Types Of Applications:

1) Console Based Applications (Core Java & Embedded Systems) Less than 5%

2) Desktop / Standalone Applications (Core Java===> Applets and Swings) Less than 20%

3) Web-Based Applications (Core Java, Advance Java,Spring, Hibernate, SQL, Devops, Testing, UI) 80%

Servers:
1)Apache Tomcat
2)Jboss
3)Glass Fish

Editions Of Java:

1) Java 2 Standard Edition (J2SE)==> Core Java

2) Java 2 Enterprise Edition (J2EE)==> Advance Java [JDBC, Servlets, JSP] (SQL quaries)

3) Java 2 Micro Edition (J2ME)==> Mobile Edition (Andriod, PDA)

Access Modifiers (4)
1) public
2) private
3) protected
4) default

Basic Java Programming elements: 3

class

```
<AccessModifier>class<ClassName>
{

}
```

interface

```
<AccessModifier>interface<InterfaceName>
{

}
```

enum

```
<AccessModifier>enum<EnumName>
{

}
```

Class Loader
- 1) Loading
- 2) Linking
- 3) Initilization

Memory area's present in JVM

- Class Area ===> Data about the class ie., metadata will be present inside class Area
- Heap Area ===> All the java Objects will be present in this heap area. [SCP==> **S**tring **C**onstant **P**ool]
- Stack Area ===> It is used to store temporary results of methods.
- PC-Registers ===> It holds the address of JVM instructions which thread is executing at a prarticular point of time
- Native method Stack ===> It is a place which holds native methods.

Execution Engine

- Intrepretor ===> Executes byte code sequentially
- Garbage Collector===> It will automatically destroys unused or unreferenced Objects from HEAP MEMORY

OOP's Features

1) Encapsulation
2) Inheritance ⇨ "Acquaring the properties of one class into another class"
3) Polymorphism
4) Abstraction

```
class ClassA
{
    public void display()
    {
       Syso("Hi");
    }
}
```

```
ClassA===> Parent class

ClassB===> Child Class
```

```
             class ClassB extends ClassA
             {
               public void show()
               {
                 syso("hello");
               }
               public static void main(String[] args)
               {
                  ClassA aobj=new ClassA(); ⇨ ClassA obj creation
     Valid   ⇨   aobj.display();  ⇨ calling classA method
   Invalid   ⇨   aobj.show(); ⇨ Calling classB method (ClassA obj)
                  ClassB bobj=new ClassB(); ⇨ ClassB obj creation
     valid   ⇨   bobj.display(); ⇨Calling ClassA method(ClassB obj)
     Valid   ⇨   bobj.show(); ⇨ Calling ClassB method
               }
             }
```

Scanned by TapScanner

Naming conventions for Java Identifiers:

===> A java identifier can start with

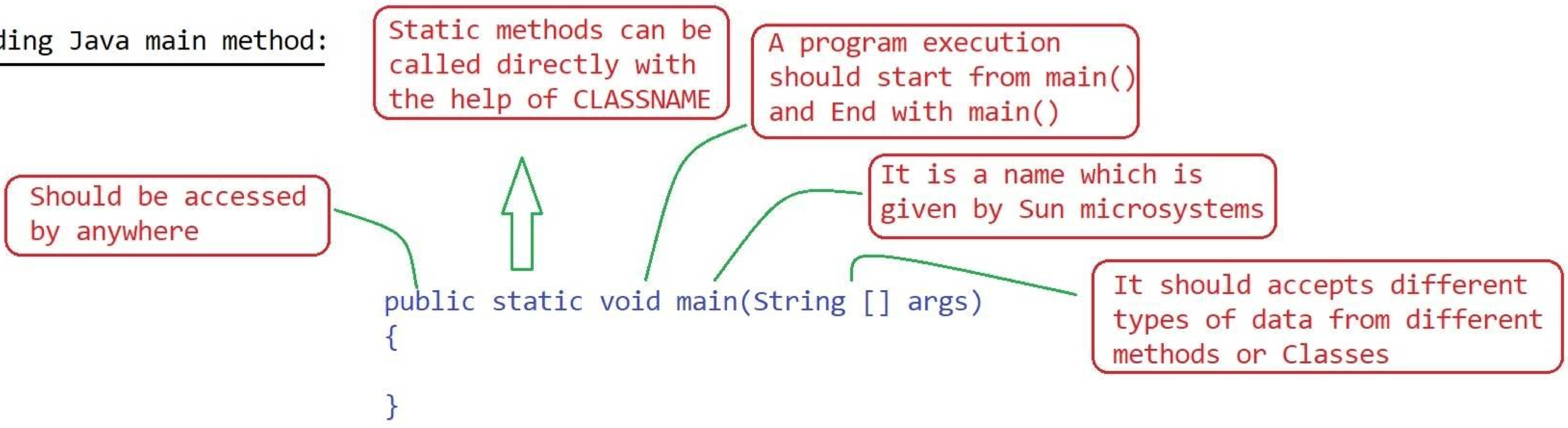
1) A to Z
2) a to z
3) $ and _

Java Separators:

;	It used to end a statement
,	It is used to differentiate variables, paramaters etc
{}	Those represent a BLOCK
()	Method represetation, we can write paramaters & conditions
[]	It is used to represent an ARRAY
.	It is used to call methods, to differentiate packages.

```
===> We can use numbers also in the identidiers (0 to 9), but a java identifier never starts with a number.
===> We can use only 2 symbols ( $ and _ ).
===> Java is case sensitive
        int a=10;
        int A=20;
===> There should not be any spaces between the identifier names. (Ex: String student Name="Sujatha";// INVALID)
===> We can take our own length for an identifer.
        int a=10; // VALID
        int aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa=10;// VALID
===> There are 50 Java language keywords are there, all those keywords we cant use them as identifier names.
===> We can use java ClassNames as identifier names but it is highly not recommended.
```

Understanding Java main method:



```
public static void main(String [] args)
{

}
```

Should be accessed by anywhere
Static methods can be called directly with the help of CLASSNAME
A program execution should start from main() and End with main()
It is a name which is given by Sun microsystems
It should accepts different types of data from different methods or Classes

Understanding java Constructor:

===> Constructor is one type of ***special*** method. ☆☆☆
===> Constructor is used to initilize an Object.

Rules:

1) Constructor should be having same name as Class Name
2) Constructor should not have any return type

Types:

There are 2 types of constructors are present

===> Paramaterized

===> Non-Paramaterized

How to call a Constructor?

Ans) COnstructor will be called simultaneously when ever we are creating an object.

Keywords: ==> In java there are 50 keywords are present
==> All the java language keywords starts with small letters
==> There are 2 reserverd keywords are present(goto & const)
==> true, false and null these are not KEYWORDS, these are literals(means Values)

Variables: In java there are '3' types of variables are present

Instance Variables :The variables which are declared inside a class outside any method or block or constructor, are known as Instance Variables.JVM will automatically initilize them, with their default values.

Static Variables :The variables which are declared inside a class outside any method or block or constructor with the help of 'static' keyword, are known as static Variables.JVM will automatically initilize them, with their default values. static variables can never be local variables.

Local Variables :The variables which are declared inside a method or block or constructor, are known as local Variables.JVM will NOT provide them with any default values. Programmer is supossed to intilize them explicitly.

Constructor:

1) Constructor is one type of special method
2) It is used to initilize the Object.

Rules:

1) Constructor should be having same name as ClassName
2) Constructor should not be having any return type.

Types Of Constructors:

1) Non-Paramaterized / Default Constructor
2) Paramaterized Constructor

Q) How to call a Constructor?

Ans) Constructors will be called simultaneously when ever we are creating an Object.

Syntax Of Method

```
<AccessModifier><ReturnType><MethodName>()
{

}
```

Syntax Of Constructor

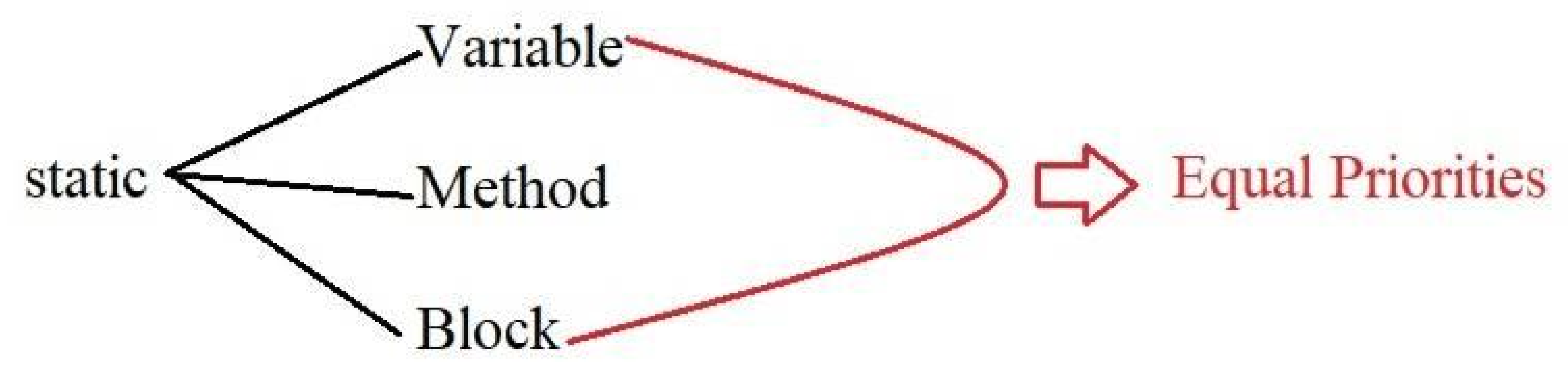
```
<AccessModifier><ClassName>()
{

}
```

Keypoints

===>In a Java program we are supossed to initilize the Class Object with the available constructors present in the class.
===>Java Compiler will automatically will provide a DEFAULT constructor for your java program, if you are not writing any constructors in your class.
===>If you are wrting any constructors(paramaterized or non-paramaterized) in your class then compiler will not provide any default constructor
===> For the constructors which are provided by the programmers we can use all the four ACCESS MODIFERS.
===> For the constructor which is given by the compiler there will be only 2 access modifiers, ie., what ever the class is having same modifer will be appended to the default constructor.

Understanding 'static' keyword:

===> It a java language keyword, which is used in 3 ways .(static keyword in java is mainly used for memory management)



Static Variable:

1) A variable which is declared as static and present inside a class out side any method ,block or constructor is known as static variable.
2) There will be **only one copy** of static variable available for the entire program. (So it is mainly used for making common properties of class as static)
3) Memory for the static variables will be assigned at the time of class loding.
4) If we change the value of a static variable then the changes will be applied to the whole program.
5) For static variables JVM will automatically assign them with their default values. But **for final static variables JVM will not assign with default values**. Programmer has to explicity assign the values.
6) static variables can never be local variables.

Static Method:

1) A method which is declared as static is known as static method.
2) static methods can be called directly with the help of ClassName
3) We can call static methods with the help of Class Object also.
4) static methods can not access non static data members (Variables) **directly**. We can access the instance variables inside a static method with the help of ClassObject
5) We can call the static method directly with the help of ClassName

Static Block:

1) If a java program is having main() and static block priority will be given to static block.
2) We can write any number of static blocks all will be executed in the defined order.
3) But if we want to run a java program 100% main() is required(after java 1.5v)

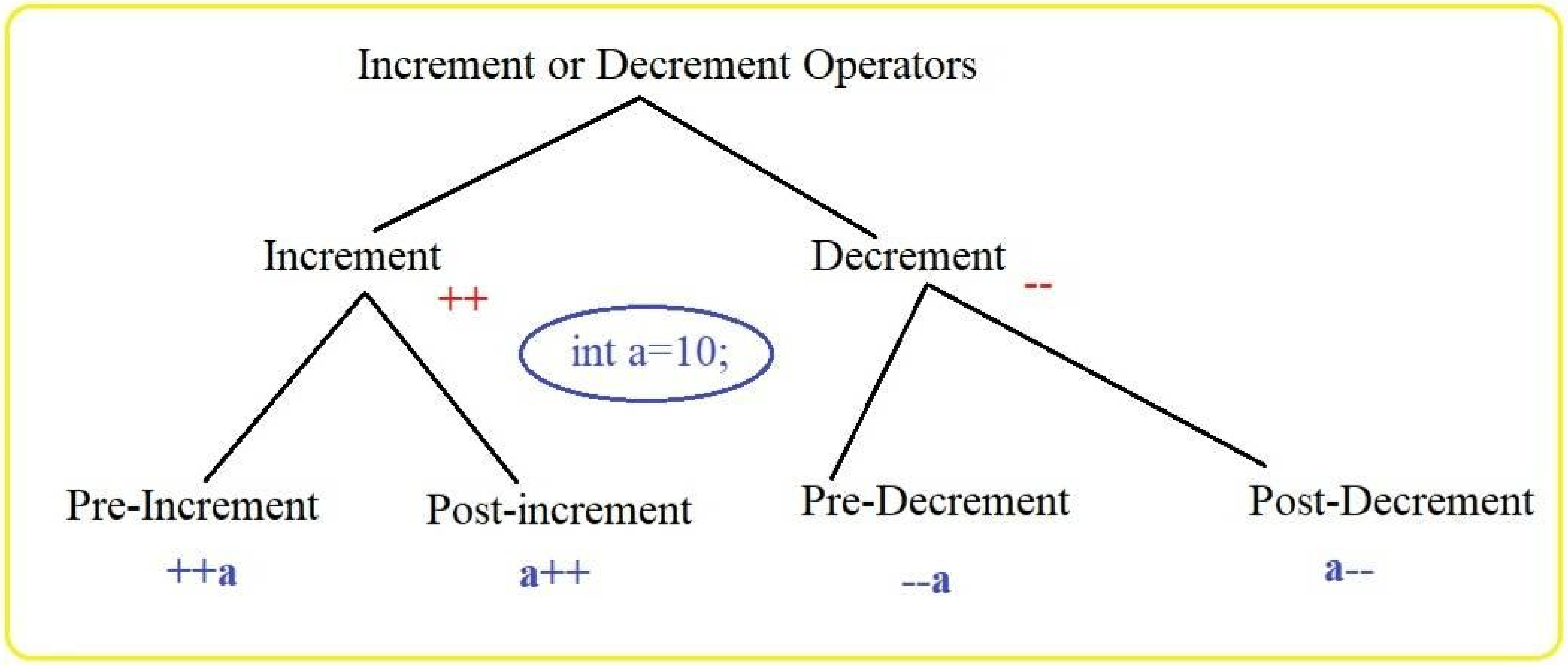
Operators:

1) Increment or Decrement Operators

2) Arithematic Operators [+ , - , * , / , %] **5**

3) Relational Operators **(6)**

4) Logical Operators **(3)**



```
int a=10;
System.out.println(a++); // 10   a=11
System.out.println(++a); // 12   a=12
System.out.println(a--);// 12   a=11
System.out.println(--a); //10   a=10
```

Type Casting: Converting one datatype into another datatype. [All the datatypes except boolean we can conver one datatype into another]

Type Casting
- Implicit Type casting (Smaller Datatype ===> Larger Datatype)
  - Note: 1) Implicit type casting will be done by the compiler automatically
  - 2) In this casting there will not be any loss of information
- Explicit Type casting (Larger Datatype ===> Smaller Datatype)
  - Note: 1) Explicit type casting should be done by the programmer
  - 2) In this casting there may be a chance of loss of information

[minimumRange + (result - maximumRange - 1)]

Ex: Implicit Type Casting

```
byte b=10;
int i=b;
System.out.println ("byte Value===>"+b); // 10
System.out.println ("int Value===>"+i); // 10
```

Ex: Explicit Type Casting

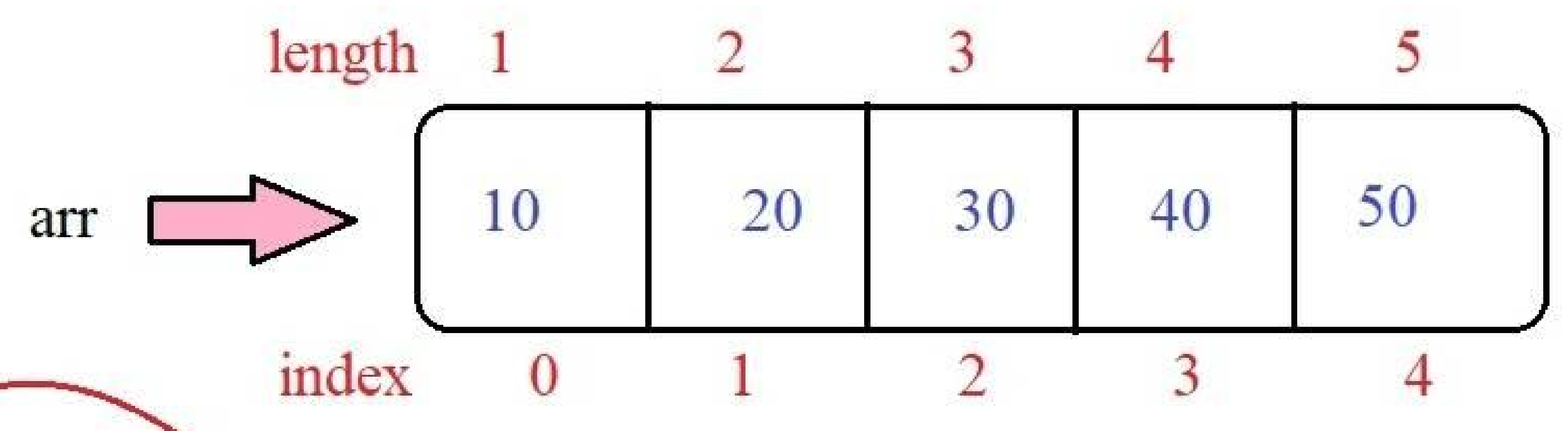
```
int i=100;
byte b=(byte) i;
System.out.println("int value===>"+i); //100
System.out.println("byte value===>"+b);//100
```

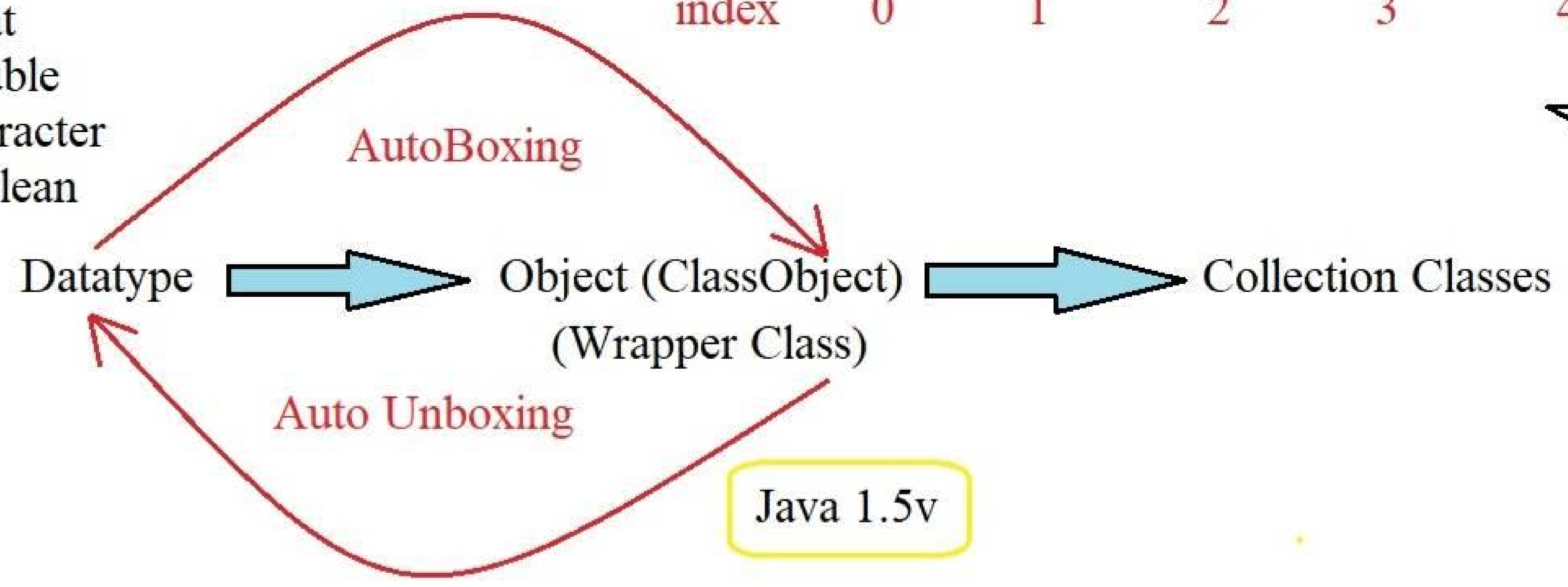
Wrapper Classes:

Array: It collects multiple elements of similiar datatypes in a continous block of memory

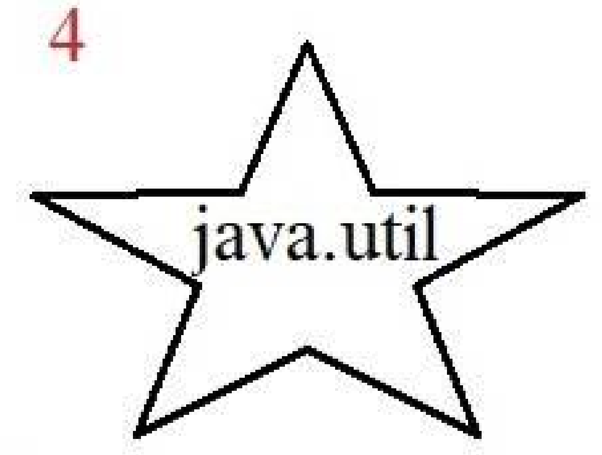
int arr[]={10,20,30,40,50};

int	⇨	Integer
byte	⇨	Byte
short	⇨	Short
long	⇨	Long
float	⇨	Float
double	⇨	Double
char	⇨	Character
boolean	⇨	Boolean





Req: I want to collect multiple elements of different datatypes.

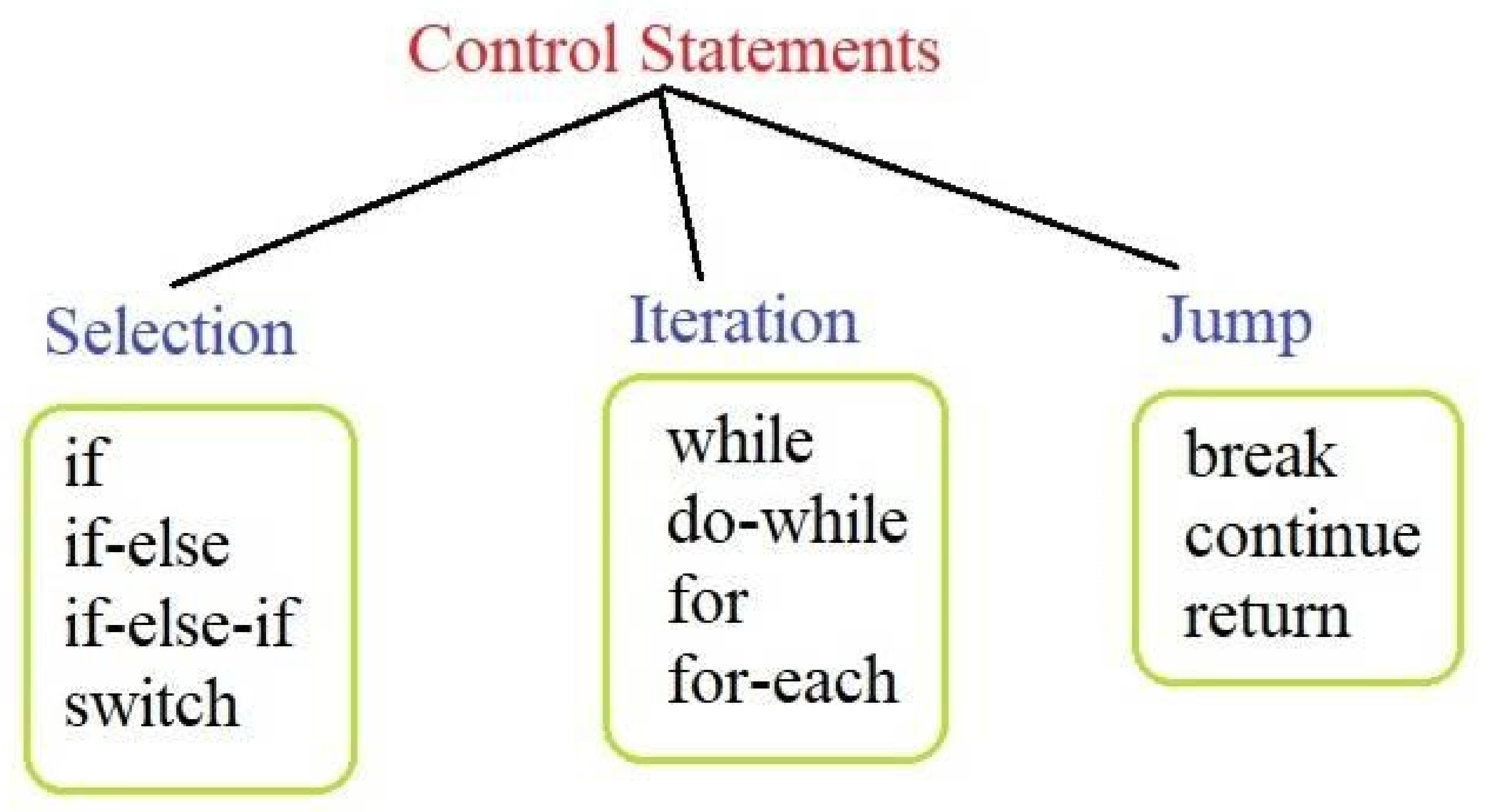


Collection Framework

- ==> ArrayList
- ==> LinkedList
- ==> Vector
- ==> HashMap
- ==> HashSet
- ==> PriorityQueue
- ==>......etc

Collection classes will accepts multiple elements of different Objects

Control Statements: Control flow statements change or break the flow of execution by implementing, decesion making, looping and branching of your program based on certion conditions known only during runtime.



Selection Statements: Selection statements allows the flow of program exection based upon conditions known only during runtime

if Statement:

syntax:
```
if(Condition) // true or false
{

}
else
{

}
```

Note:
1) We can write only if condition
2) In if statement {} are not mandatory, if you are not writing the braces then only one statement is dependent on if. And in that single statement we should not declare any values.

Understanding switch case statement:

===> switch will have many possible executions

===> Which case lable got matched with the value of the key then that case will be t
triggered

===> switch will accept byte, short, int and char up tp java 1.4V from java 1.5v
onwards switch started accepting their respective wrapper classes also.

===> From java 1.7 onwards switch started accepting Strings also.

===> Case lables and key value should be compatable datatypes.

===> In switch duplicate cases are not allowed.

===> Case label range should be with in the range of the key.

===> We can use expressions in switch key and lables

===> We are supossed to write only one default case and it can be anywhere in
the switch

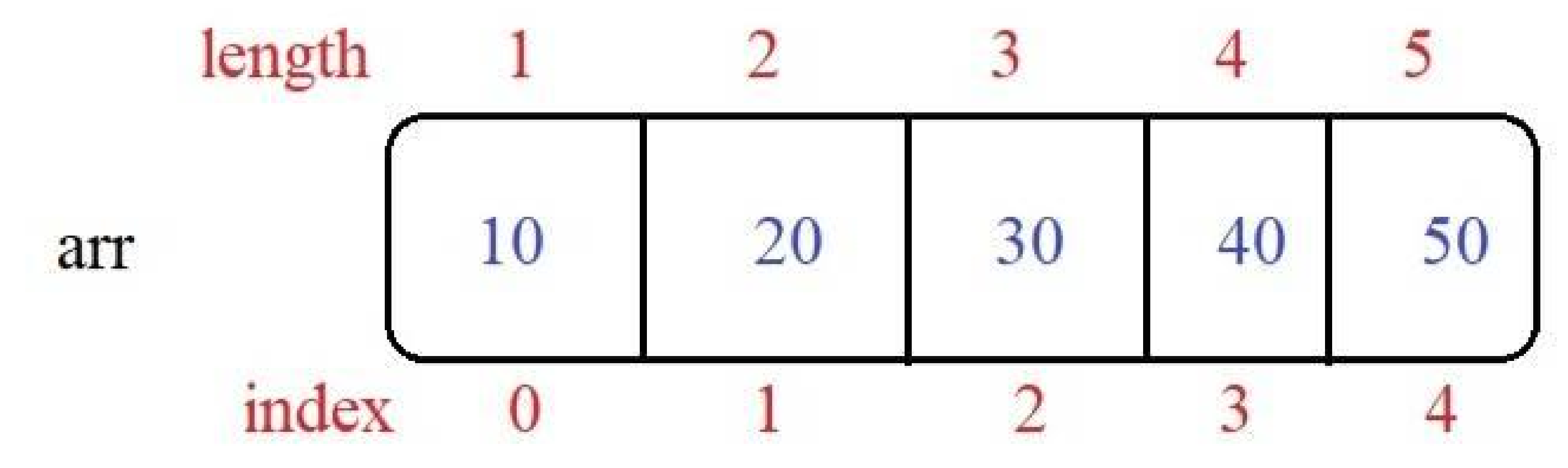
===> In swicth case there should not be any individual statements.

===> All the cases, including default and break statements are optional in switch.

===> Every case label should be compile time constants.

```
switch(key)
{
  case label:
     ---------;
     ---------;
     break;
  case label:
     ---------;
     ---------;
     break;
  case label:
     ---------;
     ---------;
     break;
}
```

```
int arr[]={10,20,30,40,50}
```



```
syso(arr[2]); // 30
syso(arr[4]); // 50
```

```
for(int i=0;i<5;i++)// 5<5
{
   syso(arr[i]); //10  20  30  40  50
}
```

```
for(int i=4;i>=0;i--)
{
  syso(arr[4]);
}
```

```
for(int x:arr)
{
    syso(x);
}
```

Scanner Class:

===> It is a predefined class which is present in $\boxed{\text{java.util}}$ package.
===> Scanner class is used to take the input from the user

```
Scanner sc=new Scanner(System.in);
```

`int result=sc.nextInt();` $\Longrightarrow$ It is going to return what ever the value which is entered by the user

Understanding packages in java:

===> A java package represents similiar types of classes, interfaces and sub packages.

===> Packages
- Predefined Packages [5000]
- Userdefined Packages

===> In every java program by default java.lang package will be imported

Access Modifiers:

There are 4 types of access modifiers are present in Java:

public

private

protected

default

Q) In how many ways we can make a class which is present in different package available for your program?

Ans) There are '3' ways for making the class from different package available for our program

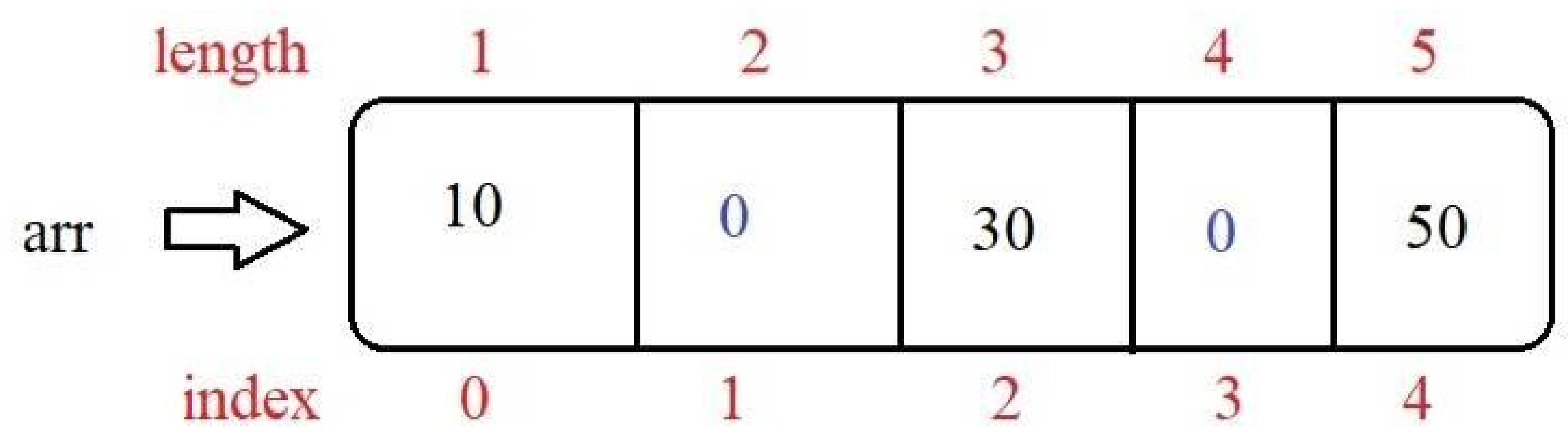
⇨ By using import PackageName.*; [All the classes which are present in that package will be imported in your program]

⇨ By using import PackageName.ClassName; [Only mentioned class will be imported]

⇨ By using fully qualified ClassName; [Only mentioned class will be imported and **there is no need to use import statement**]

Array: It collects multiple elements of similiar dataype in a continouis block of memory.

```
int a=10;          int arr[]=new int[5];
int b=20;           arr[0]=10;
int c=30;           arr[2]=30;
int d=40;           arr[4]=50;
int e=50;           arr[5]=60;// AIOB
```



```
int arr1[]=new int[5];

int arr2[];
arr2=new int[5];

int arr3[]={10,20,30,40,50};

int arr4[]=new int[]{10,20,30,40,50};
```

4

```
for(int i=0;i<arr.length;i++)
{
    System.out.println(arr[i]);
}
```

```
for(int i=arr.length-1;i>=0;i--)
{
    System.out.println(arr[i]);
}
```

```
for(int x:arr)
{
    System.out.println(x);
}
```

Disadvantages

==> Arrays accepts only homogeneous data.
==> There is no method support in array
==> The length of an array is fixed.

COLLECTION FRAMEWORK

length	Vs	length()
It is used in array		It is used in String
It is a final variable		It is a final method
It is used to get the length of an array		It is used to get the length of String

```
class ClassA
{
public static void main(String args[])
{

  int arr[][]={{1,2,3},{4,5,6},{7,8,9}};

  for(int i=0;i<3;i++) // i=0  0<3
   {
    for(int j=0;j<3;j++)
     {
      System.out.print(arr[i][j]+" ");
     }
    System.out.println();
   }
}
}
```

$$\begin{array}{cc} & \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{array}{c} 0 \\ \\ 1 \\ \\ 2 \end{array} & \begin{bmatrix} \substack{1 \\ (0,0)} & \substack{2 \\ (0,1)} & \substack{3 \\ (0,2)} \\ \substack{4 \\ (1,0)} & \substack{5 \\ (1,1)} & \substack{6 \\ (1,2)} \\ \substack{7 \\ (2,0)} & \substack{8 \\ (2,1)} & \substack{9 \\ (2,2)} \end{bmatrix} \end{array}$$

Understanding Java Strings:

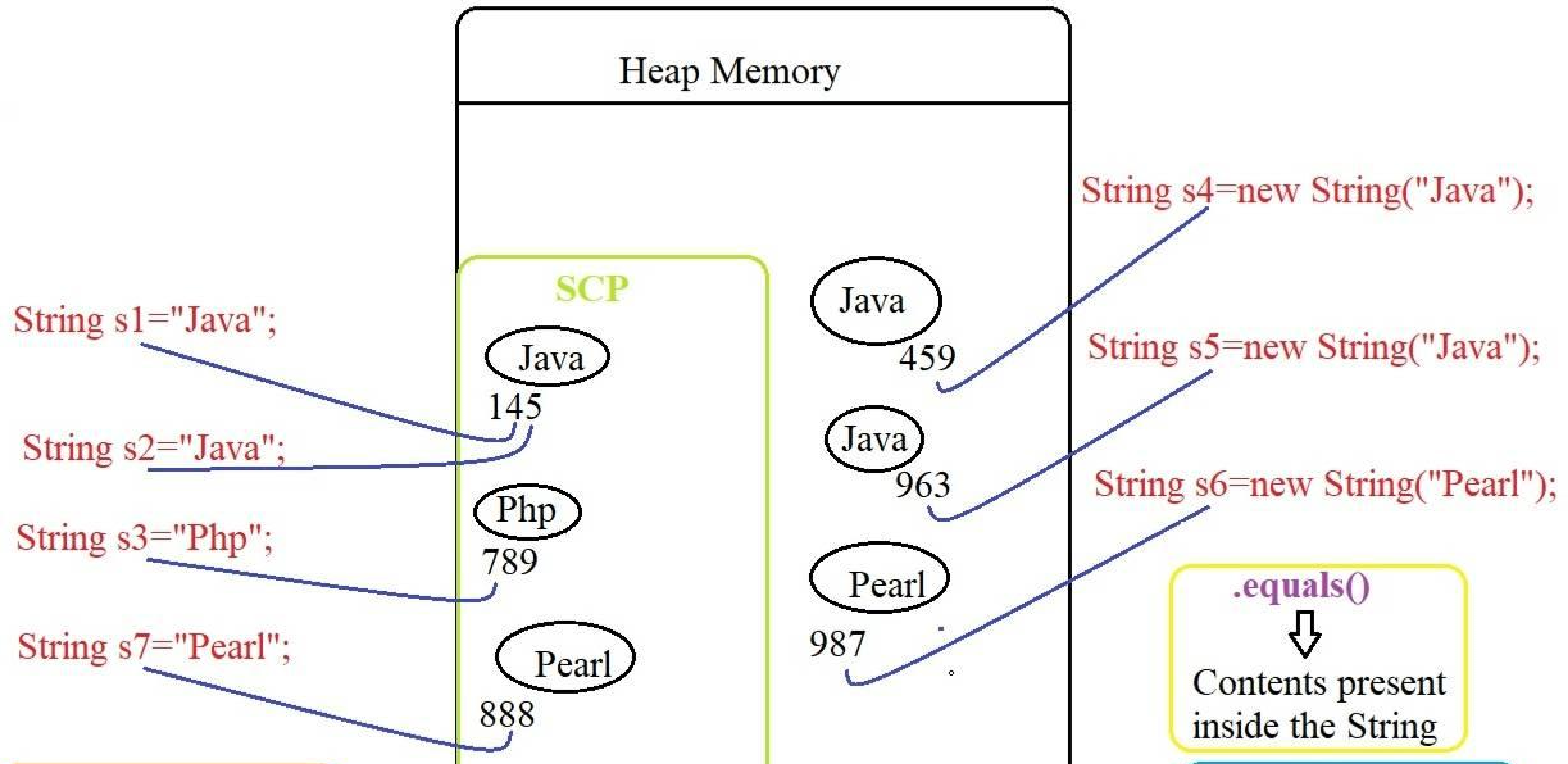
===> Strings are immutable.
===> String is a collection of group of charcaters.
===> Strings acts as both Class & a Datatype

```
String s1="Java"; // Java

String s2=new String("Java"); // Java

char c[]={'J','a','v','a'}
String s3=new String(c);  // Java

String s4=new String(c,2,2); // ?
```

Q) String s1="Java"; $\Rightarrow$ Case 1

(Vs)

String s2=new String("Java"); $\Rightarrow$ Case2



In SCP there is no chance of Duplicate Objects.

.equals()
⇩
Contents present inside the String

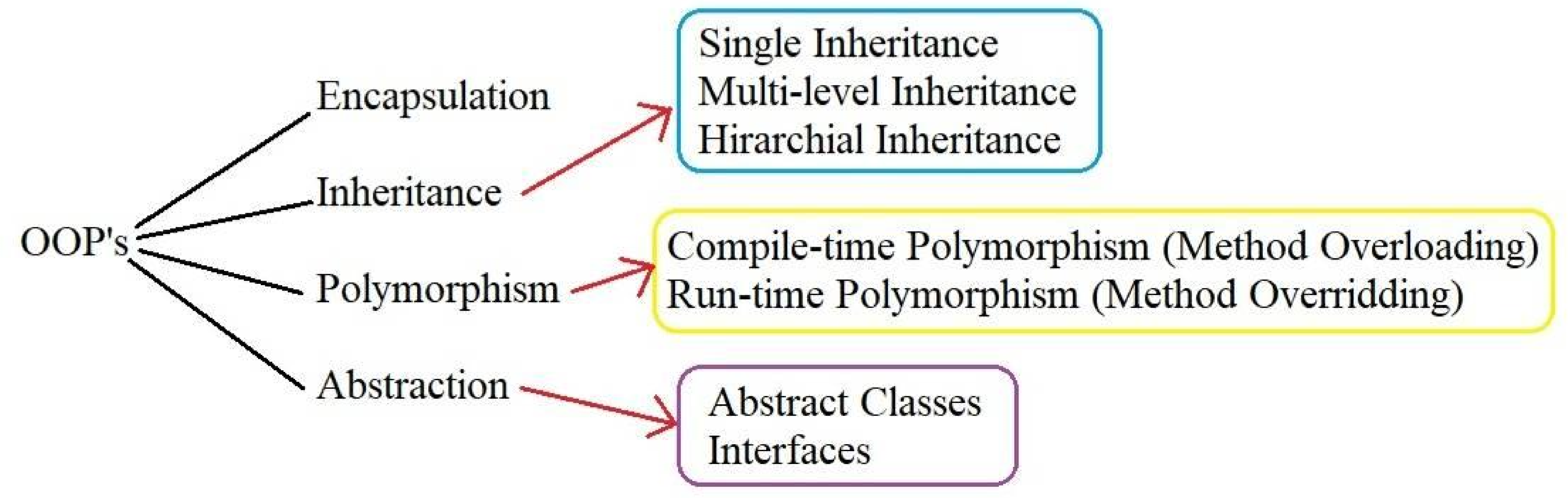
==
⇩
Checks the address locations

Scanned by TapScanner

Strings
- String [Strings which are created by String class are IMMUTABLE, .equals() will check the contents present in the Strings]
- StringBuffer [Strings which are created by StringBuffer class are MUTABLE, .equals() will check the address locations of the Strings]
- StringBuilder [Strings which are created by StringBuilder class are MUTABLE, .equals() will check the address locations of the Strings]

StringBuffer ===> Synchronized (introduced in Java 1.0v)

StringBuilder ===> Not-Synchronized (introduced in Java 1.5v)



empId
empName
empDep
empSal

Encapsulation

1 Defination : Encapsulation means wrapping up of data (or) Binding up of data in to a single unit.

2 Defination : Encapsulation means it is a process of making fields as private and providing access to those fields, with the help of public methods, ie., through setters and getters.

Inheritance: Acquaring the properties of one class into another class is known as Inheritance.
If we want to inherit one class into another class we need to use the keyword extends.

```
public class ClassA
{
   void meth1()
   {
      syso("ClassA method called");
   }
}
```

ClassA	-	ClassB
Parent Class		Child Class
Super Class		Sub Class
Base Class		Derived Class

```
public class ClassB extends ClassA
{
  void meth2()
  {
     syso("Class B method called");
  }
  public static void main(String args[])
  {
     ClassB bobj=new ClassB();
     bobj.meth2(); //VALID
     bobj.meth1(); //VALID       => IS-A-RELATION
     ClassA aobj=new ClassA();
     aobj.meth1()
                                 => HAS-A-RELATION
   }
}
```

Keypoints:

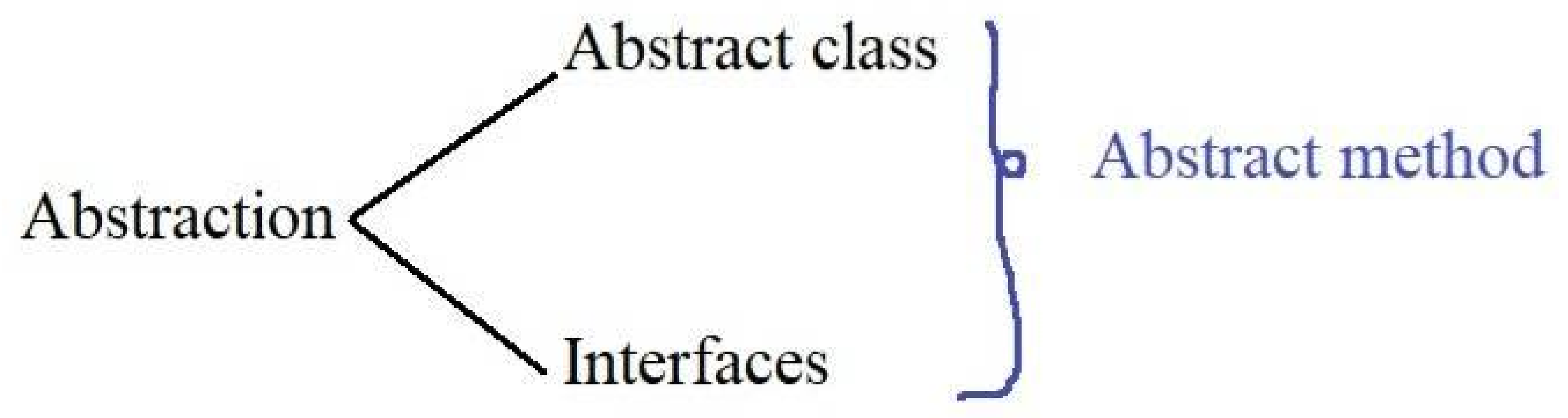
1) We can hold child class object with parent class reference. And with that reference we can call only parent class methods.
2) We can't hold parent class object with child class reference.(We will get an compile time error).
3) We can hold Child class object with child class reference. And with that reference we can call BOTH child class methods an parent class methods.
4) We can hold parent class object with parent class reference, and with that reference we can call only parent class methods.

```
public class ClassB extends ClassA
{
  public static void main(String args[])
  {
     ClassA aobj1=new ClassB(); //Only parent
     ClassB bobj1=new ClassA(); // INVALID
     ClassB bobj2=new ClassB();//Parent & Child
     ClassA aobj2=new ClassA();//Only parent
  }
}
```

Abstraction: It is a process of hiding implementation details from the user and showing only the necessary details to the user is known as Abstraction

abstract method: <AccessModifier> abstract <ReturnType><MethodName>();

===> A method which is declared as abstract with 'abstract' keyword is known as abstract method.
===> abstract method always ends with semicolon(;)
===> abstract method doesnot have any body
===> Implementation of the abstract method will be given with the help of Method Overridding



abstract method can be Overloaded

abstract method can be Overridden

abstract class:

===> A class which is declared as abstract with the help of 'abstract' keyword is known as abstract class.
===> In side an abstract class we can write both abstract methods and normal methods.
===> It is not mandatory that we should 100% write an abstract method in the abstract class. (Writing abstract method in abstract class is optional).***
===> In a normal java class if we are writing an abstract method, then 100% we need to declare that class as abstract, or else we will be getting compile time error.
===> For an abstract class we cant create an Object.
===> We can write constructors, static methods,and even main() also in abstract class.
===> If we are inheriting an abstract class in a normal class, then if there are any abstract methods present in that abstract class, then 100% we need to provide implementation(by using Method Overridding) for those abstract methods in the child class. Otherwise we will be getting an Compile time error.
===> If we dont want to provide implementation for the abstract methods present in the abstract class, then make your child class also as abstract

Interface:

===>Interface is not a class, It is a blue print of a class.

Syntax:

```
<AccessModifier>interface<InterfaceName>
{

}
```

===> In interface every method is by default public abstract.
===> In interface every variable is by default public static final.
===> We cant write constructor in an Interface.
===> After 1.7v onwards we can write default methods, static methods and main()
===> From 1.9v onwards we can write private methods also inside interface.
===> We cant create an Object for Interface
===> In interfaces there are

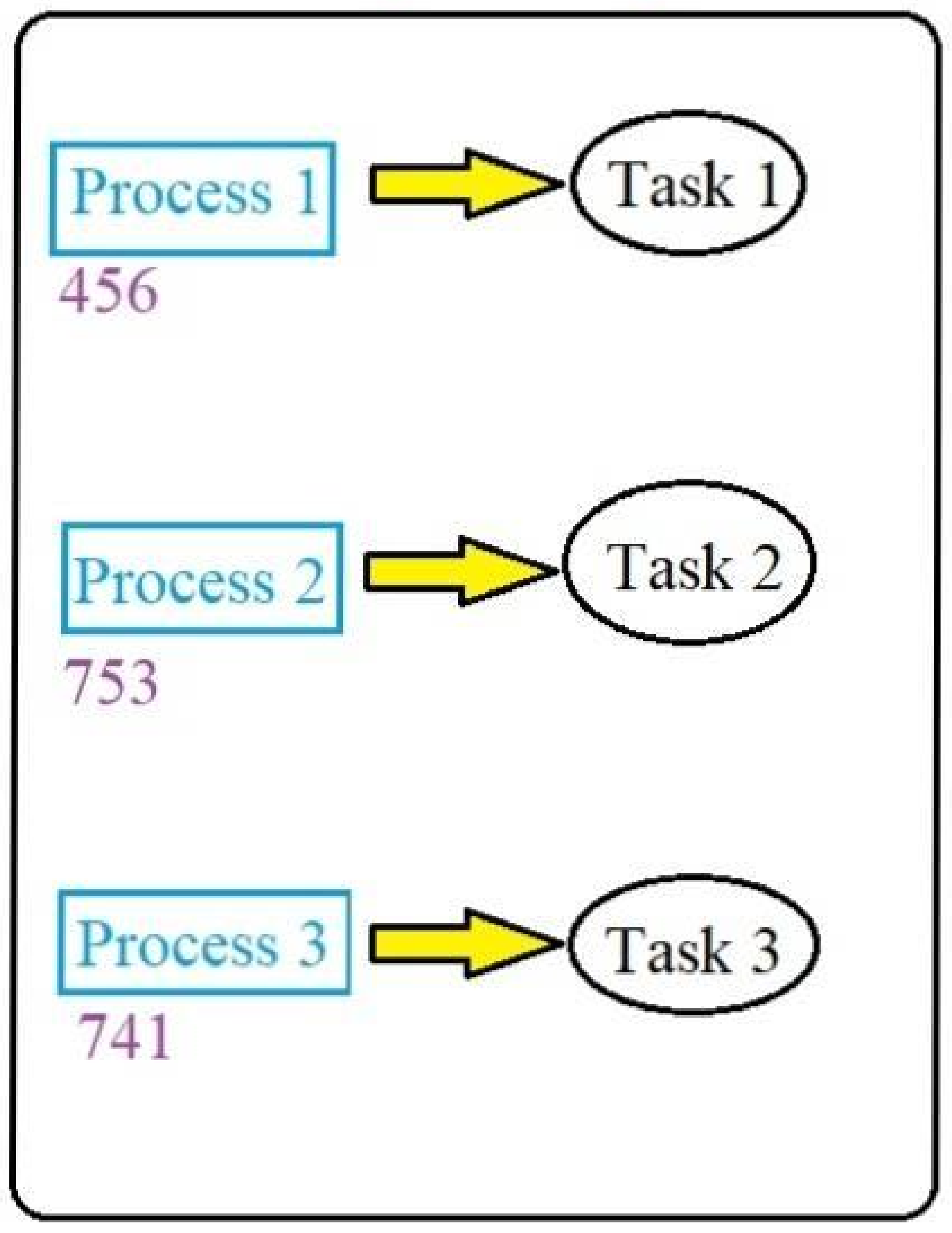
⇨ Marker Interface [It is an Empty Interface] Ex: Cloneable, Serilizable

⇨ Functional Interface [It will have only one abstract method] Ex: Runnable

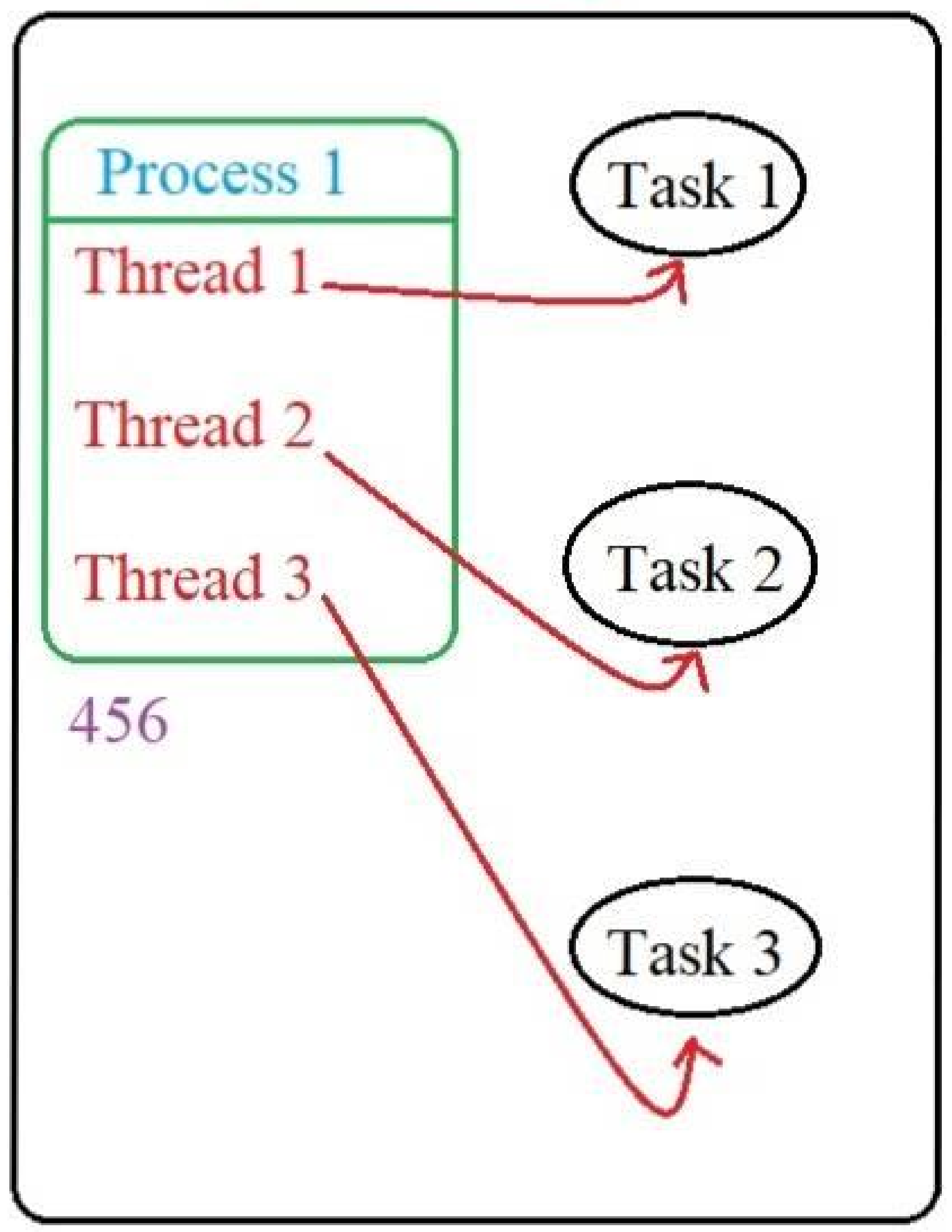
Multitasking: Performing multiple tasks simultaneously at the same time by using a single processor inorder to optimize the utilization of cpu.

Multitasking
- Process-Based Multitasking [Multiprocessing]
- Thread-Based Multitasking [Multithreading]

Multiprocessing



Multithreading



Thread:

1) Thread is a smallest unit of a process.
2) Thread is light weight process.
3) Process acts as a host for a thread.
4) Atlease one process is required for creating a Thread.
5) Threads share same address location
6) Context-Switching is easy in Threads.

```
public void run()
{

}
```

Thread Creation:

We can create a thread in '2' ways
===> By extending Thread class
===> By implementing Runnable Interface.

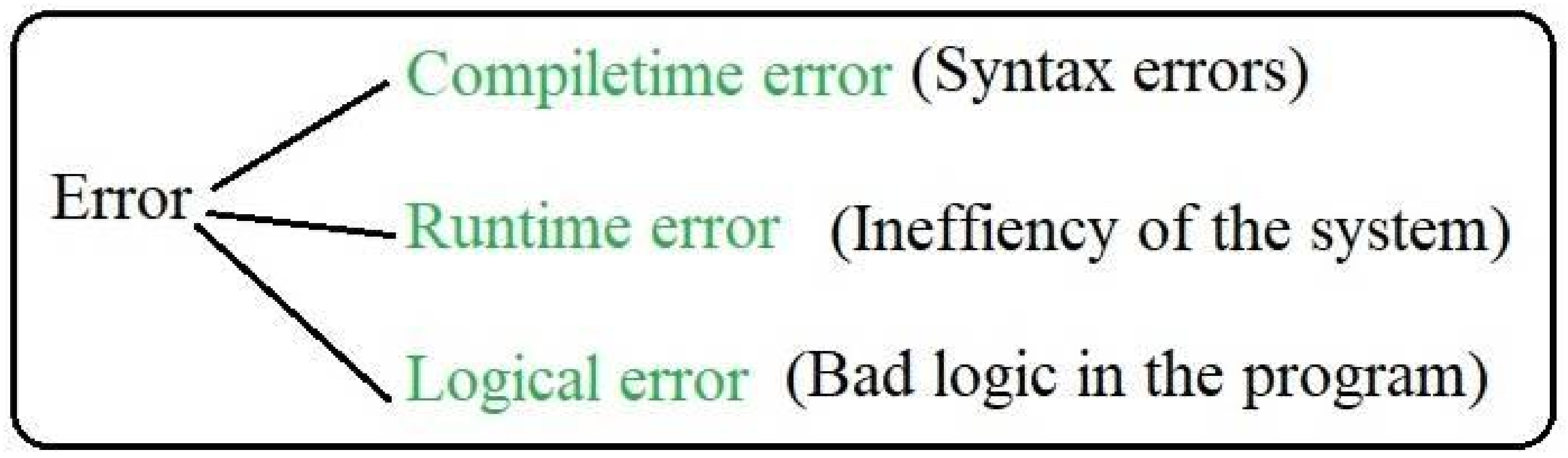
Thread Life Cycle Stages ➡ NEW===> RUNNABLE===> BLOCKED====> WAITING====> TIMED-WAITING=====> TERMINATED

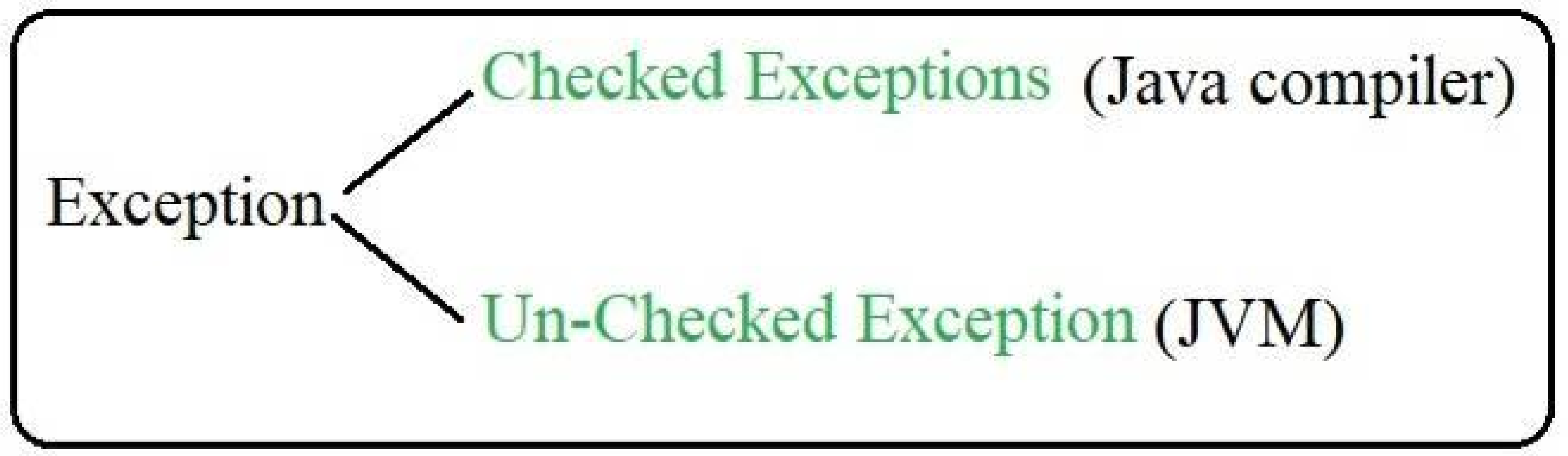
Exception Handling

Default Exception Handler

Error: If an error occured in our program the program will be terminated. We cant save our program

Exception: If an exception occured in our program the program will be terminated. But we can save our program


Error
Compiletime error (Syntax errors)
Runtime error (Ineffiency of the system)
Logical error (Bad logic in the program)


Exception
Checked Exceptions (Java compiler)
Un-Checked Exception (JVM)

How to handle an Exception:

```
try
{
    // we need to write suspecious code
}
catch()
{
   // we need to catch the exception
}
finally
{
   // It will always gets executed
}
```

throw :It is used to throw user defined exception msgs.

throws : It is used to escape from exception handling

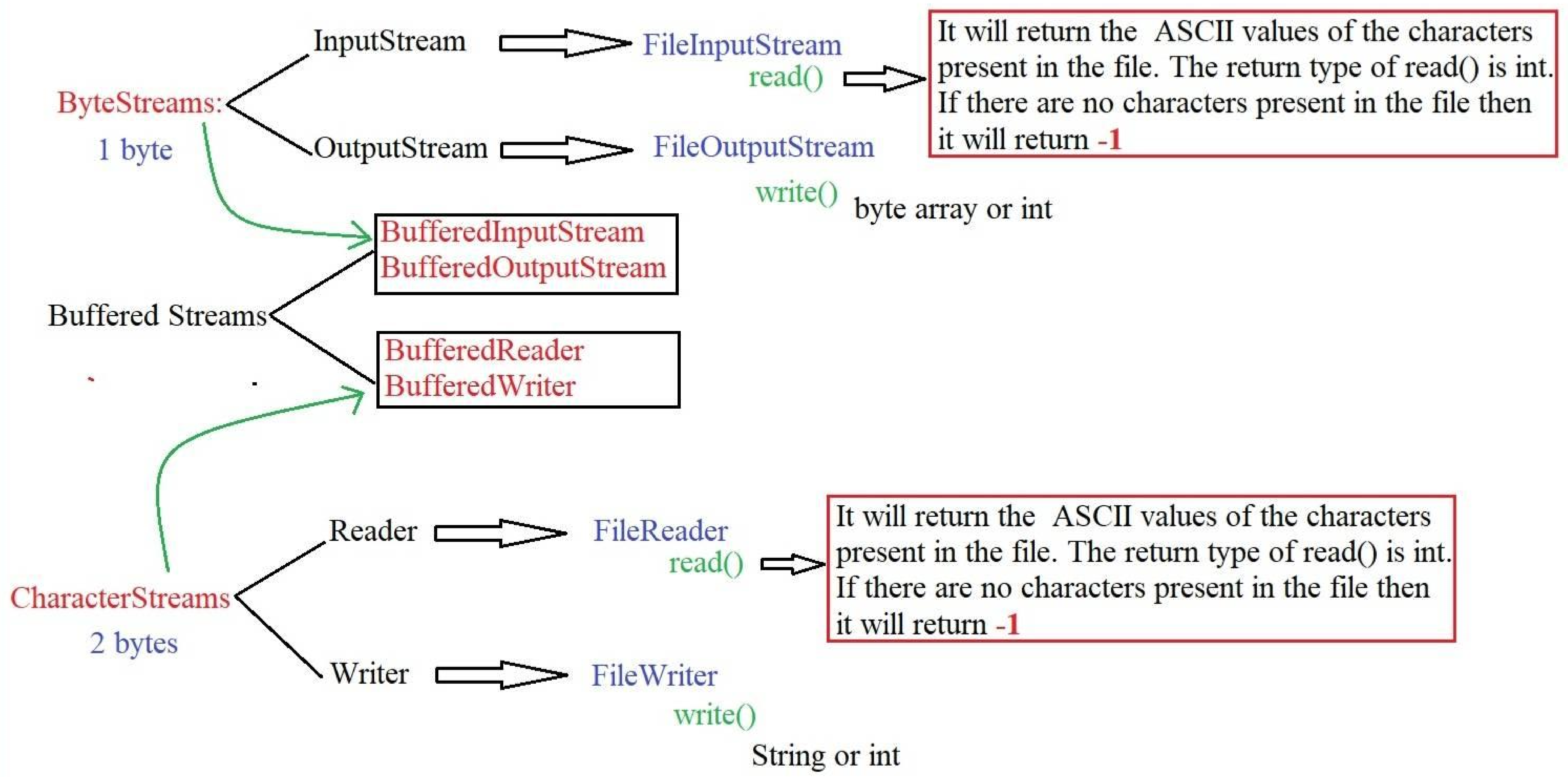
IO-Streams:

1) Byte Streams

2) Character Streams

3) Data Streams

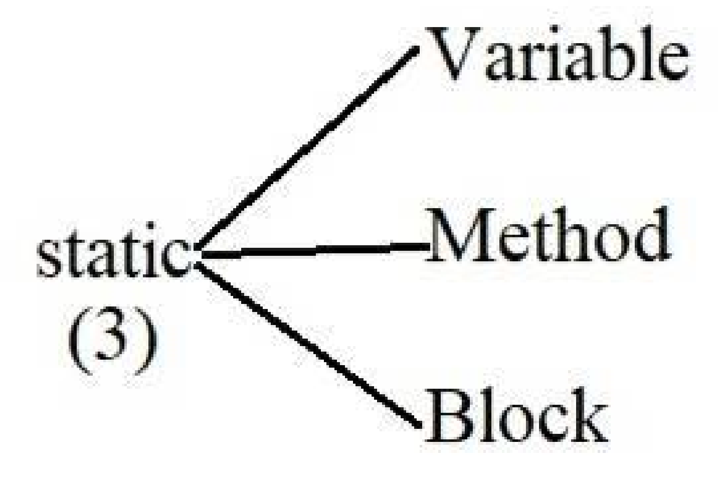
4) Object Streams

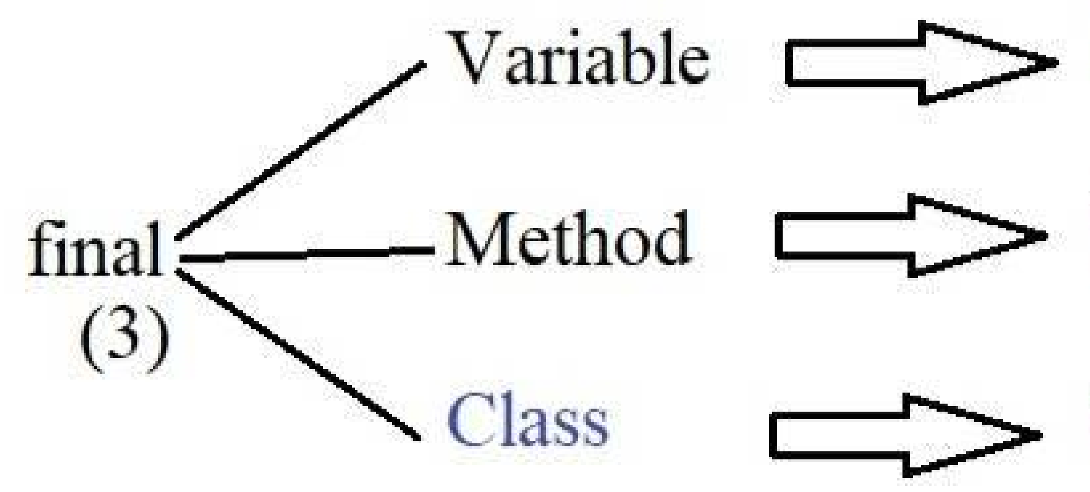


It will return the ASCII values of the characters present in the file. The return type of read() is int. If there are no characters present in the file then it will return **-1**

It will return the ASCII values of the characters present in the file. The return type of read() is int. If there are no characters present in the file then it will return **-1**

Understanding final keyword: It is used to restrict the user





We cant change the values of final variables (CONSTANTS)

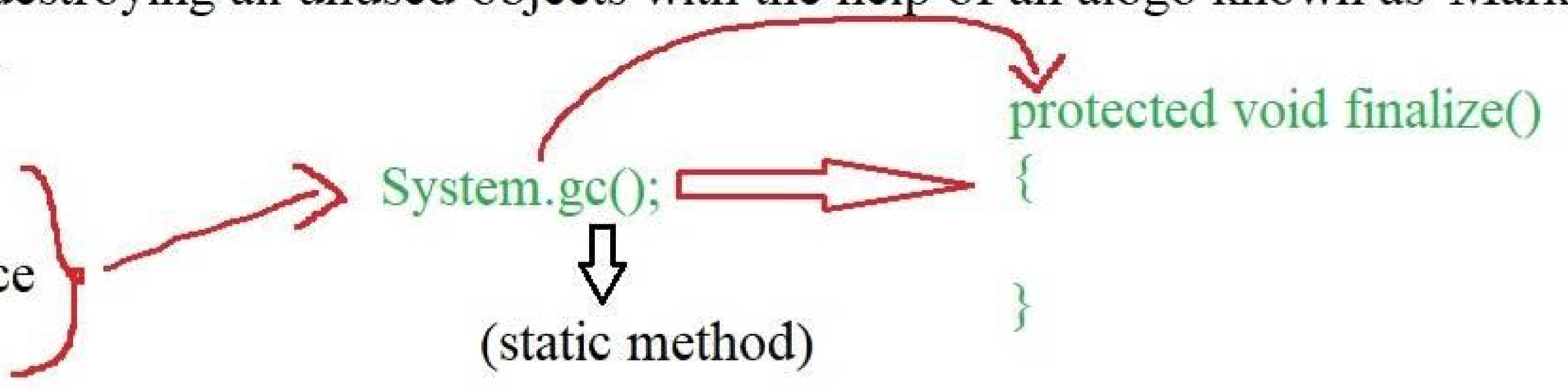
We cant provide overridding for final methods (But we can INHERIT)

We cant inherit the final class

Garbage Collection: It is a process of marking used objects and destroying all unused objects with the help of an alogo known as 'Mark & Sweep' from the heap memory. Garbage collector is known as DEAMON thread.

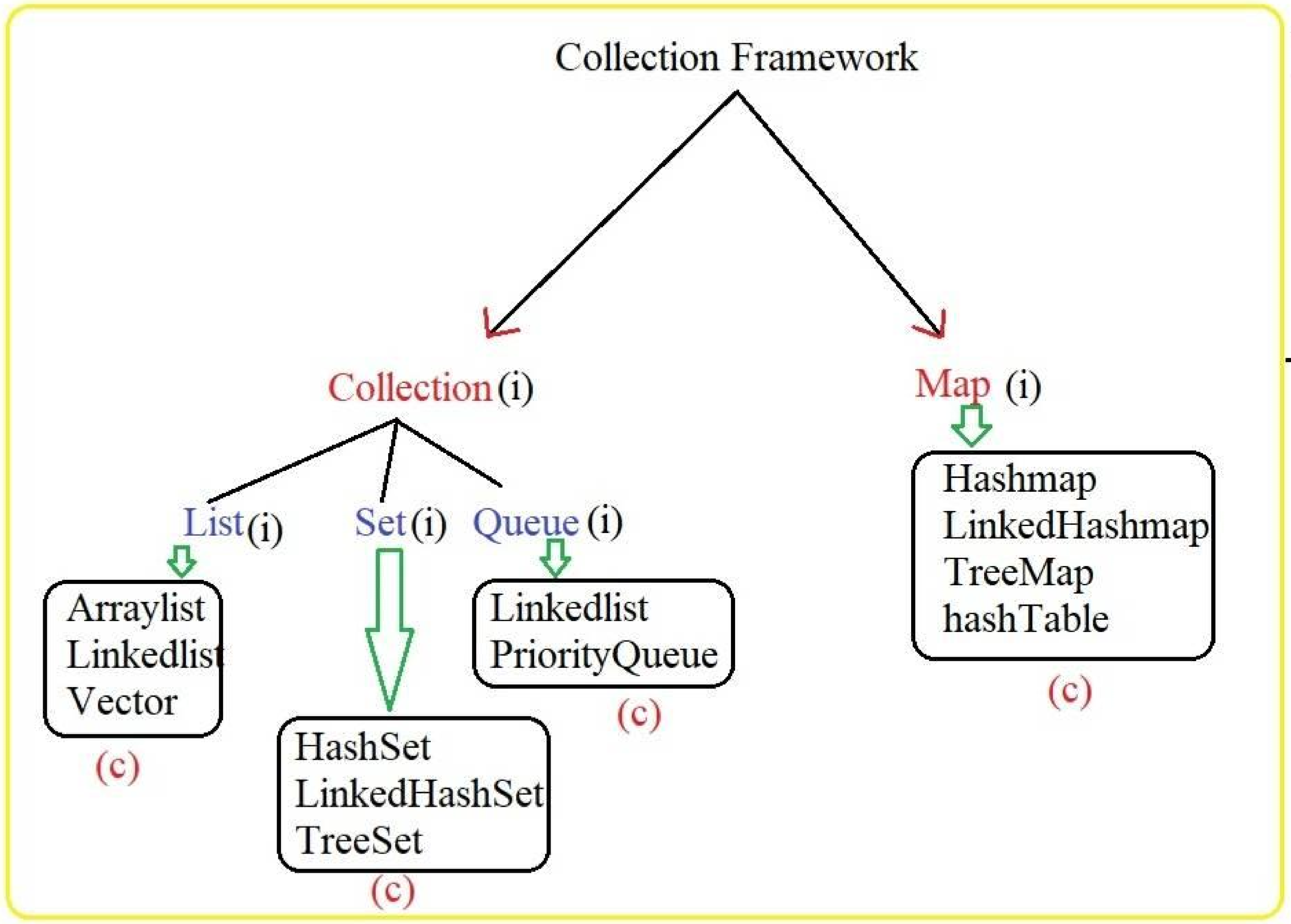
Q) When an object is eligible for garbage collection?

A) ===> If we are nullifying the reference
===> If we are assigning the reference to another reference
===> All the Objects which are created in side a method.



```
protected void finalize()
{

}
```

Collection Framework:



Iterator: iterates only in forward direction
ListIterator: iterates both in forward and backward directions
Enumeration: It is used in legacy classes

&

for-loop
for-each loop

List : In list elements are stored just like an array, It will allows duplicates.

Set: In set elements are stored just like an array, but it will not allow duplicates

Queue: Elements are stored in the form of FIFO.

Map: Elements are stored in the form of Key-Value pairs [EX: $\underset{K}{101}$-$\underset{V}{\text{"Java"}}$]