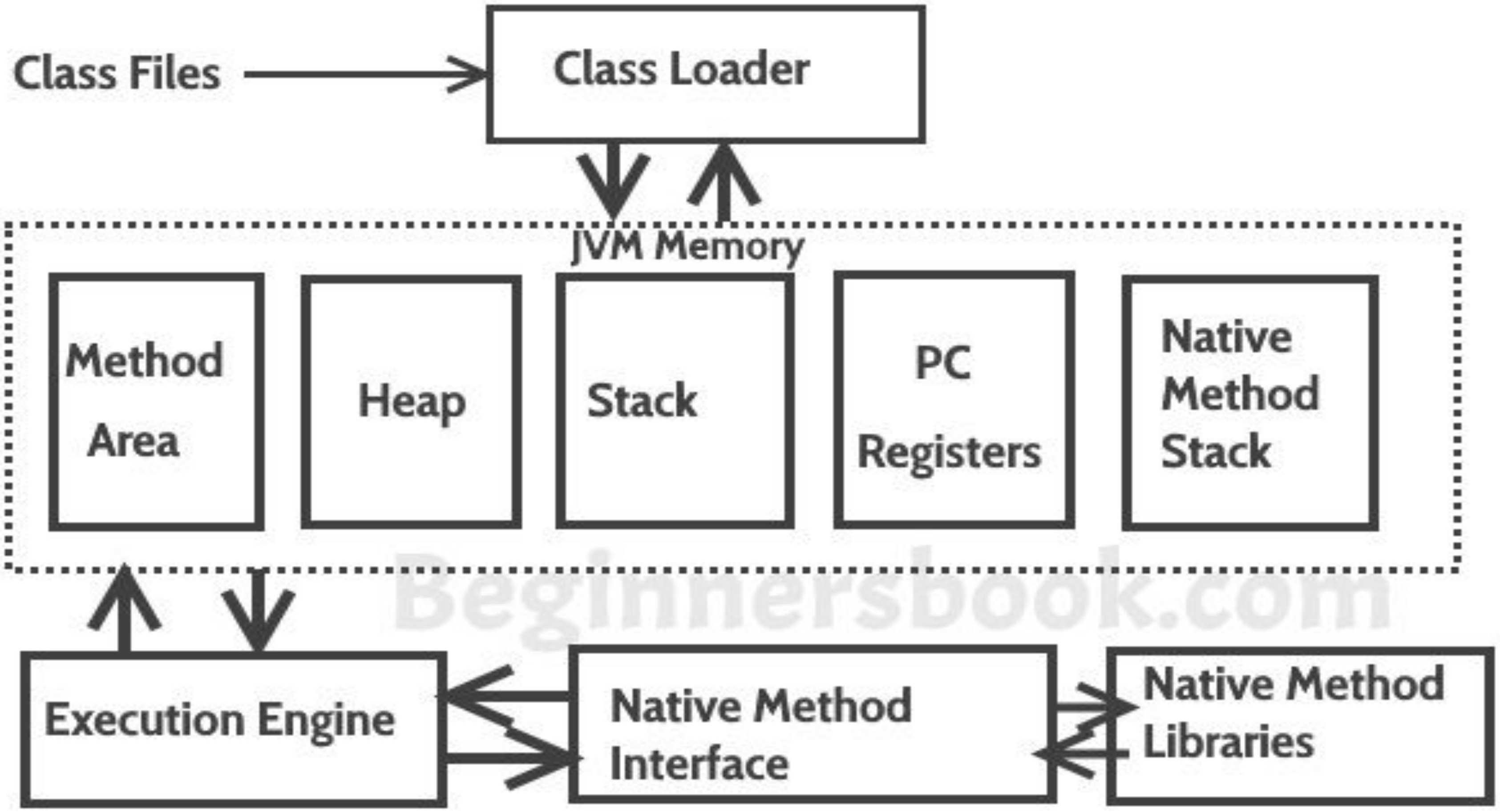
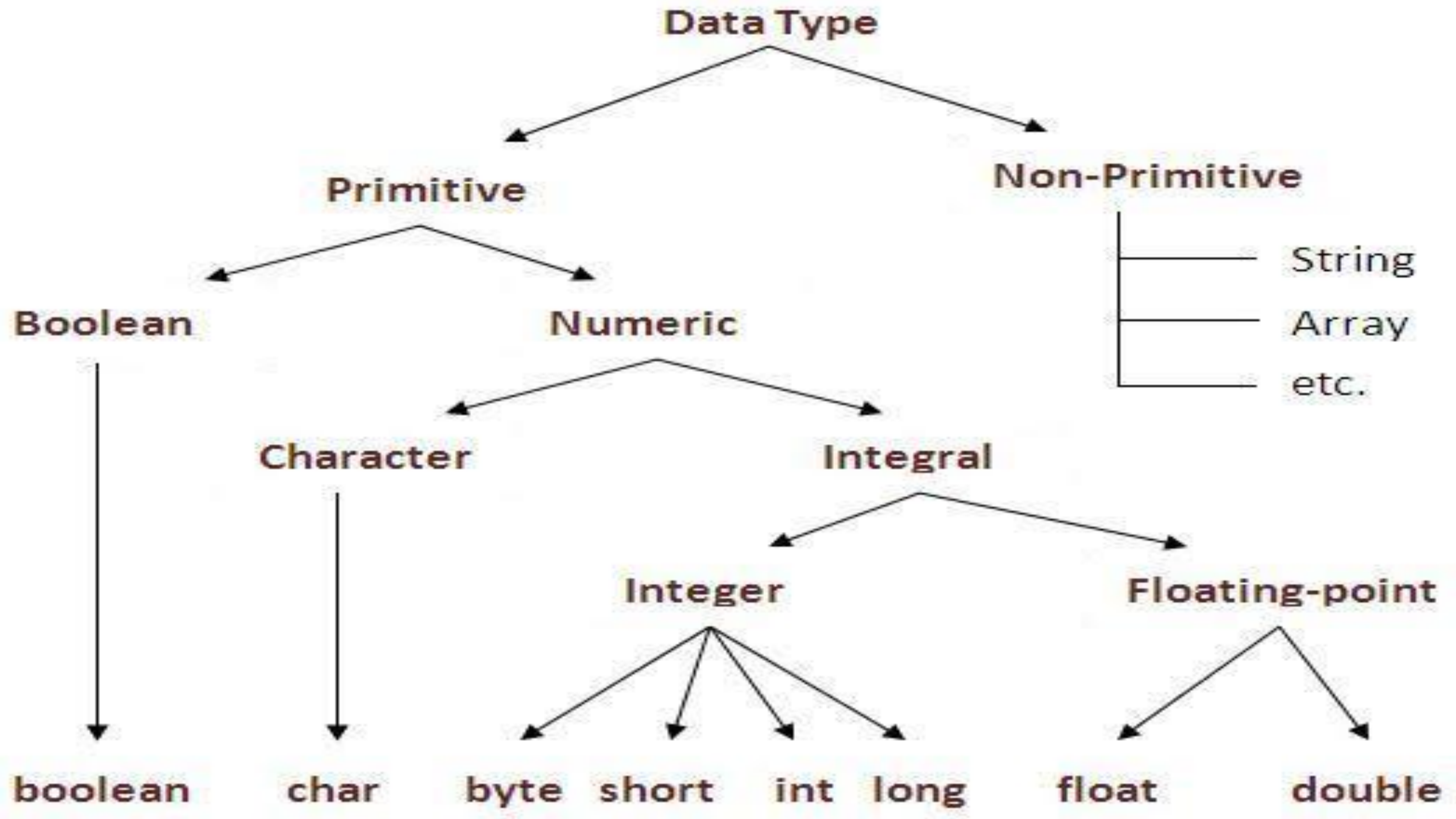
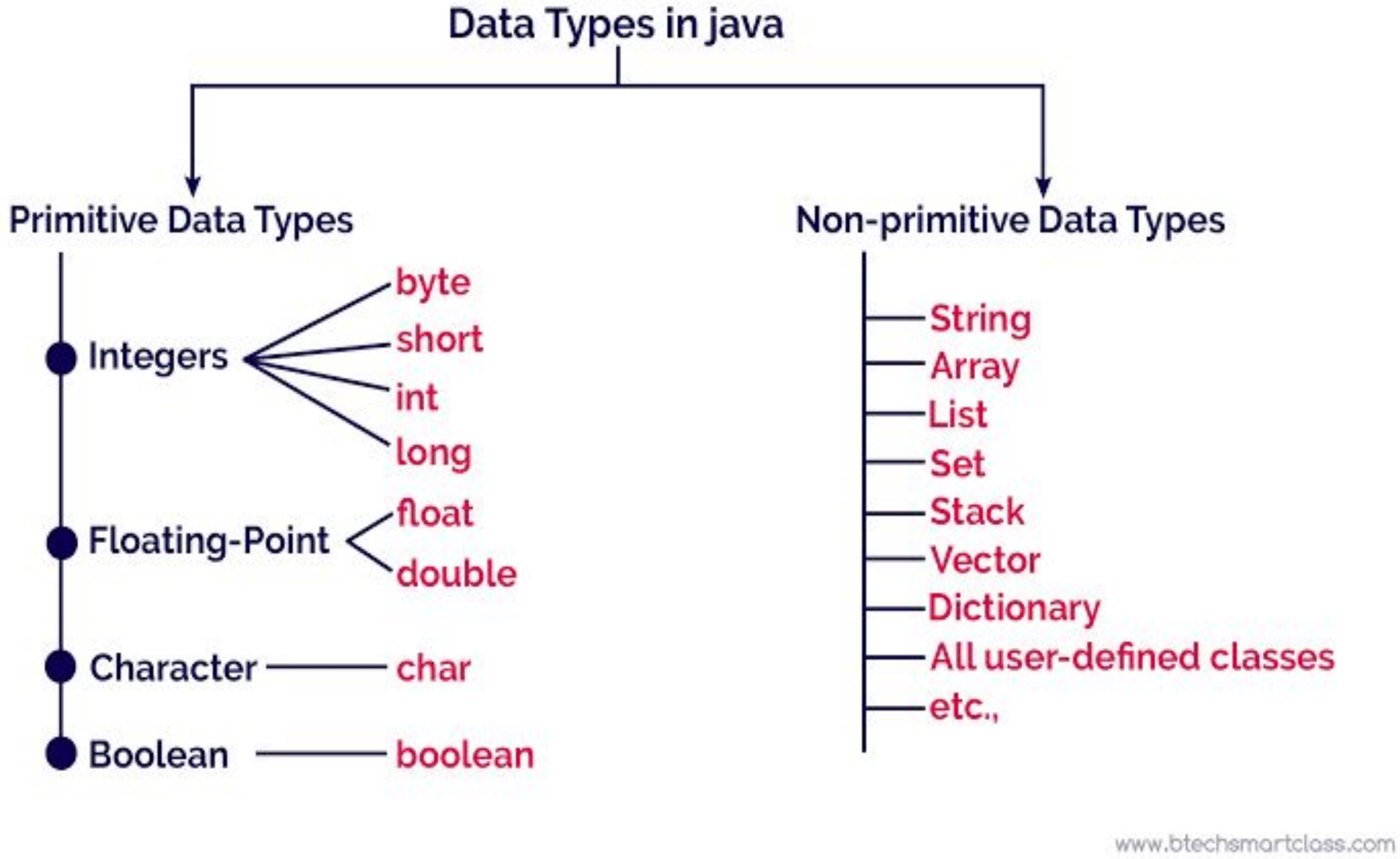
Working of JVM



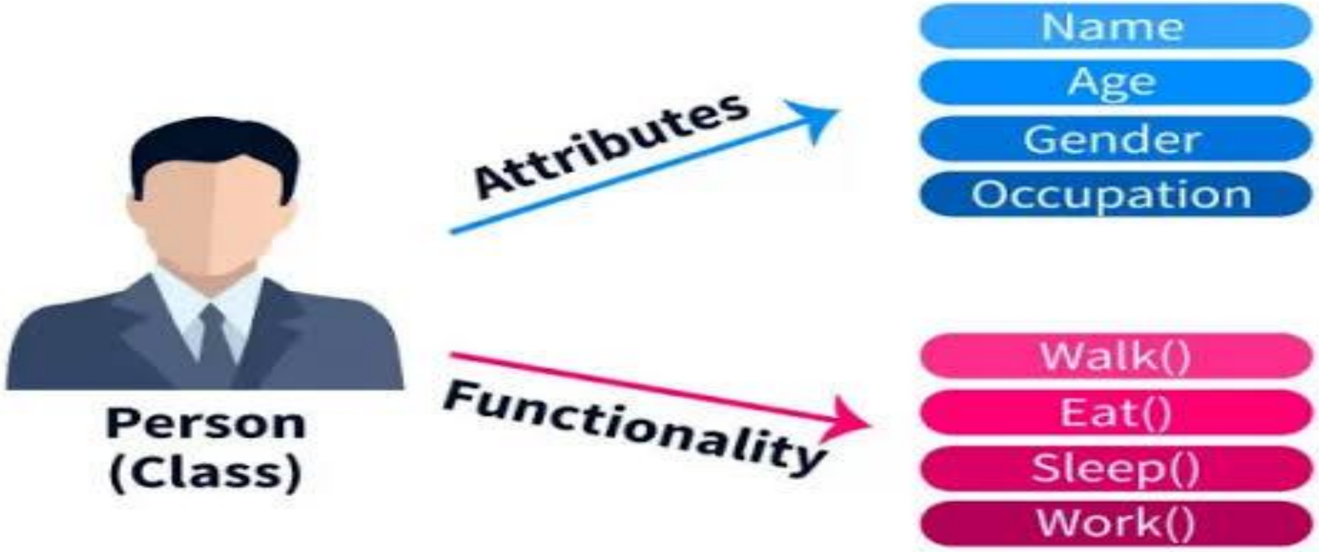

Java Code (.java)
JAVAC Compiler
Byte Code (.class)
JVM
JVM
JVM
Mac
Windows
Linux


Class Files
Class Loader
JVM Memory
Method Area
Heap
Stack
PC Registers
Native Method Stack
Execution Engine
Native Method Interface
Native Method Libraries


Data Type
Primitive
Non-Primitive
String
Array
etc.
Boolean
Numeric
Character
Integral
Integer
Floating-point
boolean
char
byte
short
int
long
float
double

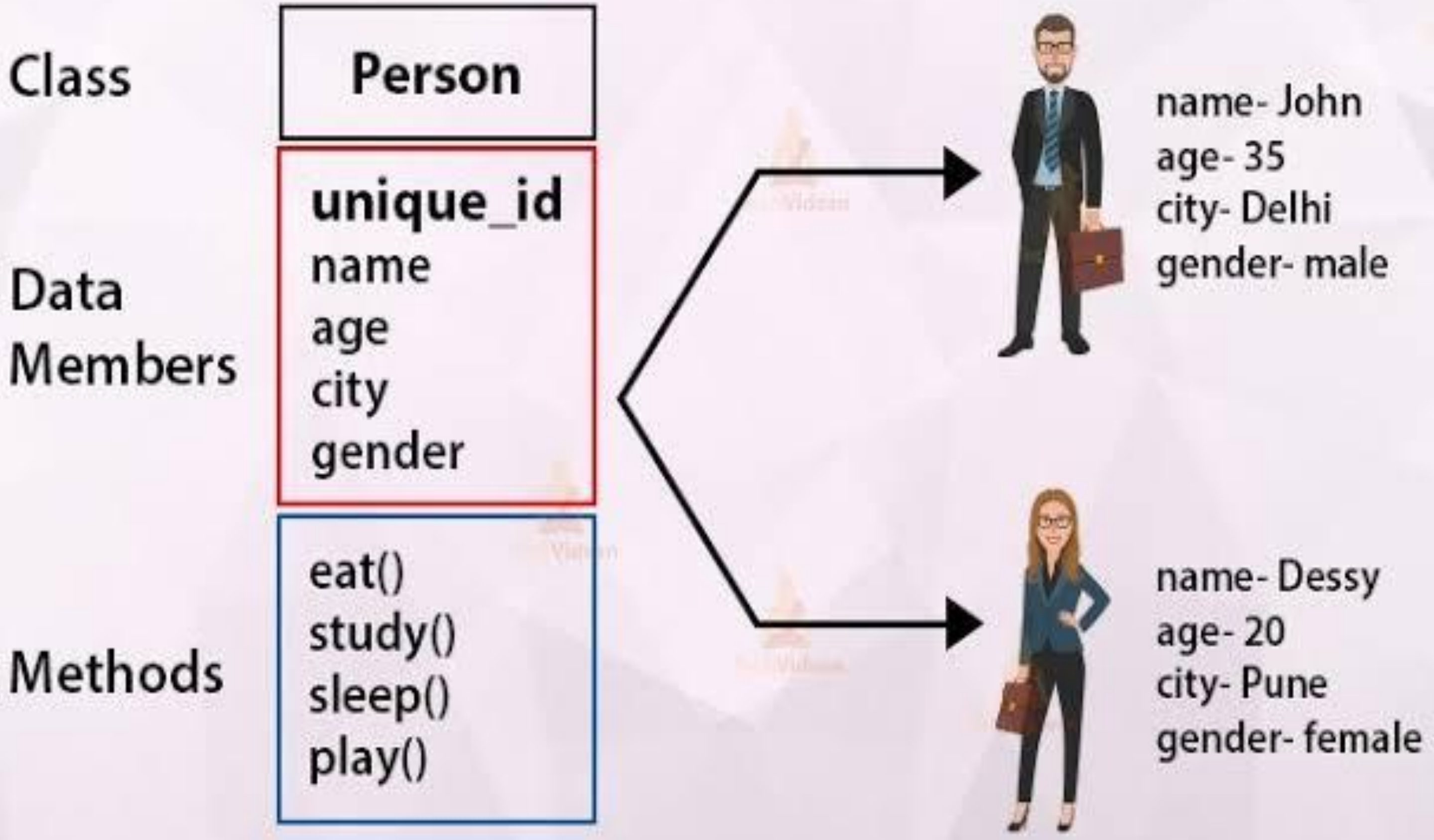

Data Types in java
Primitive Data Types
Integers
byte
short
int
long
Floating-Point
float
double
Character
char
Boolean
boolean
Non-primitive Data Types
String
Array
List
Set
Stack
Vector
Dictionary
All user-defined classes
etc.,
www.btechsmartclass.com

What is Class?

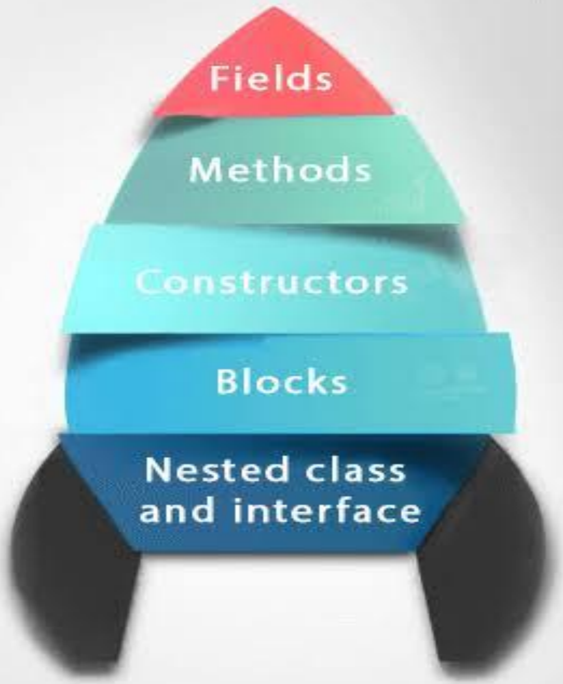


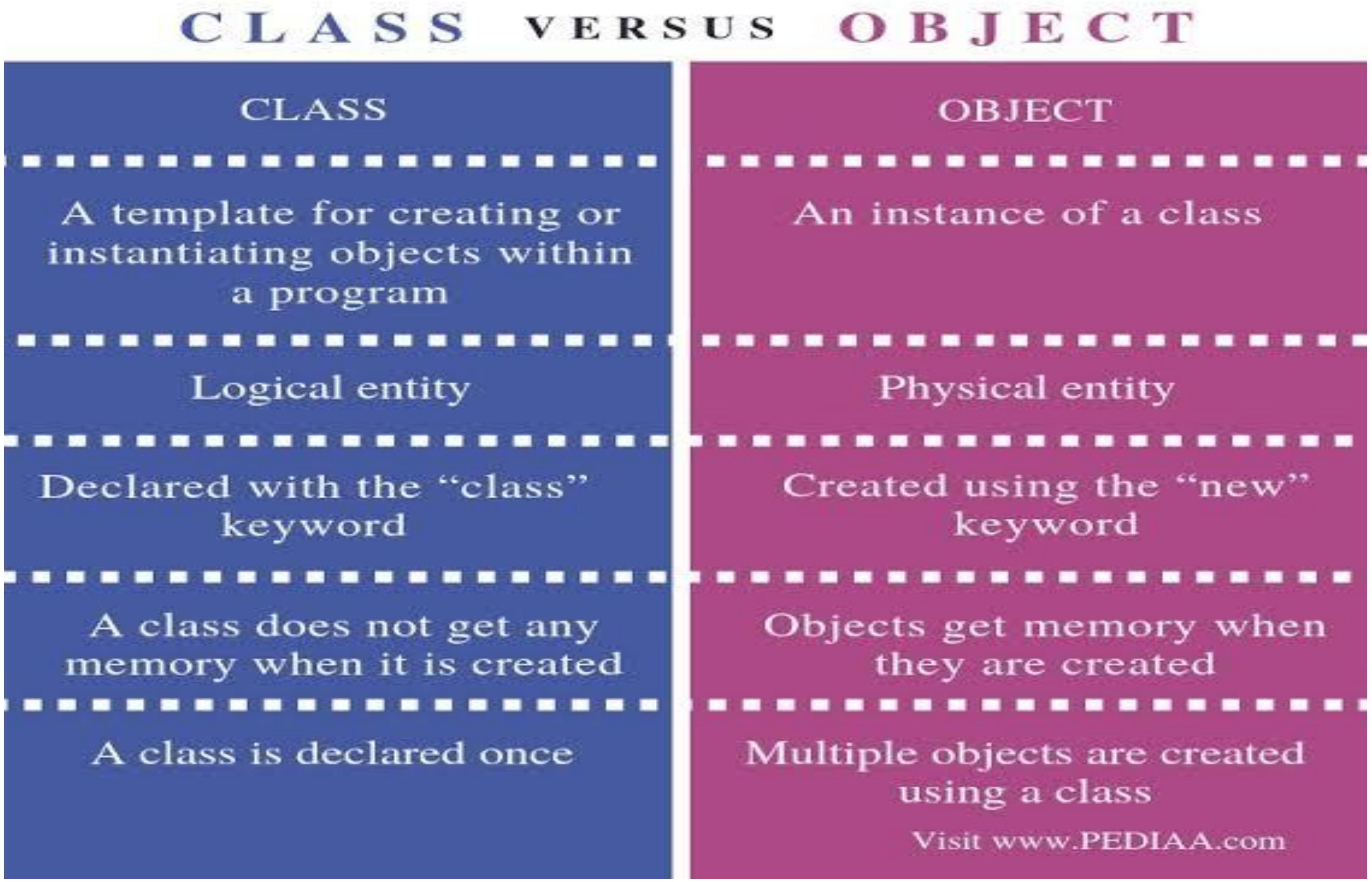
SCALER Topics

Java Class & Objects



Class in Java

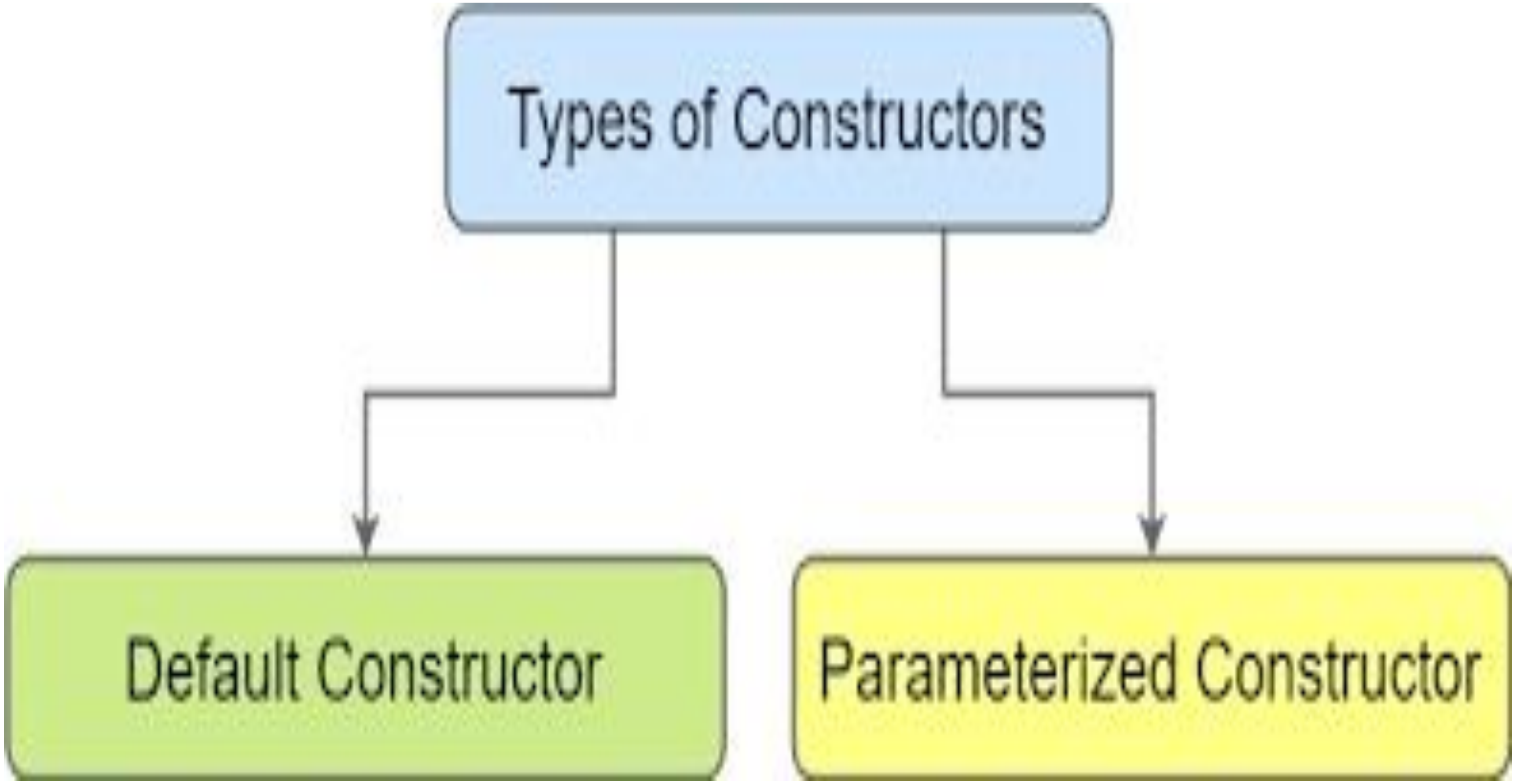


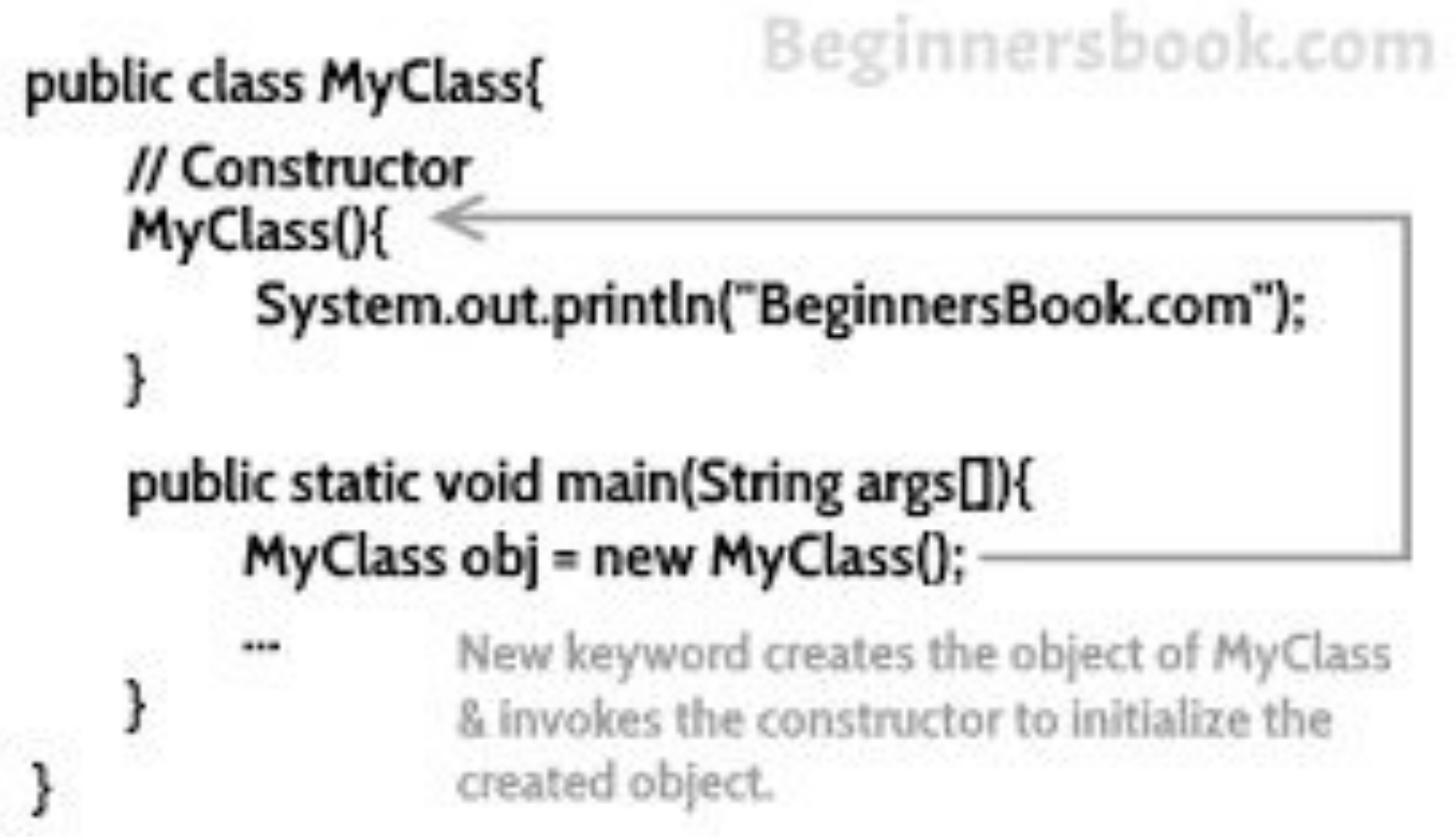


CLASS VERSUS OBJECT

CLASS	OBJECT
A template for creating or instantiating objects within a program	An instance of a class
Logical entity	Physical entity
Declared with the “class” keyword	Created using the “new” keyword
A class does not get any memory when it is created	Objects get memory when they are created
A class is declared once	Multiple objects are created using a class

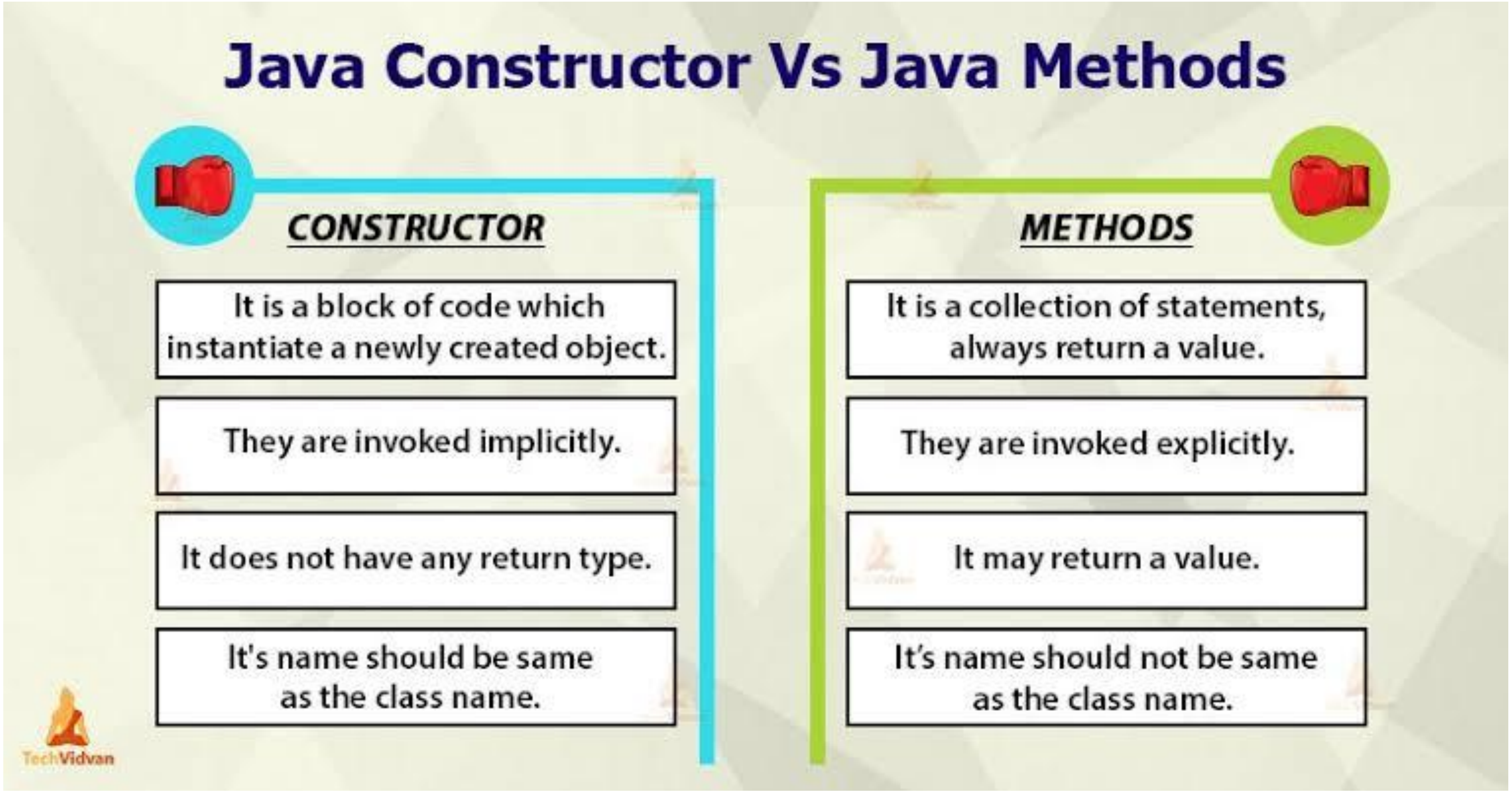
Visit www.PEDIAA.com


Types of Constructors
Default Constructor
Parameterized Constructor



```
public class MyClass{
    // Constructor
    MyClass(){
        System.out.println("BeginnersBook.com");
    }

    public static void main(String args[]){
        MyClass obj = new MyClass();
        ...
    }
}
```

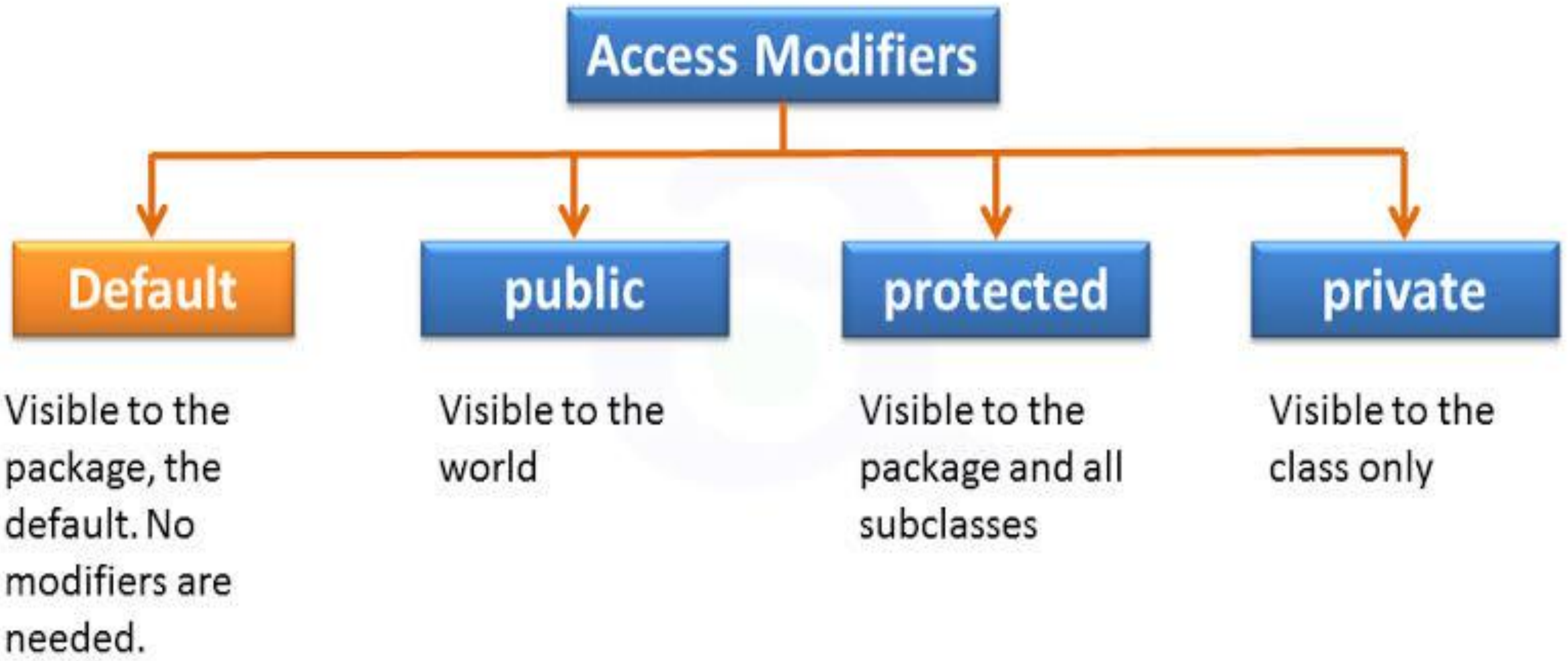
New keyword creates the object of MyClass & invokes the constructor to initialize the created object.

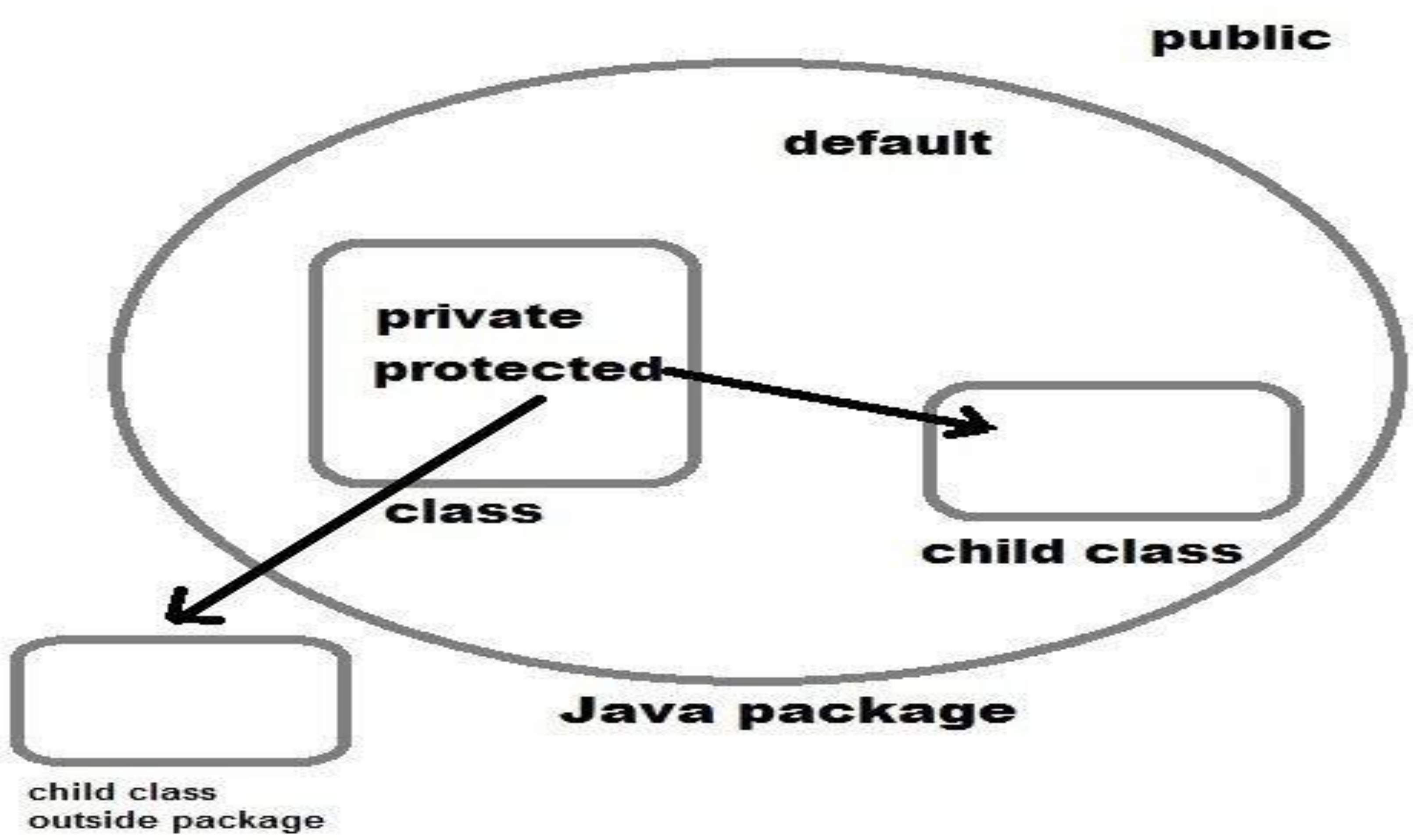


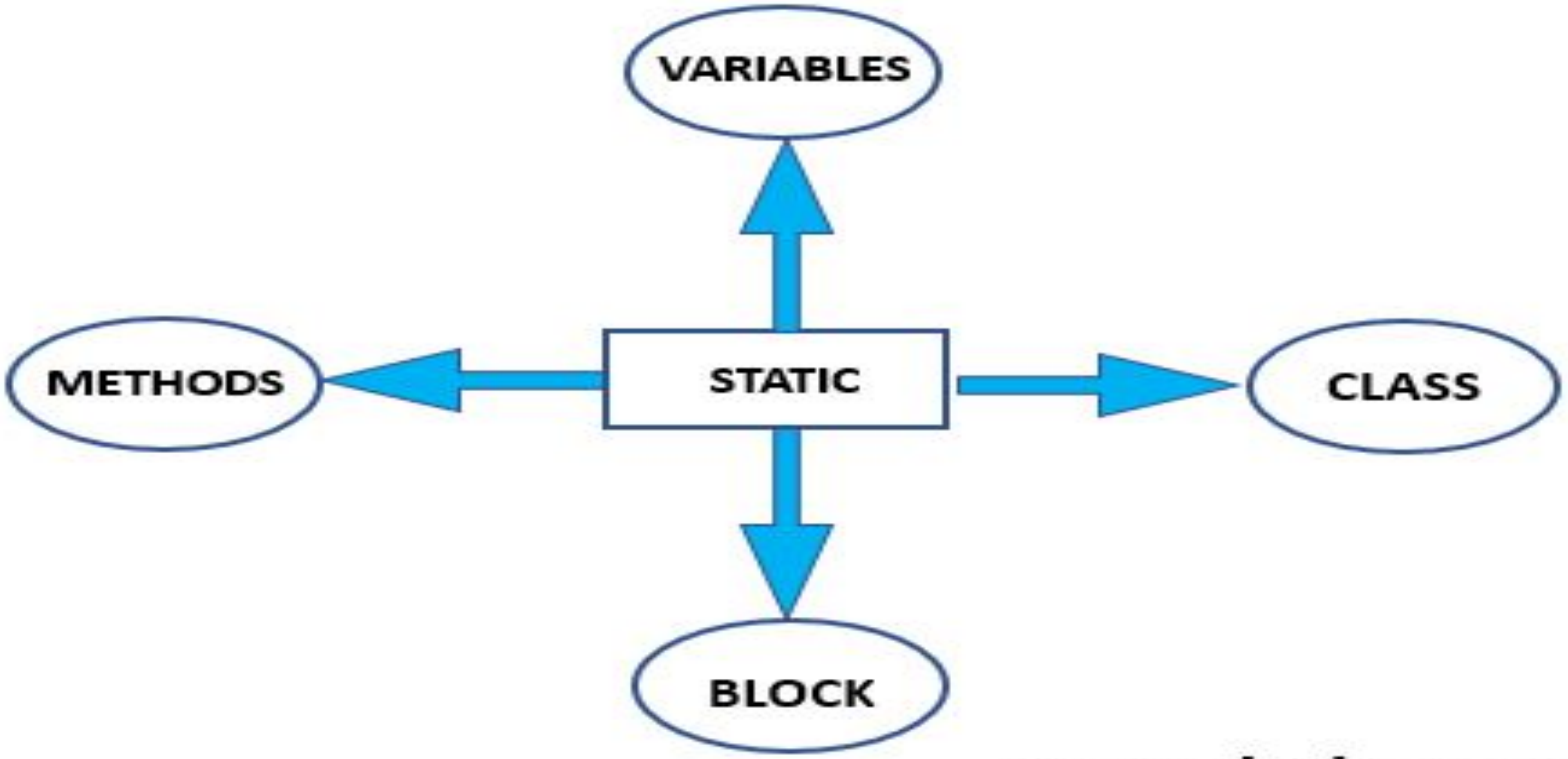
Java Constructor Vs Java Methods

CONSTRUCTOR	METHODS
It is a block of code which instantiate a newly created object.	It is a collection of statements, always return a value.
They are invoked implicitly.	They are invoked explicitly.
It does not have any return type.	It may return a value.
It's name should be same as the class name.	It's name should not be same as the class name.

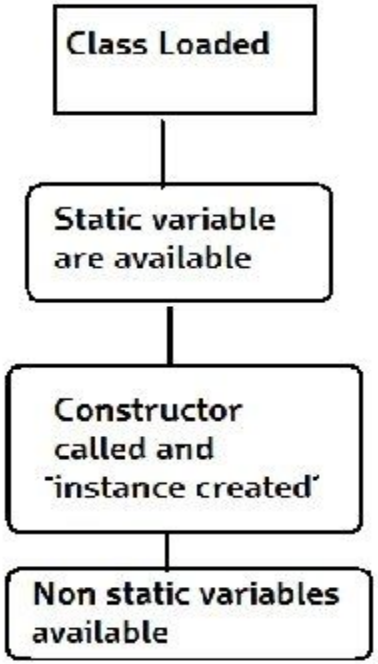
Access Modifiers	Non-Access Modifiers
private	static
default or No Modifier	final
protected	abstract
public	synchronized
	transient
	volatile
	strictfp


Access Modifiers
Default
public
protected
private
Visible to the package, the default. No modifiers are needed.
Visible to the world
Visible to the package and all subclasses
Visible to the class only


public
default
private
protected
class
child class
Java package
child class
outside package


VARIABLES
METHODS
STATIC
CLASS
BLOCK

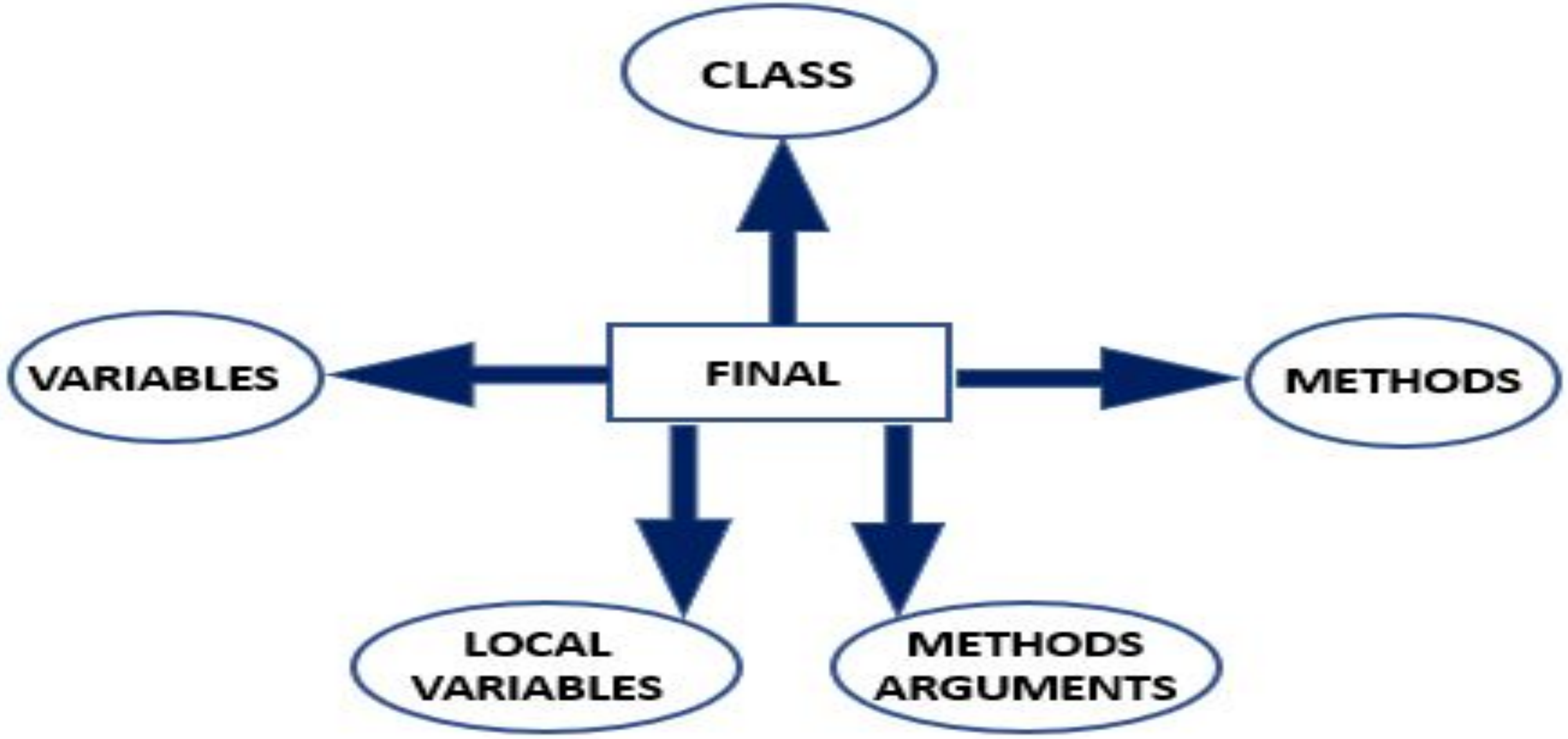
How it works

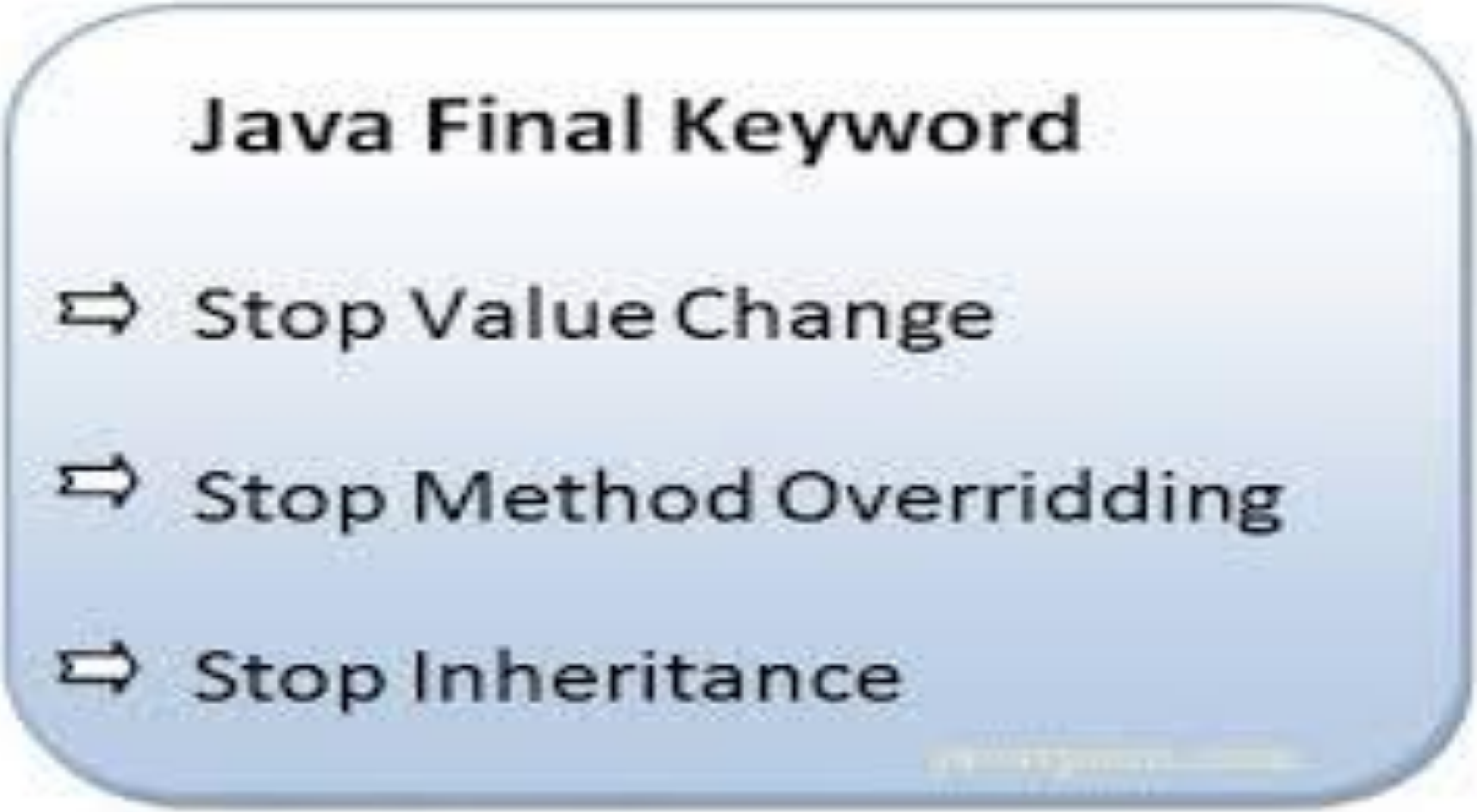

Class Loaded
Static variable are available
Constructor called and "instance created"
Non static variables available

Basic Steps of how objects are created

1. Class is loaded by JVM
2. Static variable and methods are loaded and initialized and available for use
3. Constructor is called to instantiate the non static variables
4. Non static variables and methods are now available

- As all the non static variable are available only after the constructor is called, there is a restriction on using non static variable in static methods.


CLASS
VARIABLES
FINAL
METHODS
LOCAL VARIABLES
METHODS ARGUMENTS

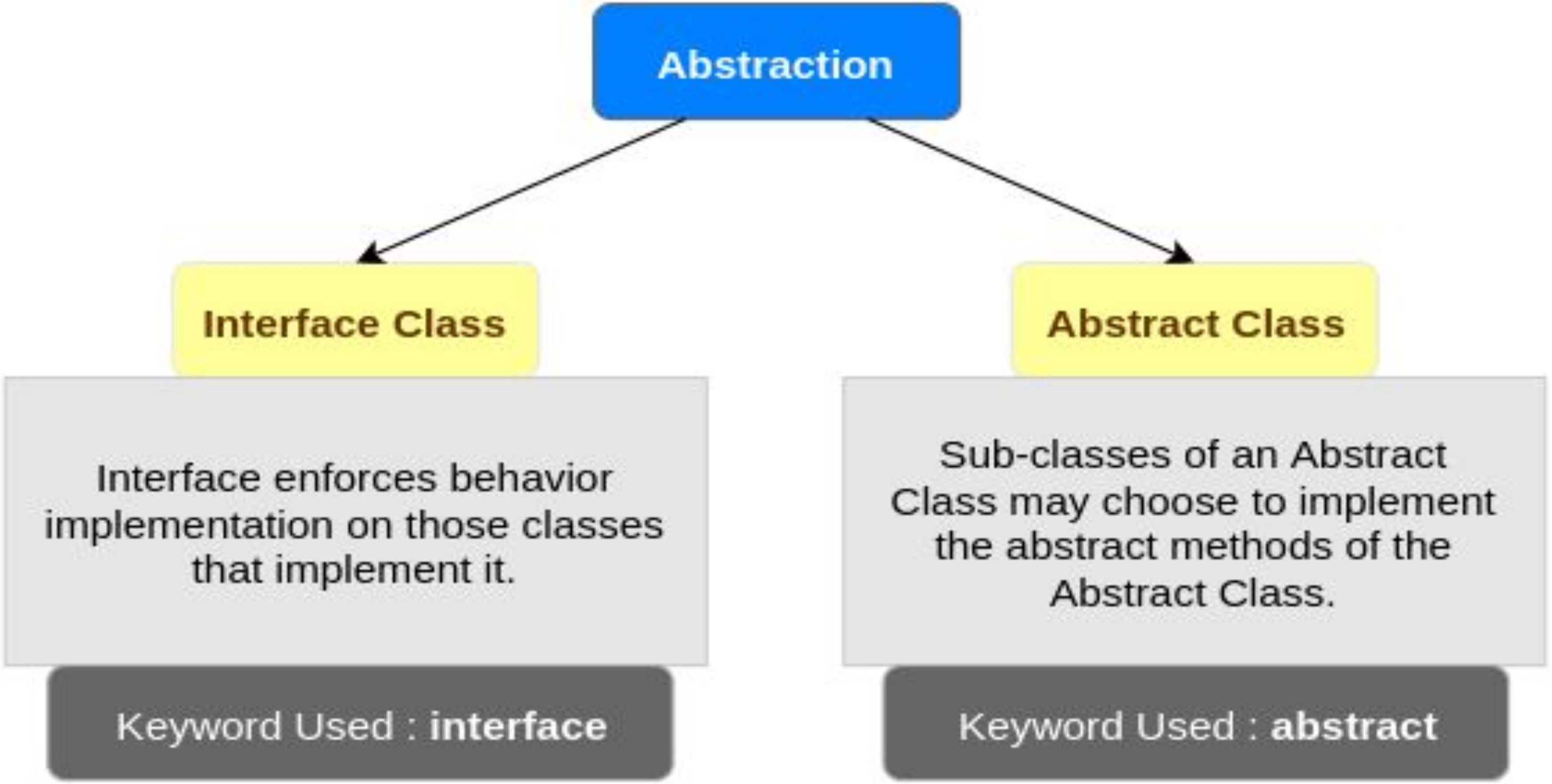


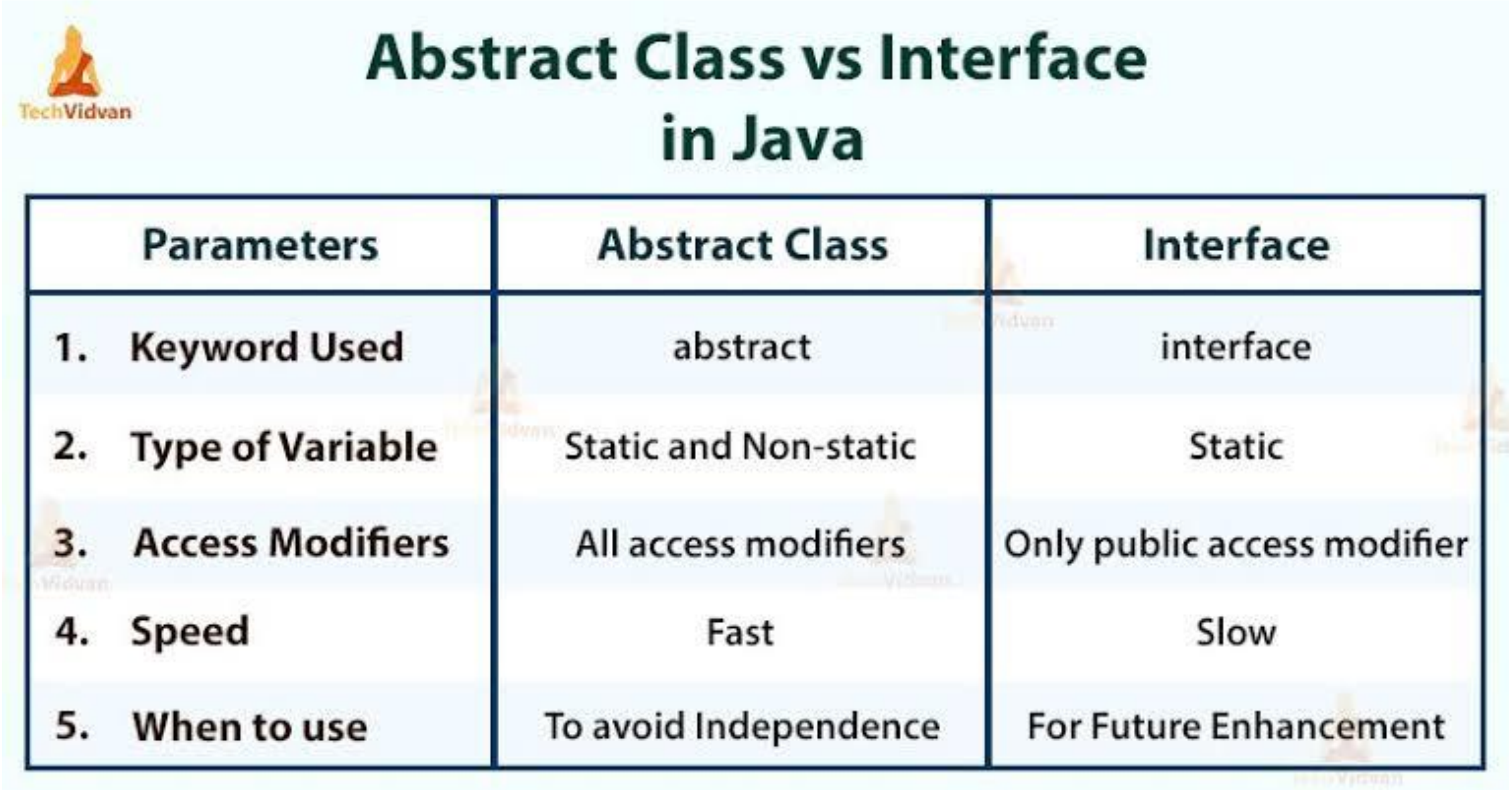
Java Final Keyword

⇨ Stop Value Change

⇨ Stop Method Overridding

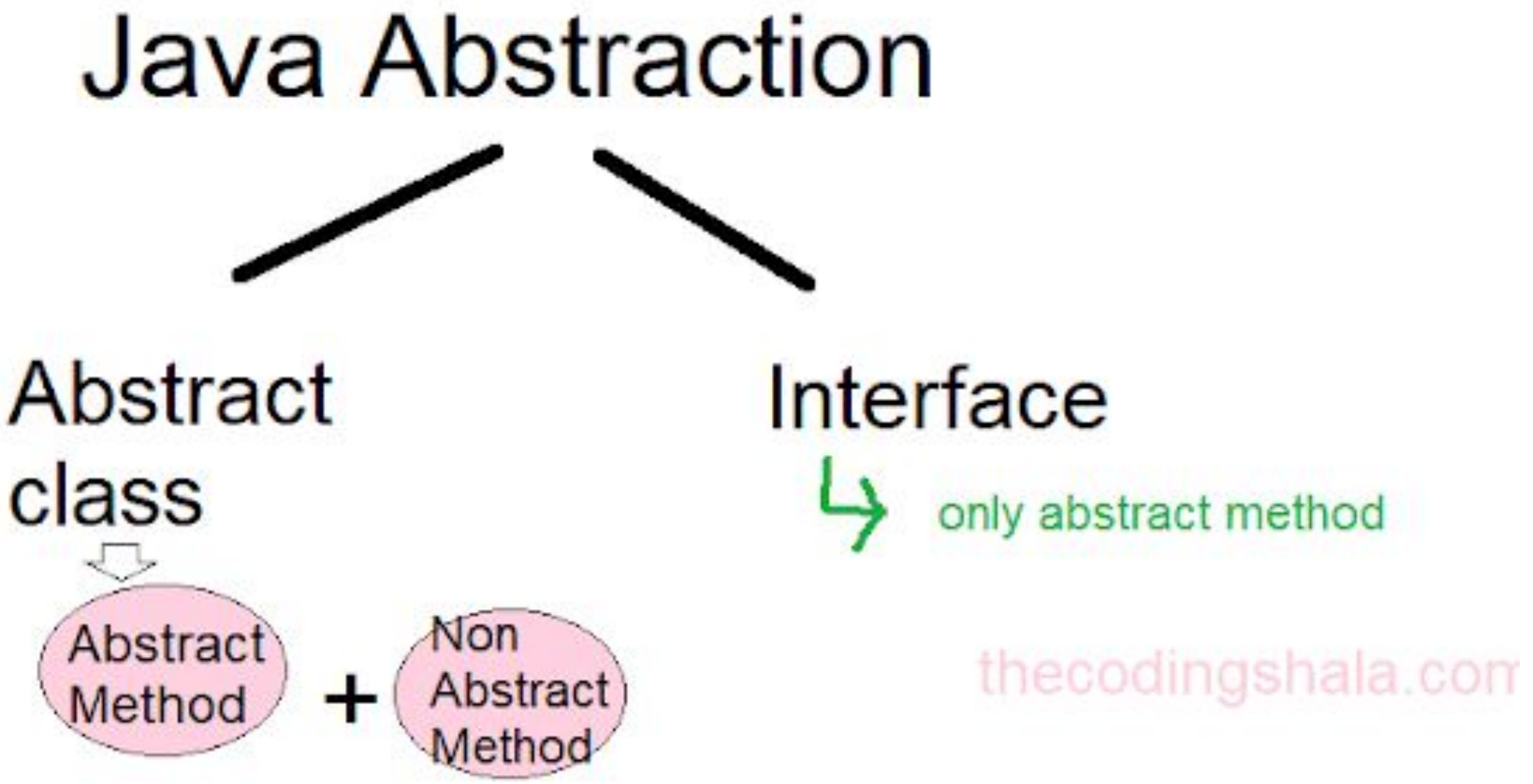
⇨ Stop Inheritance

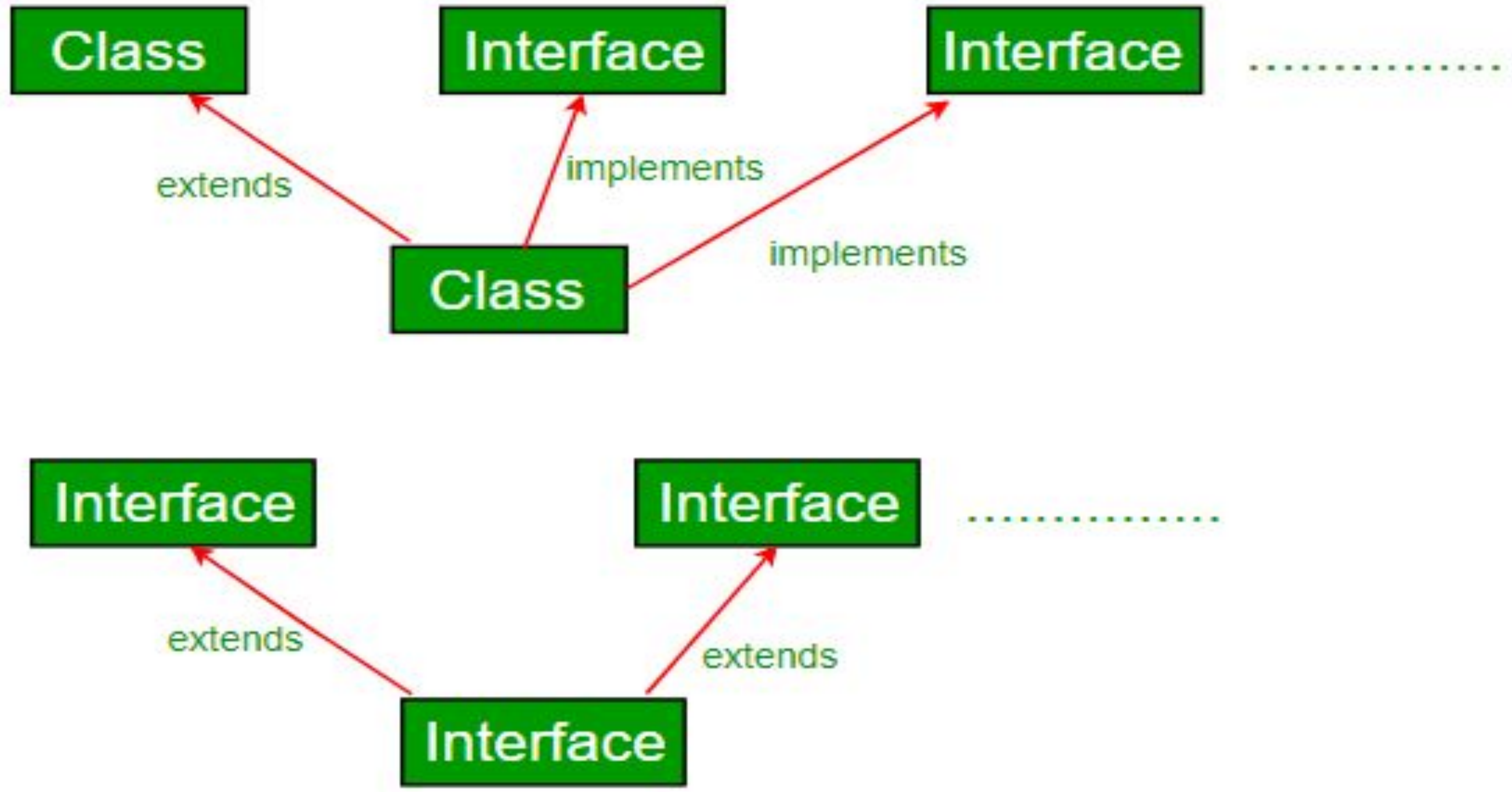

Abstraction
Interface Class
Abstract Class
Interface enforces behavior implementation on those classes that implement it.
Sub-classes of an Abstract Class may choose to implement the abstract methods of the Abstract Class.
Keyword Used : **interface**
Keyword Used : **abstract**



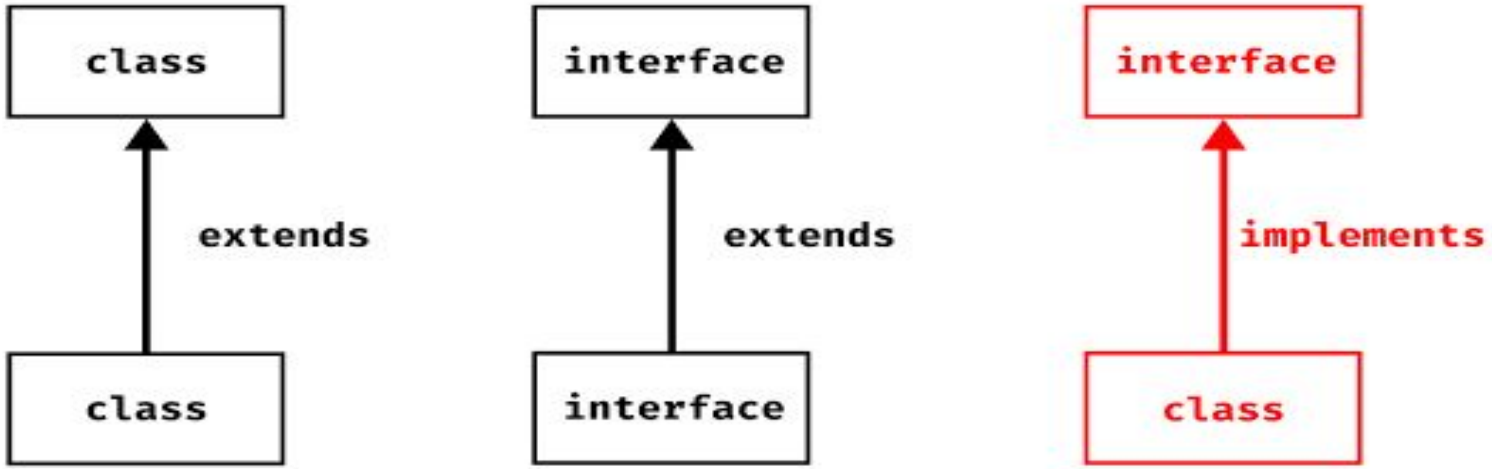
Abstract Class vs Interface in Java

Parameters	Abstract Class	Interface
1. Keyword Used	abstract	interface
2. Type of Variable	Static and Non-static	Static
3. Access Modifiers	All access modifiers	Only public access modifier
4. Speed	Fast	Slow
5. When to use	To avoid Independence	For Future Enhancement


Java Abstraction
Abstract class
Interface
only abstract method
Abstract Method
+
Non Abstract Method
thecodingshala.com

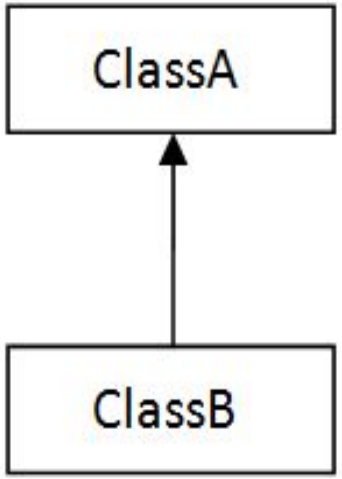

Class
Interface
Interface
....
extends
implements
Class
implements
Interface
Interface
....
extends
extends
Interface

EXTENDS VS IMPLEMENTS JAVA

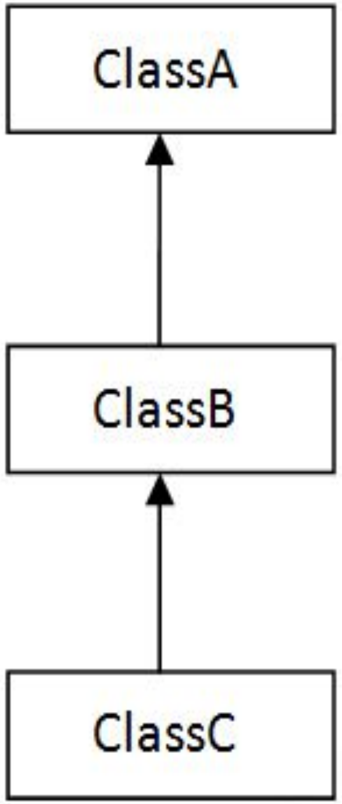


extends keyword is used for extending classes and interfaces

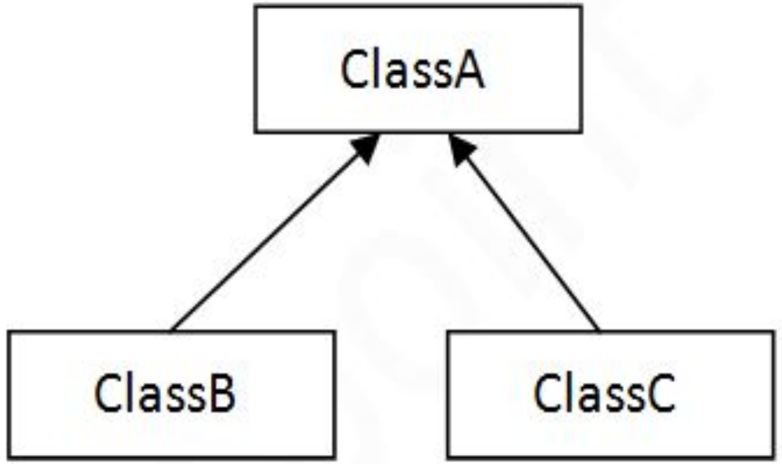
implements keyword is used for implementing interfaces to a class



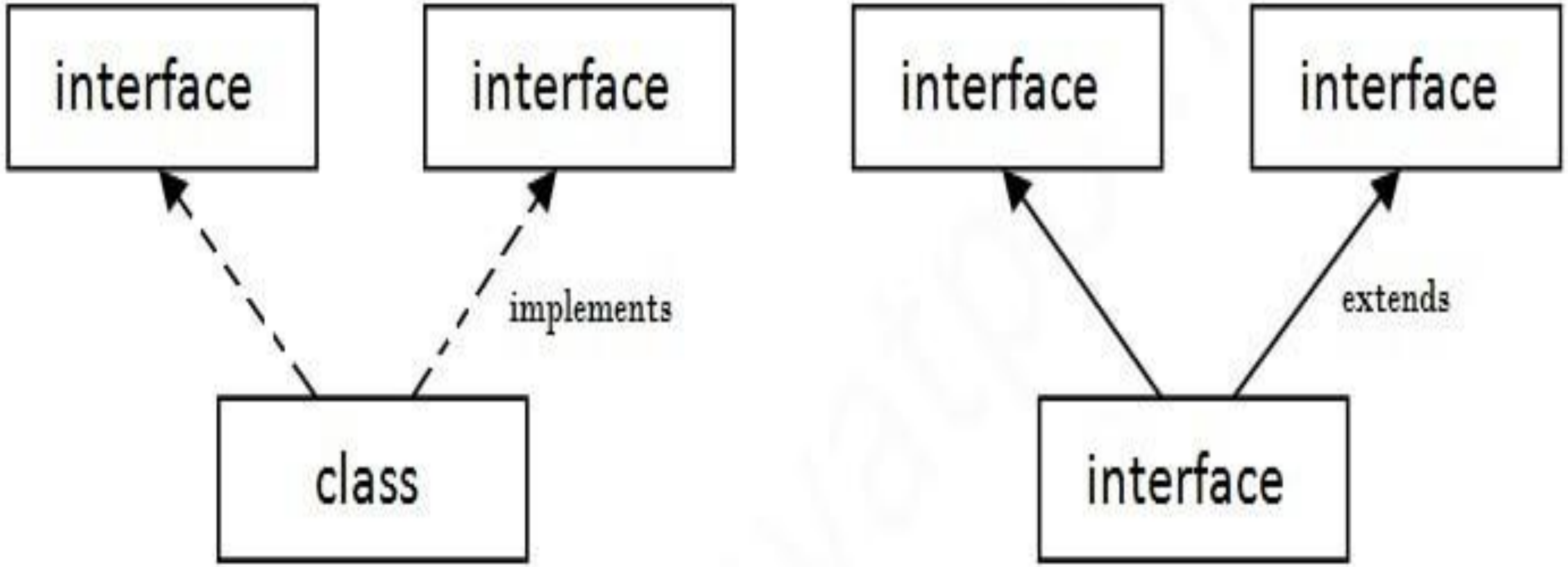
1) Single



2) Multilevel



3) Hierarchical



Multiple Inheritance in Java

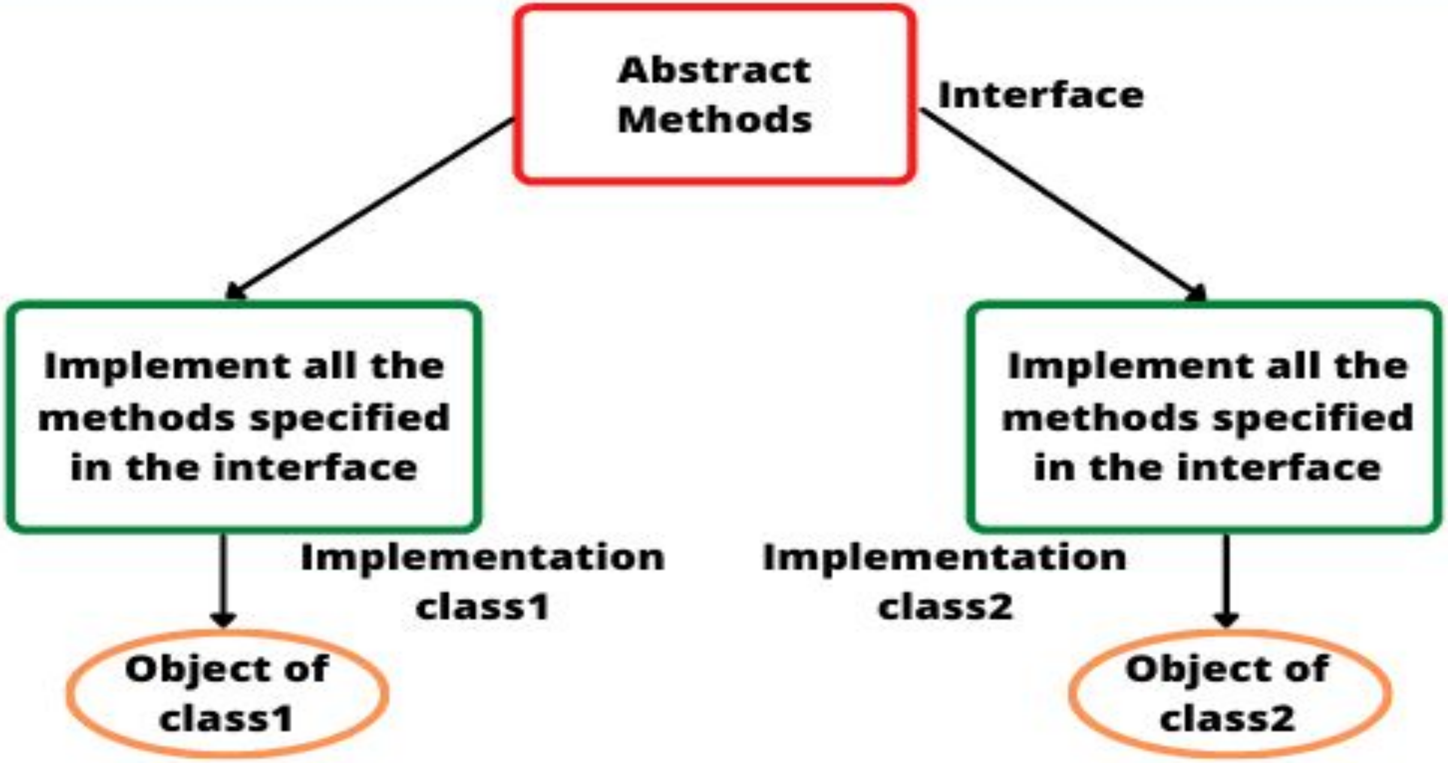
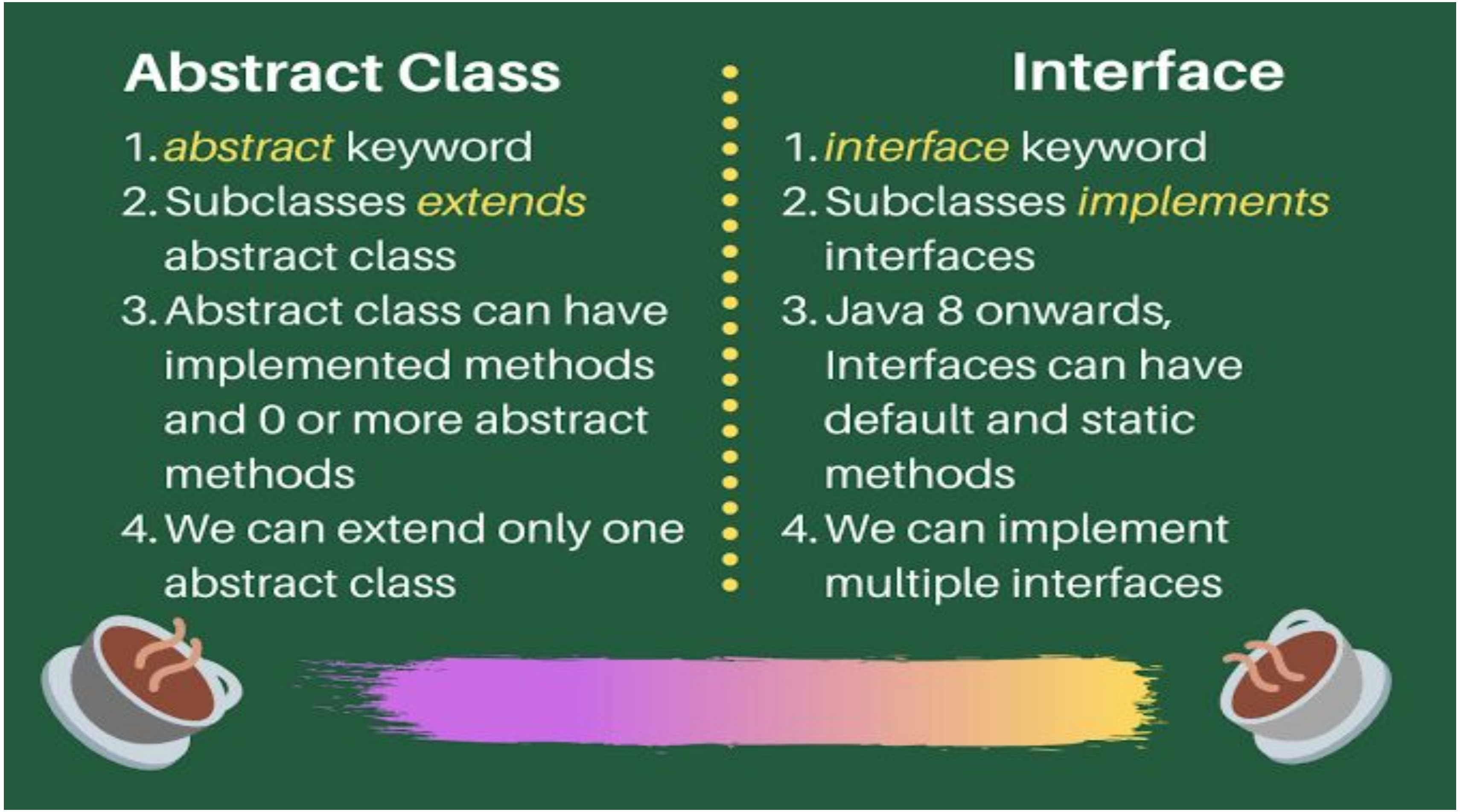
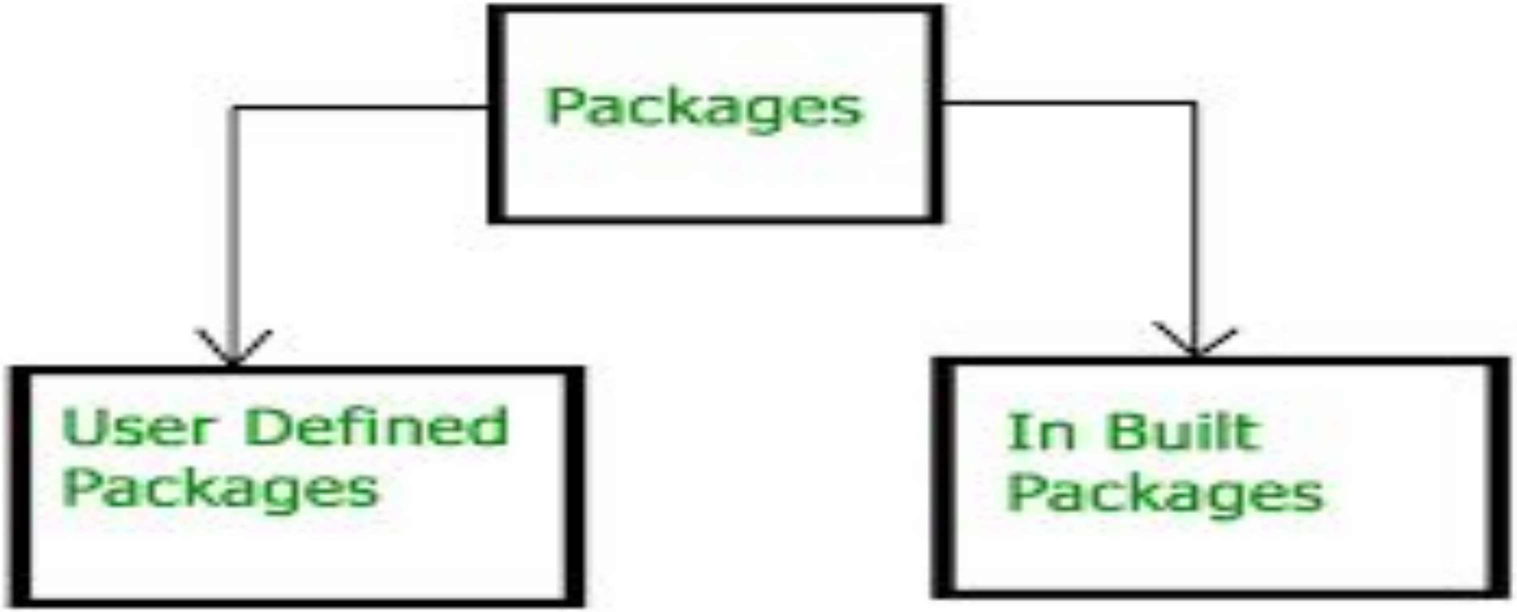
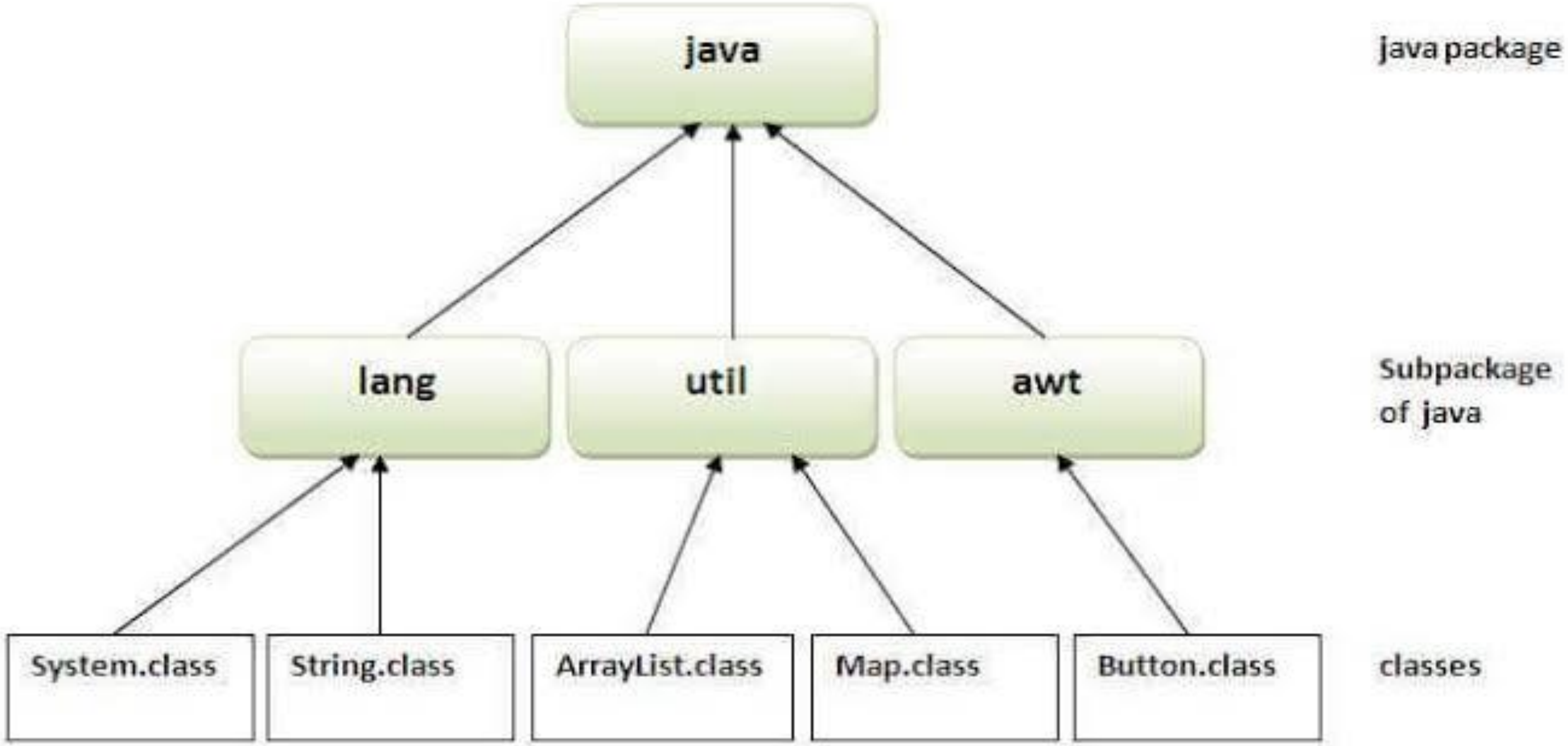

Abstract Methods
Interface
Implement all the methods specified in the interface
Implement all the methods specified in the interface
Implementation class1
Implementation class2
Object of class1
Object of class2

Fig: Interface and its Implementation Classes



Abstract Class	Interface
1. *abstract* keyword	1. *interface* keyword
2. Subclasses *extends* abstract class	2. Subclasses *implements* interfaces
3. Abstract class can have implemented methods and 0 or more abstract methods	3. Java 8 onwards, Interfaces can have default and static methods
4. We can extend only one abstract class	4. We can implement multiple interfaces


Packages
User Defined Packages
In Built Packages


java
java package
lang
util
awt
Subpackage of java
System.class
String.class
ArrayList.class
Map.class
Button.class
classes

Steps For Creating Package :

To create a user defined package the following steps should be involved :-

1: Declare the package at the beginning of a file using the syntax :-

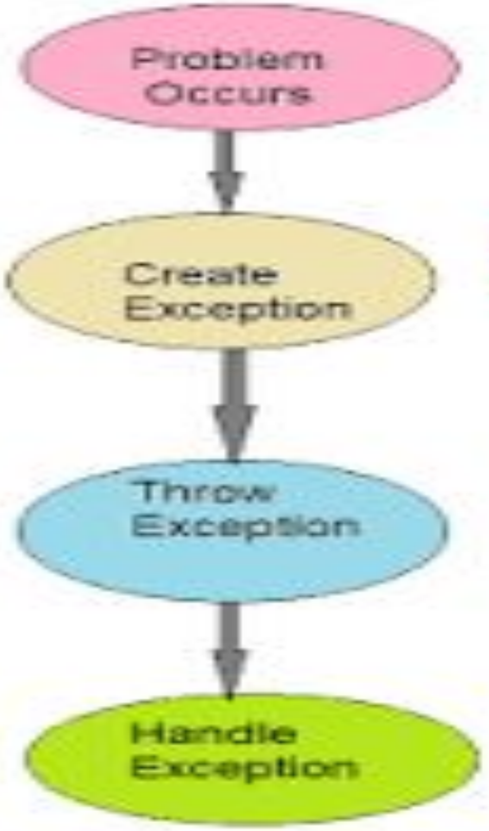
```
package packageName;
```

2: Define the class that is to be put in the package & declare it public.

3: Create a subdirectory under the directory where the main source files are stored.

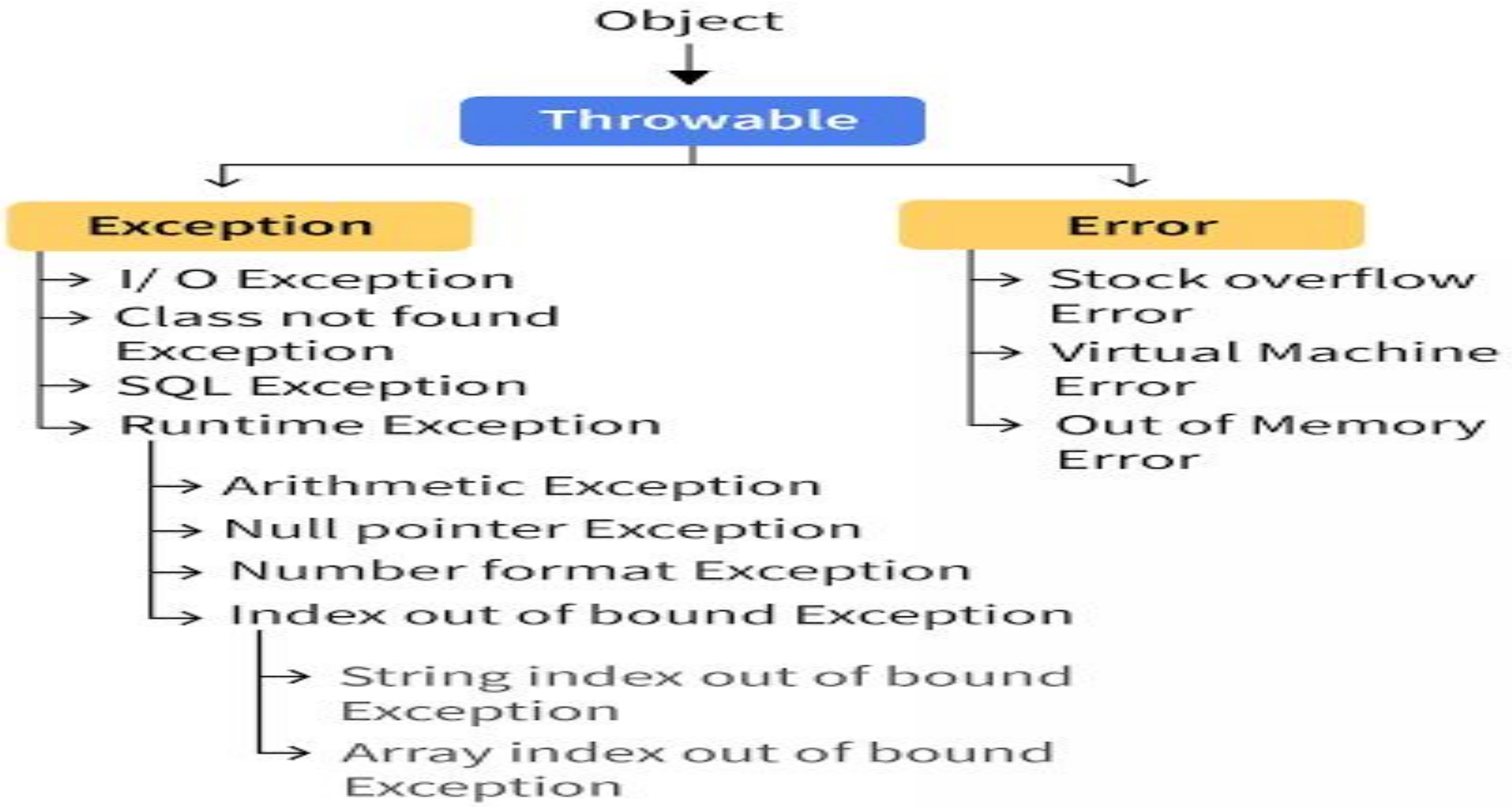
4: Store the listing as the classname.java file in the subdirectory created.

5: Compile the file. This create .class file in the subdirectory.



Exception Handling in Java

try
catch
throw
throws
finally



The error indicates trouble that primarily occurs due to the scarcity of system resources. The exceptions are the issues that can appear at runtime and compile time.

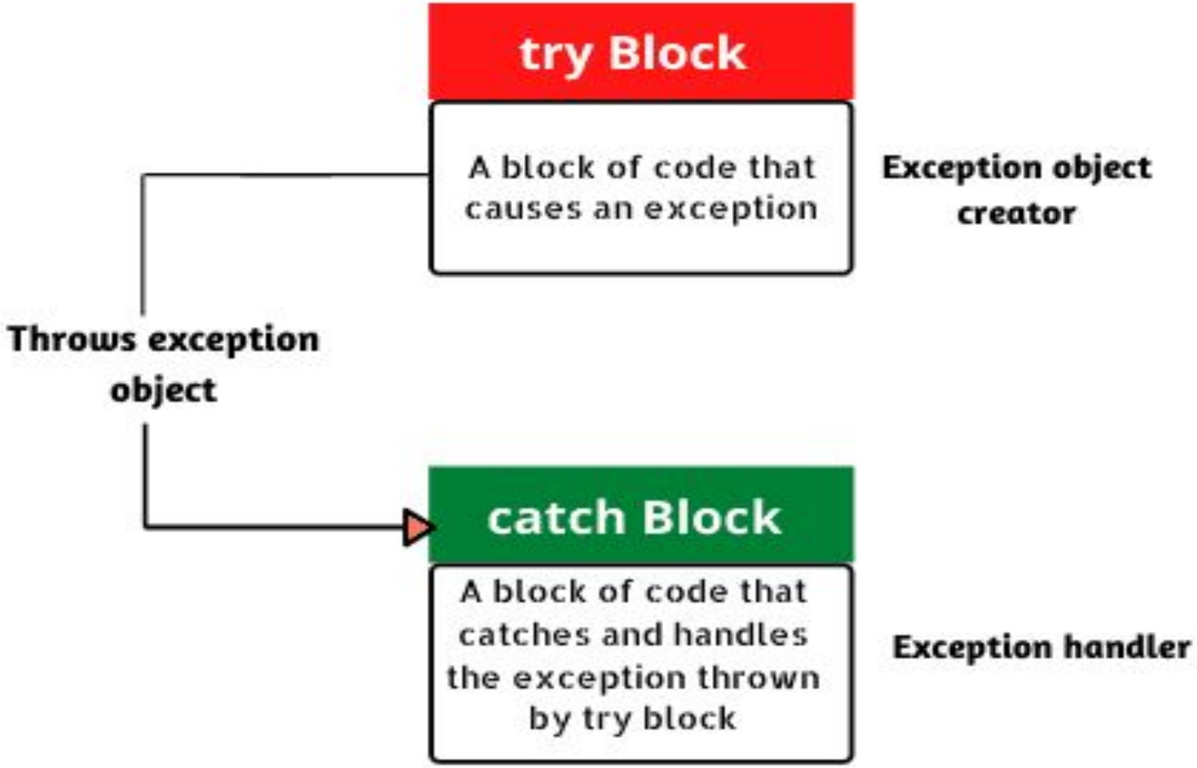
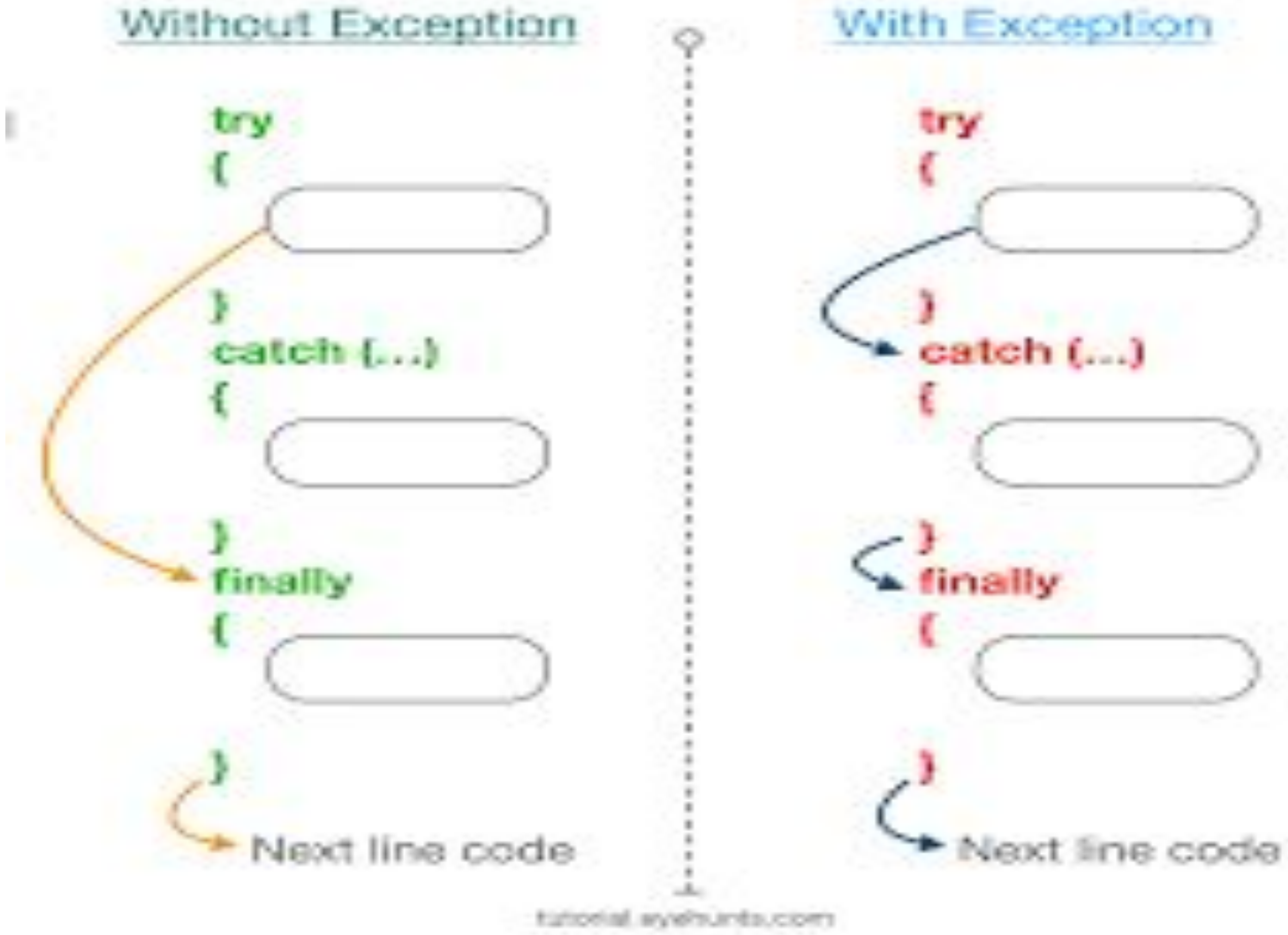
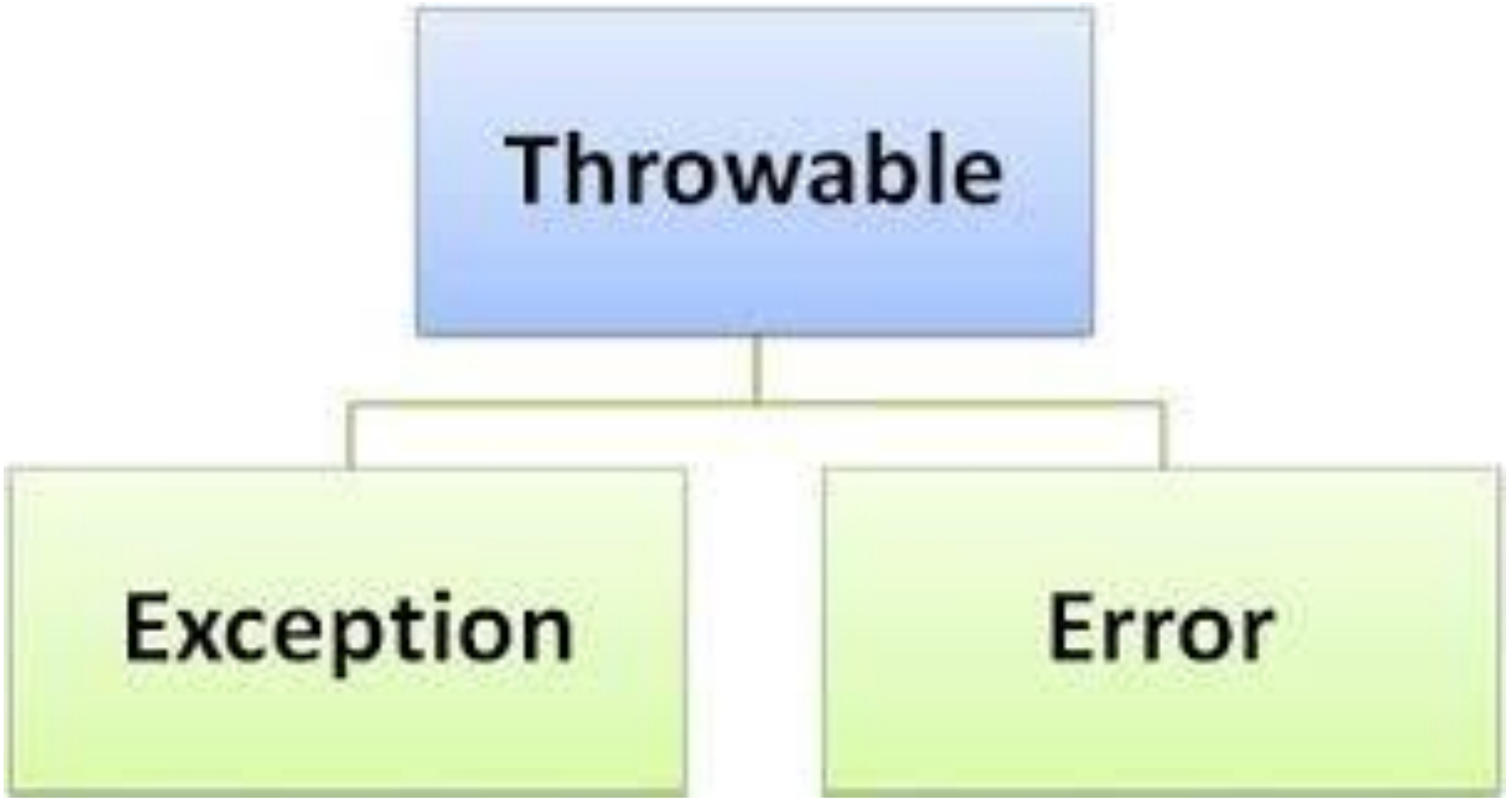
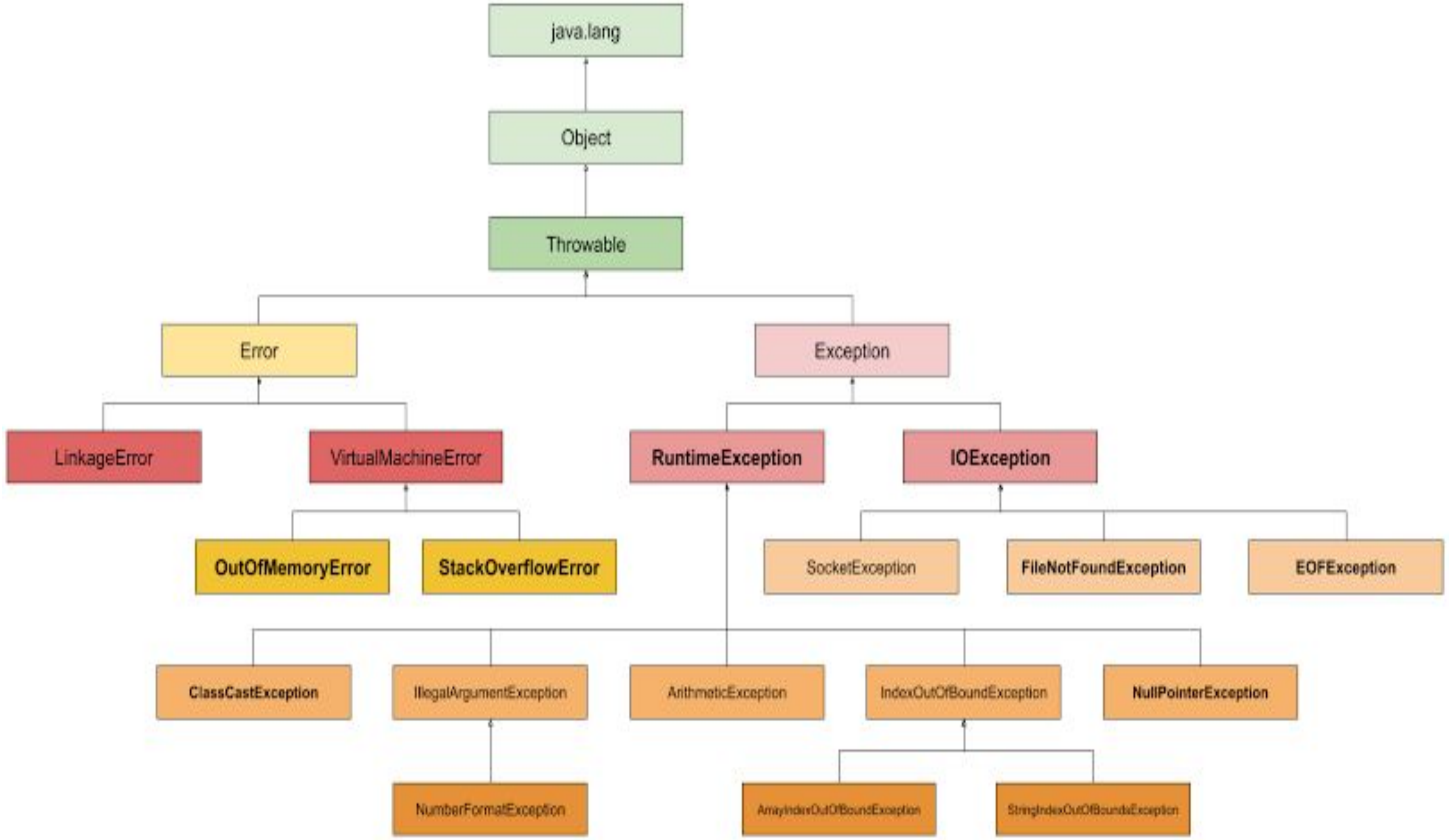

try Block
A block of code that causes an exception
Exception object creator
Throws exception object
catch Block
A block of code that catches and handles the exception thrown by try block
Exception handler

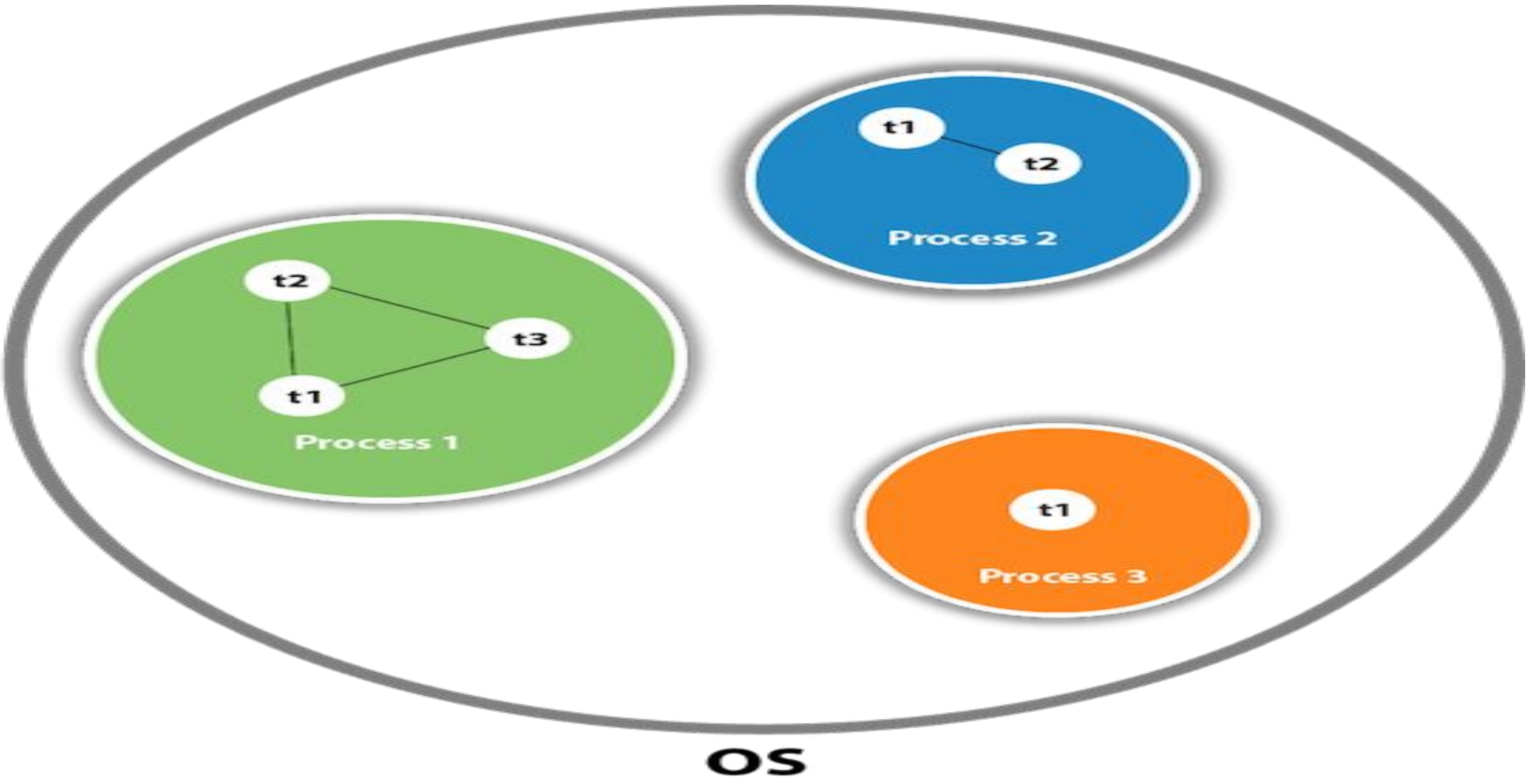
Fig: Exception handling mechanism


Without Exception
try
{
}
catch (...)
{
}
finally
{
}
Next line code
With Exception
try
{
}
catch (...)
{
}
finally
{
}
Next line code
tutorial.eyehunts.com


Throwable
Exception
Error


java.lang
Object
Throwable
Error
Exception
LinkageError
VirtualMachineError
RuntimeException
IOException
OutOfMemoryError
StackOverflowError
SocketException
FileNotFoundException
EOFException
ClassCastException
IllegalArgumentException
ArithmeticException
IndexOutOfBoundException
NullPointerException
NumberFormatException
ArrayIndexOutOfBoundException
StringIndexOutOfBoundsException

Multithreading



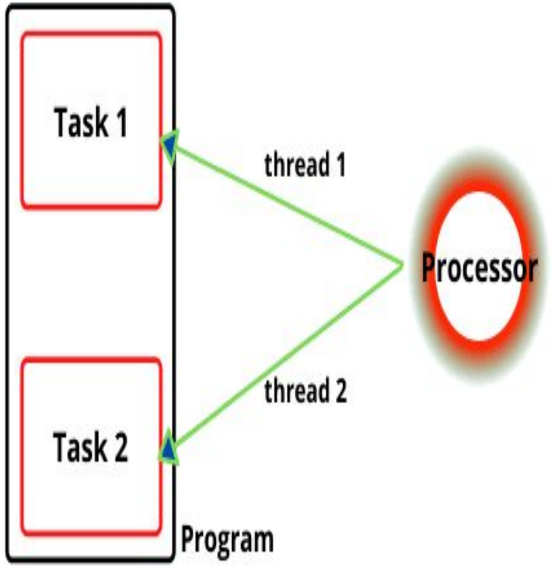
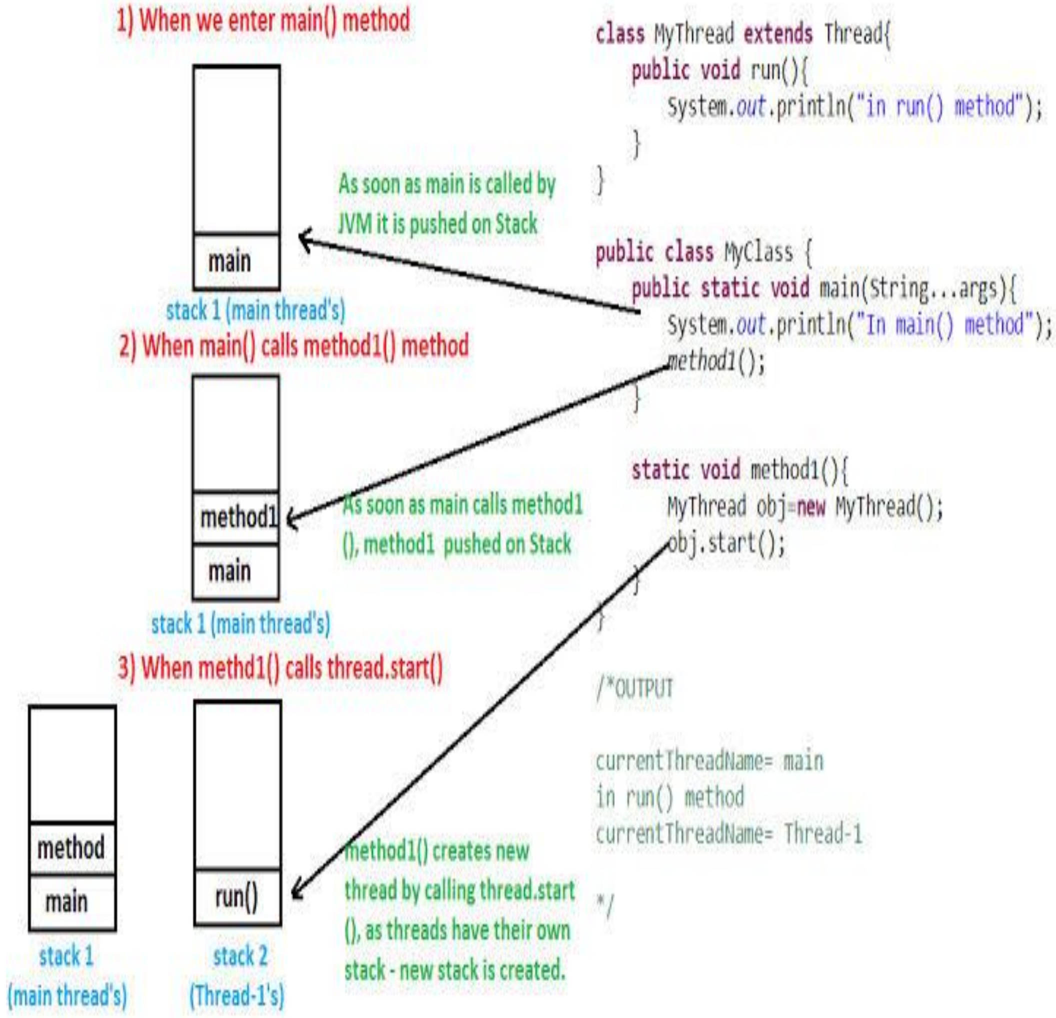


Fig: Thread-based Multitasking


1) When we enter main() method
main
stack 1 (main thread's)
As soon as main is called by JVM it is pushed on Stack
2) When main() calls method1() method
method1
main
stack 1 (main thread's)
As soon as main calls method1 (), method1 pushed on Stack
3) When methd1() calls thread.start()
method
main
stack 1 (main thread's)
run()
stack 2 (Thread-1's)
method1() creates new thread by calling thread.start (), as threads have their own stack - new stack is created.

```java
class MyThread extends Thread{
    public void run(){
        System.out.println("in run() method");
    }
}

public class MyClass {
    public static void main(String...args){
        System.out.println("In main() method");
        method1();
    }

    static void method1(){
        MyThread obj=new MyThread();
        obj.start();
    }
}

/*OUTPUT

currentThreadName= main
in run() method
currentThreadName= Thread-1

*/
```

How to create Thread in Java

Two ways to create thread in Java

- By extending Thread class
- By implementing Runnable Interface

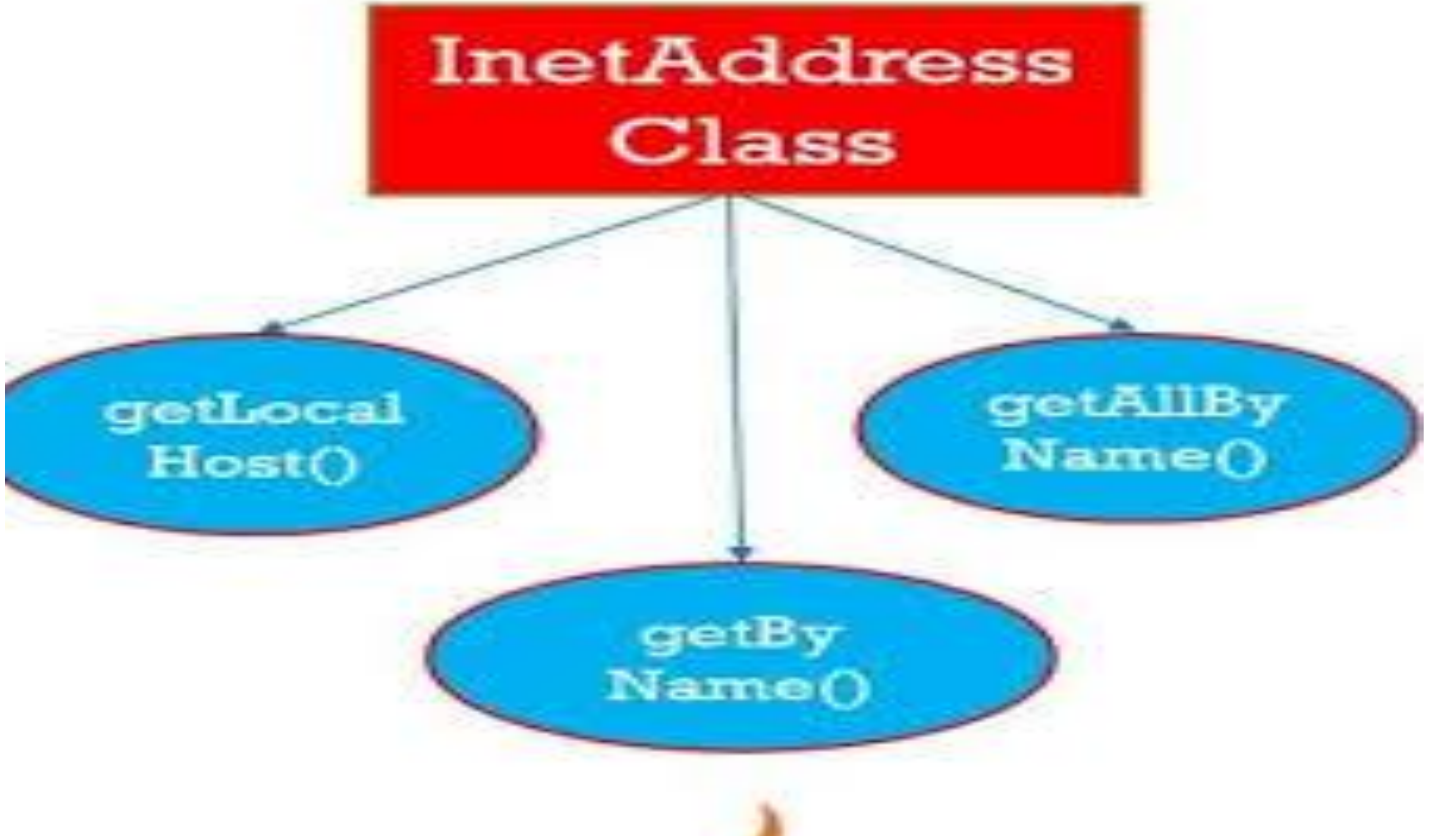
How to create thread

- There are two ways to create a thread:
  - By extending Thread class
  - By implementing Runnable interface.
- **Thread class:**
  - Thread class provide constructors and methods to create and perform operations on a thread.
- **Constructors of Thread class:**
  - Thread()
  - Thread(String name)
  - Thread(Runnable r)
  - Thread(Runnable r,String name)

Java enum

TopJavaTutorial.com

- Declare enum
- Accessing enums
- compare enums
- enum with body
- enum singleton


InetAddress Class
getLocal Host()
getAllBy Name()
getBy Name()

Internet Addresses

◆ Transmission Control Protocol (TCP) : To obtain reliable, sequenced data exchange.

◆ User Datagram Protocol (UDP) : To obtain a more efficient, best-effort delivery.

◆ GetByName() Method

static InetAddress getByName (String hostName) throws UnknownHostException

- ***Determines the IP address of a host, given the host's name.***

◆ getAllByName() Method

static InetAddress[] getAllByName (String hostName) throws UnknownHostException

- ***Given the name of a host, returns an array of its IP addresses, based on the configured name service on the system.***

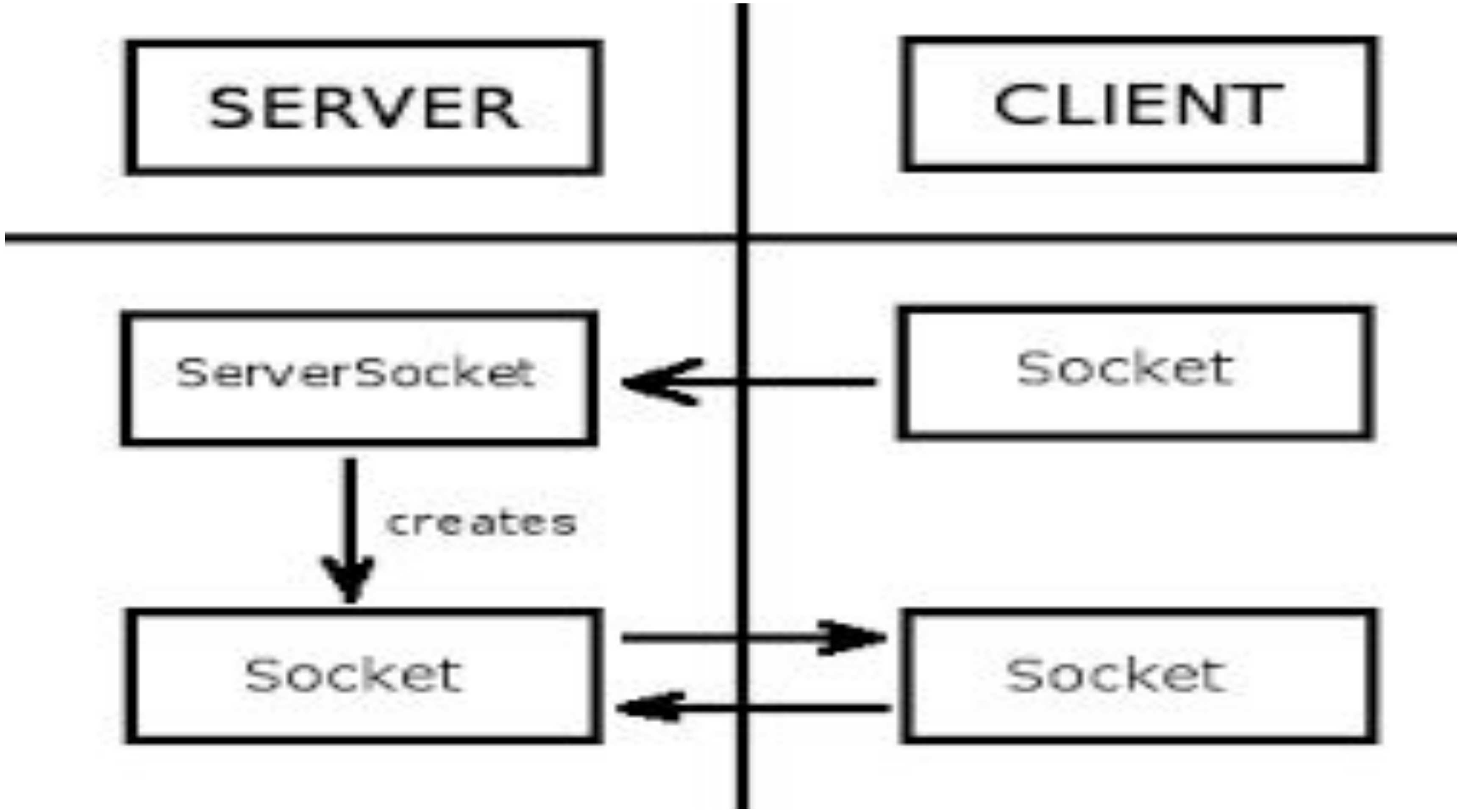
◆ getAddress() Method

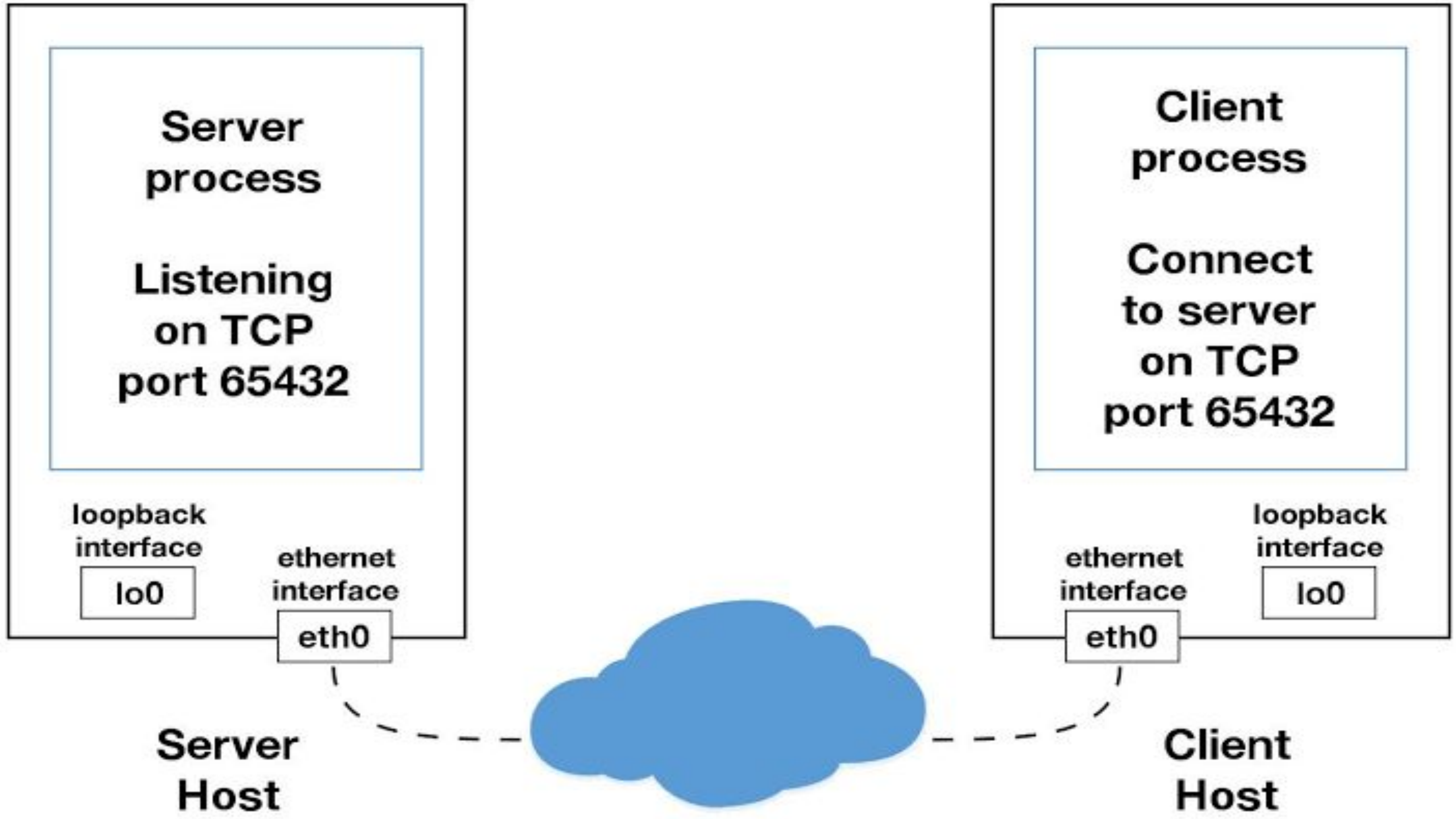
Byte[] InetAddress getLocalHost()

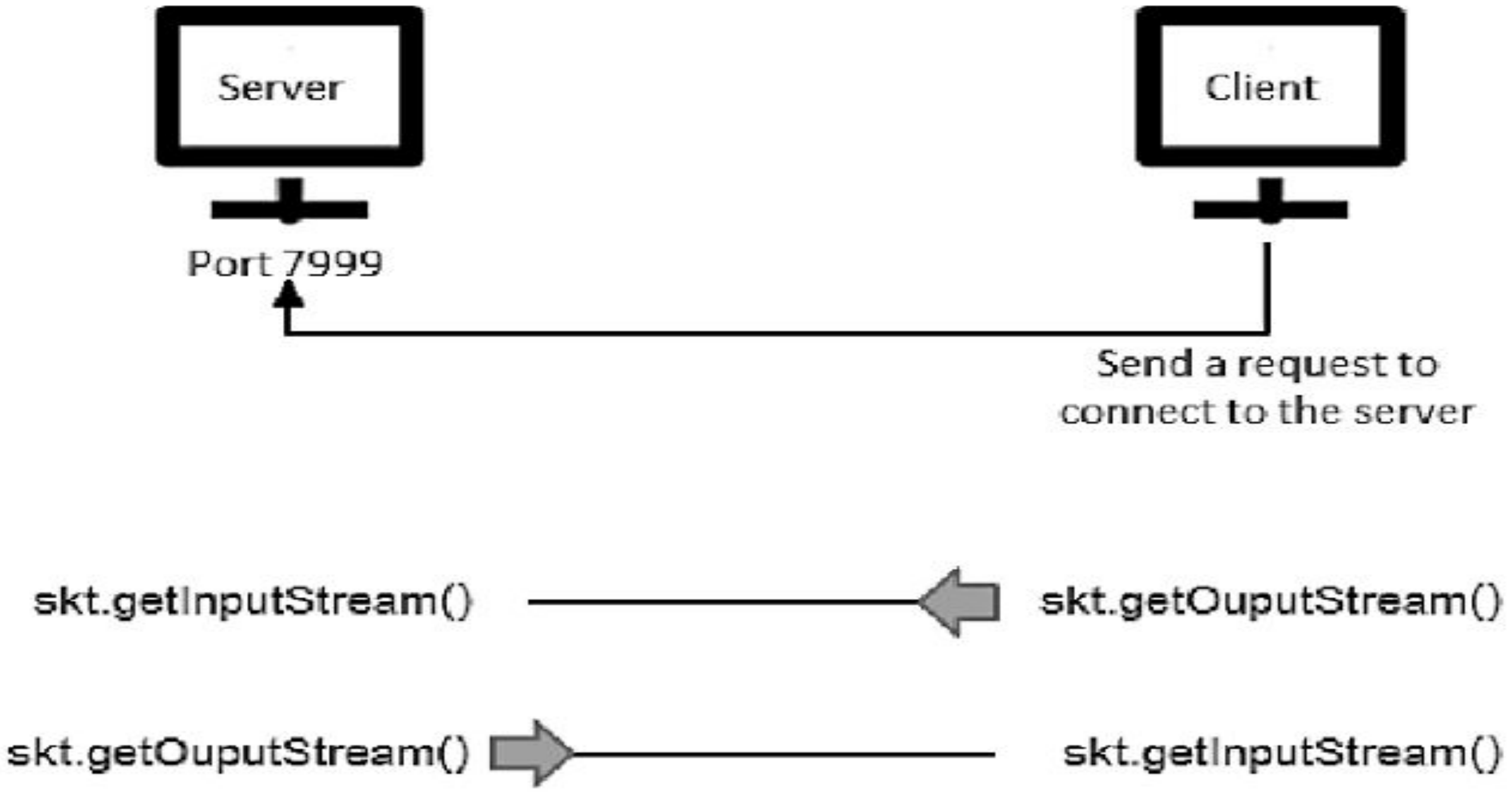
- **Returns the raw IP address of this InetAddress object.**

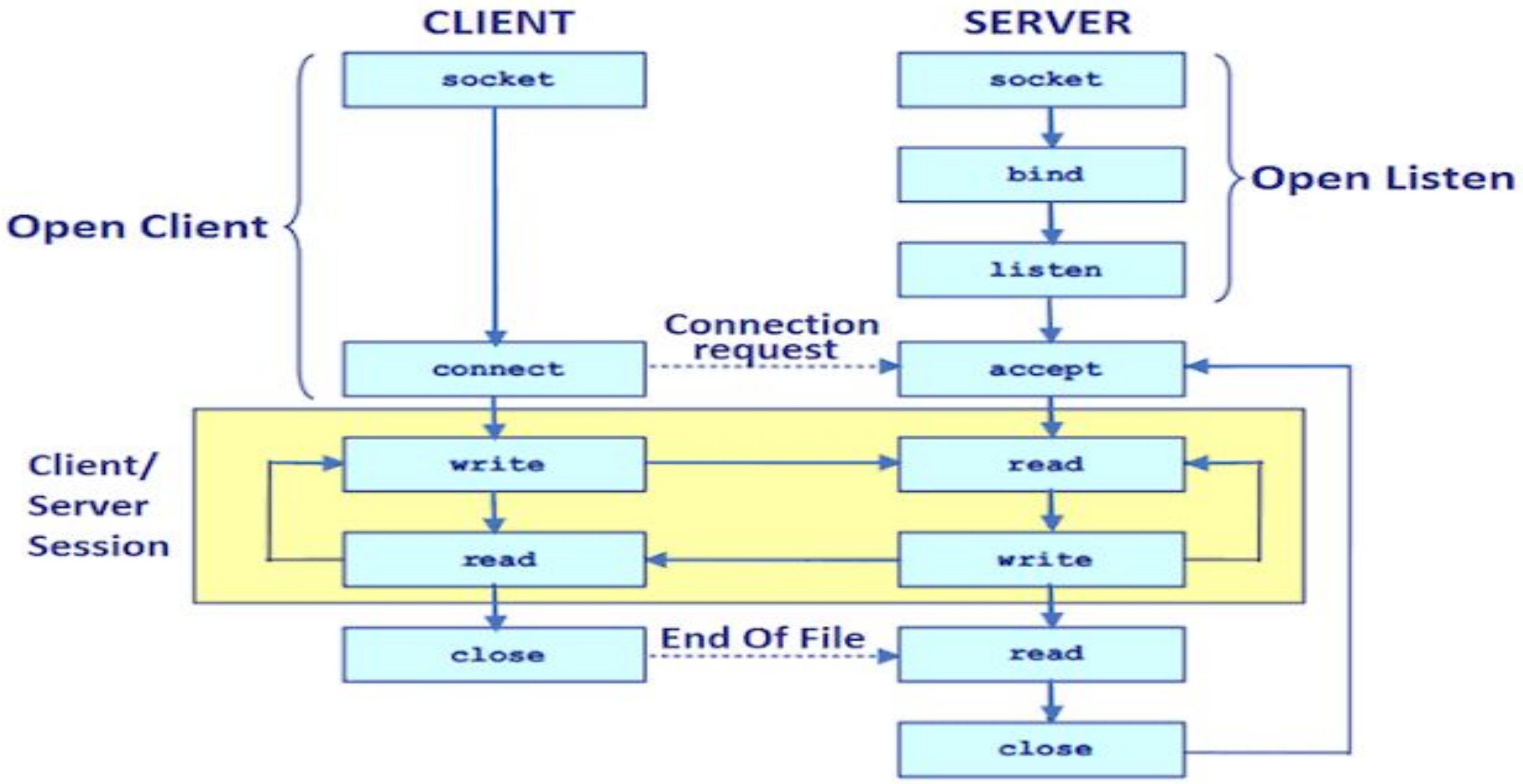
```
import java.net.*;

class InetAddressDemo {
  public static void main(String args[]) {
    try {
      InetAddress ias[] =
        InetAddress.getAllByName(args[0]);
      for (int i = 0; i < ias.length; i++) {
        System.out.println(ias[i].getHostName());
        System.out.println(ias[i].getHostAddress());
        byte bytes[] = ias[i].getAddress();
        for (int j = 0; j < bytes.length; j++) {
          if (j > 0)
            System.out.print(".");
          if (bytes[j] >= 0)
            System.out.print(bytes[j]);
          else
            System.out.print(bytes[j] + 256);
        }
        System.out.println("");
      }
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

http://java.sun.com/javase/ja/8/docs/ja/api/java/net/InetAddress.html


SERVER
CLIENT
ServerSocket
Socket
creates
Socket
Socket


Server process
Listening on TCP port 65432
loopback interface
lo0
ethernet interface
eth0
Server Host
Client process
Connect to server on TCP port 65432
ethernet interface
eth0
loopback interface
lo0
Client Host

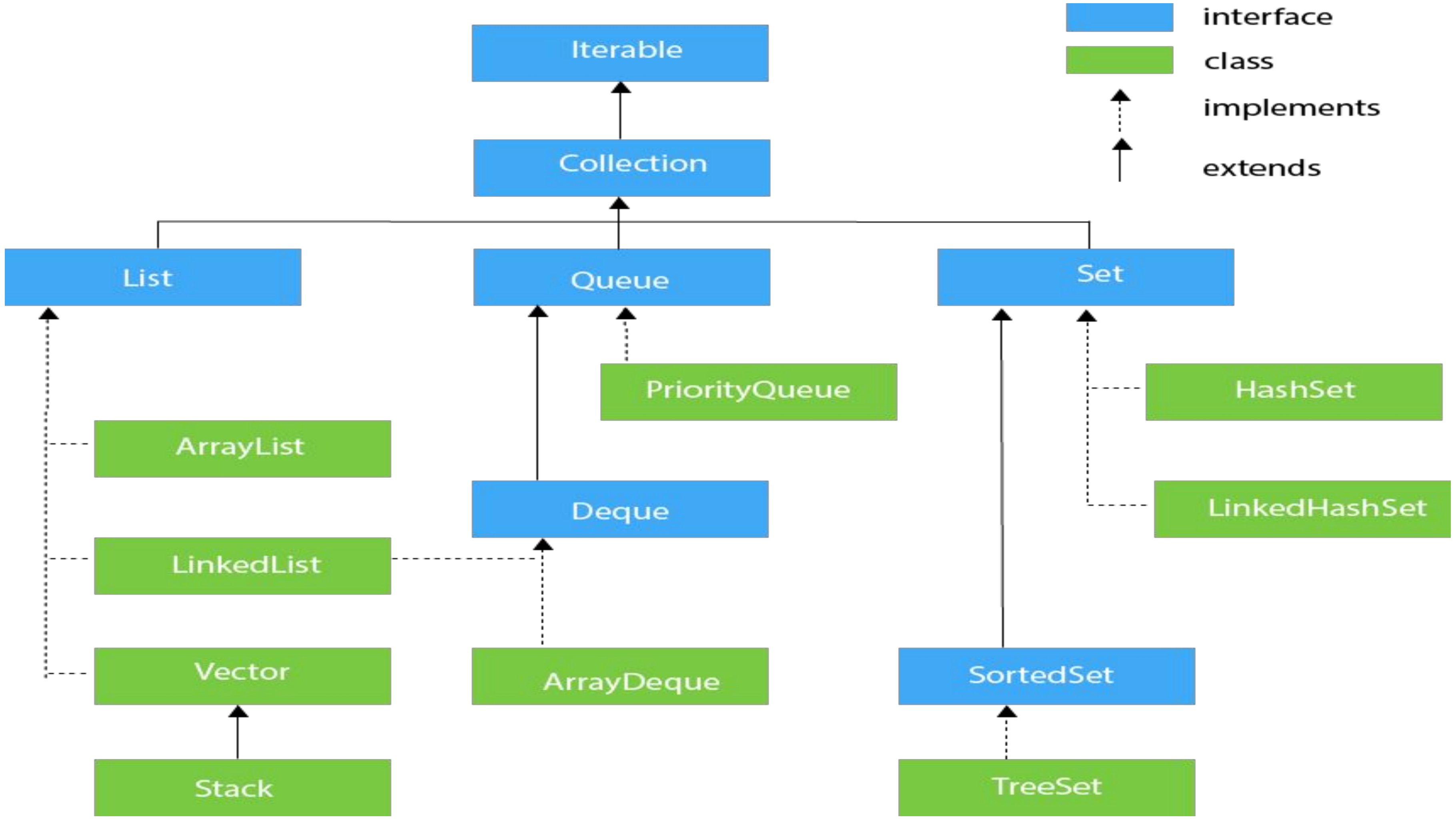

Server
Client
Port 7999
Send a request to connect to the server
skt.getInputStream()
skt.getOuputStream()
skt.getOuputStream()
skt.getInputStream()


CLIENT
SERVER
Open Client
Open Listen
socket
socket
bind
listen
Connection request
connect
accept
Client/ Server Session
write
read
read
write
End Of File
close
read
close

SOCKET API

Benefits of a Collection Framework

- Reduces programming effort
  - Powerful data structures and algorithms
- Increases program speed and quality
  - High quality implementations
  - Fine tuning by switching implementations
- Reduces the effort of learning new APIs
  - Uniformity of the framework
  - APIs of applications
- Encourages software reuse
  - New data structures and algorithms


interface
class
implements
extends
Iterable
Collection
List
Queue
Set
PriorityQueue
HashSet
ArrayList
Deque
LinkedHashSet
LinkedList
Vector
ArrayDeque
SortedSet
Stack
TreeSet