AKTU BTech (2nd Year)
& MCA (2nd Sem)

OOPs With JAVA

Unit-1 ONE SHOT

Introduction

By: Vikas Sir

Java Programming by Vikas Sir

Chapter No 1: (Introdution)

- Java is a high-level, object-oriented programming language that is designed to be platform-independent.
- It allows developers to write code that can run on any device equipped with the Java Virtual Machine (JVM).

OR

- Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language.

Java Programming by Vikas Sir

History Of Java Programming Language:-

- Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time.

- 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.

- 2)Firstly, it was called "Greentalk" by James Gosling, and the file extension was .gt. .java

- 3) After that, it was called Oak and was developed as a part of the Green project.

Java Programming by Vikas Sir

History Of Java Programming Language:-

- Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.
- 4) In 1995, Oak was renamed as "Java" because it was already a trademark by Oak Technologies.
- 5) JDK 1.0 was released on January 23, 1996. After the first release of Java, there have been many additional features added to the language.
- 1997: Java Development Kit (JDK) 1.1 was released, bringing several new features, including inner classes, JavaBeans, and JDBC (Java Database Connectivity).

Java Programming by Vikas Sir

History Of Java Programming Language:-

- 1998: The release of Java 2 (JDK 1.2) marked a significant update, with the introduction of Swing for GUI development, the Collections framework.
- 2004: Java 5 (originally called J2SE 5.0) introduced major enhancements like generics, metadata, enumerated types, and the enhanced for loop.
- 2006: Sun released Java under the GNU General Public License (GPL), making it open-source software.
- 2010: Oracle Corporation acquired Sun Microsystems, taking over the development and stewardship of Java.

Java Programming by Vikas Sir

History Of Java Programming Language:-

- 2018: Java 10 introduced local-variable type inference with the var keyword, and Java 11 provided long-term support (LTS), featuring new enhancements and deprecations.

- In 2024, Java's future lies in its integration with cutting-edge technologies like machine learning, cloud computing, and the Internet of Things (IoT).

- For instance, Java 8 Future introduces features for asynchronous computations, simplifying the development of multithreaded applications.

Java Programming by Vikas Sir

Characteristics Java Programming Language:-

Object-Oriented:

- Java is based on objects. An object can represent a real-world thing like a car or a concept like a bank account. Each object has attributes (properties) and behaviors (methods).

Platform-Independent:

- Write your code once, and it can run anywhere. Java programs are compiled into bytecode, which can be executed on any device with a Java Virtual Machine (JVM).

Simple and Easy to Learn:

- Java's syntax is straightforward and similar to other popular programming languages like C and C++. This makes it easier for new developers to pick up.

Java Programming by Vikas Sir

Characteristics Java Programming Language:-

Secure:

- Java has built-in security features that help protect against malicious code. For example, it runs inside a virtual machine, which adds an extra layer of protection.

Robust:

- Java emphasizes reliability. It has strong memory management, error handling, and type-checking mechanisms, reducing the chances of crashes and bugs.

Multithreaded:

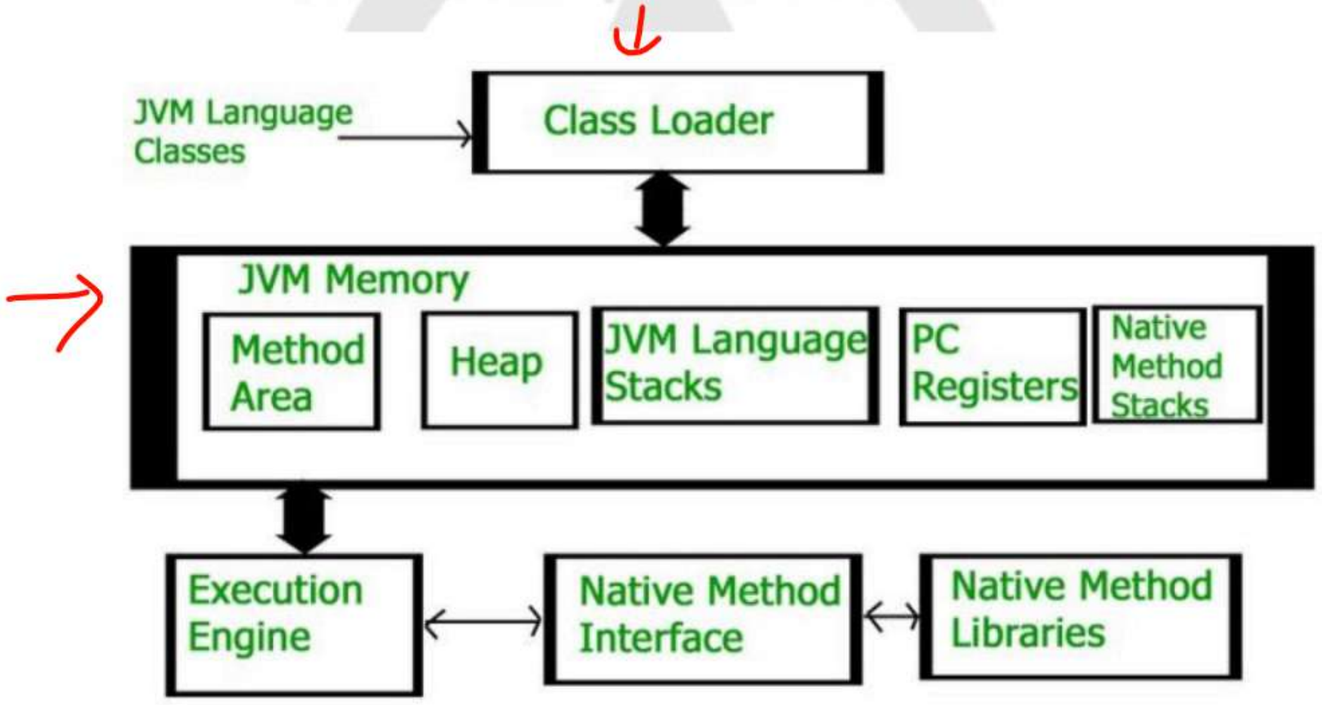
- Java can perform many tasks at the same time (multithreading). This is useful for creating applications that need to perform multiple operations simultaneously, like games or web servers.

Java Programming by Vikas Sir

JVM(Java virtual machine):-

- AJVM: JVM is an abstract machine. It is a software-based, virtual computer that provides runtime environment in which java bytecode can be executed.

Architecture of JVM



Java Programming by Vikas Sir

JVM contains the Component:

- 1. **Classloader:** Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader.
- 2. **Class area:** Class area stores per-class structures such as the runtime constant pool, field and method data.
- 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.

Java Programming by Vikas Sir

- 5.Program Counter (PC) register : PC register contains the address of the Java virtual machine instruction currently being executed.

- 6. Native method stack: It contains all the native methods used in the application.

- 7. Execution engine: It contains:

i. A virtual processor

ii. Interpreter: Read bytecode stream then execute the instructions.

Java Programming by Vikas Sir

JVM(Java virtual machine):-

iii. **Just-In-Time(JIT) compiler:** It is used to improve the performance. JIT compiles parts of the bytecode that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.

- 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.

- 9.**Role of JVM**: The primary role of the Java Virtual Machine (JVM) in the execution of programs is to act as an intermediary between the compiled Java bytecode and the host operating system and hardware.

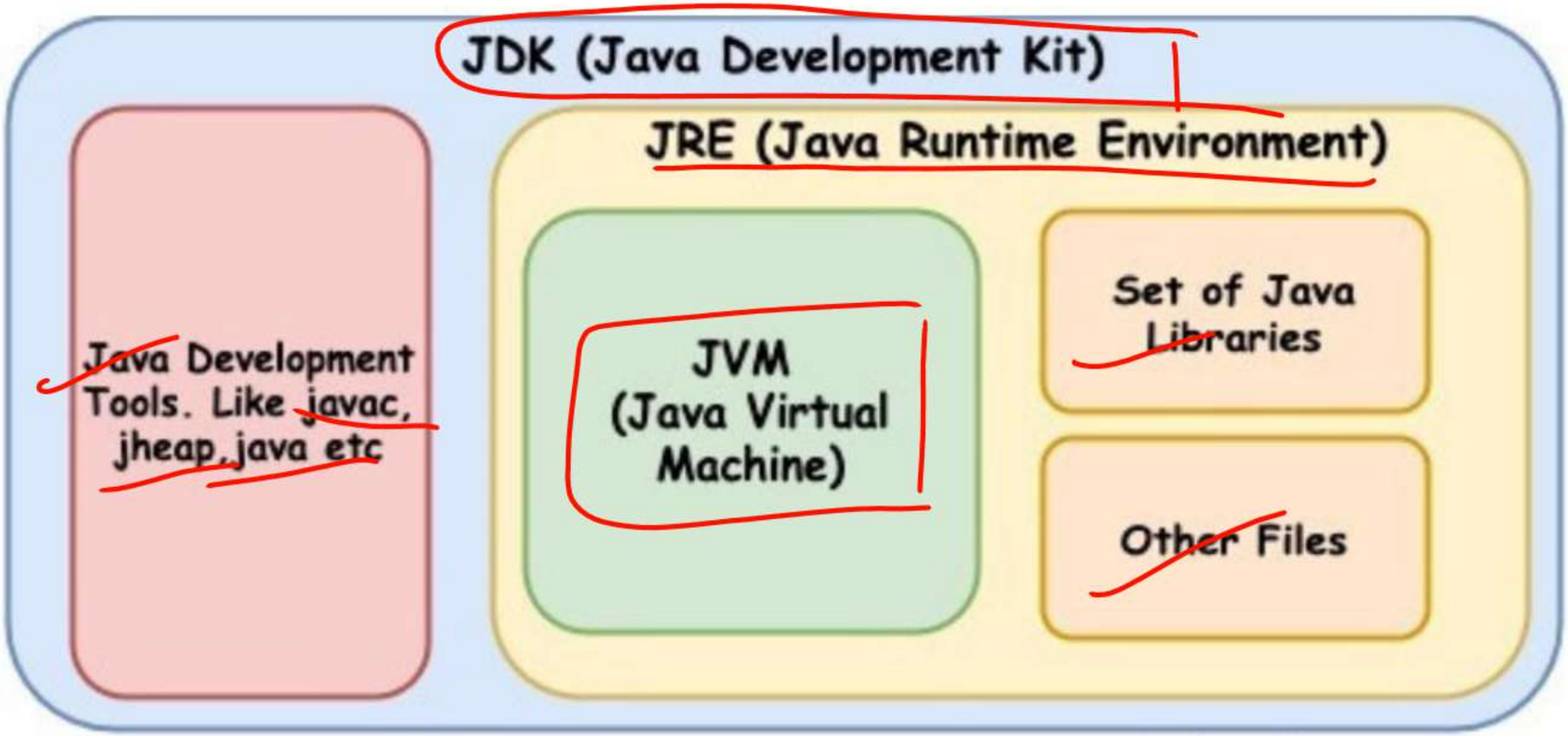
Java Programming by Vikas Sir

JDK(Java Development Kit)

- **Java Development Kit (JDK):** It's a software package you install on your computer to develop Java programs. It includes:
  - **Compiler (javac):** Translates your Java code into bytecode that the computer can understand.
  - **Java Runtime Environment (JRE):** Allows you to run Java programs.
  - **Libraries:** Pre-written code that you can use to build your applications.
  - **Tools:** Utilities like debuggers and documentation generators.

Java Programming by Vikas Sir

JDK(Java Development Kit)



Java Programming by Vikas Sir

Java Source File Structure:-

A Java source file has a specific structure and follows certain conventions to ensure that the code is organized and easily understandable.

Package Declaration:

The package declaration defines the namespace for the classes in the file. It is the first line in the source file if present.

Example:- package com.example.myapp;

Import Statements:

Import statements allow you to use classes from other packages without needing to specify their fully qualified names every time.

Example: import java.util.List; ,import java.util.ArrayList;

Java Programming by Vikas Sir

Class Declaration:

A Java source file can contain multiple classes, but only one public class. The name of the public class must match the filename.

```
public class MyClass {
    // class content
}
```

myclass.java

Constructors:

Constructors are special methods that are called when an object of the class is created. They have the same name as the class and no return type.

```
public MyClass(int number, String name) {
    this.number = number;
    this.name = name;
}
```

Java Programming by Vikas Sir

Main Method (Optional):

Entry point of the application.

```
public static void main(String[] args) {
  MyClass obj= new MyClass(10, "Alice");
  obj.display();
   }
```

Methods:

- Functions that define the behavior of the class.

```
public void display()
{
      System.out.println(Hello);
}
```

```
package com.example.myapp;
        import java.util.List;
        import java.util.ArrayList;
public class MyClass {
    private int number;
    private String name;
    public MyClass(int number, String name) {
        this.number = number;
        this.name = name;
    }
    public void display() {
        System.out.println("Number: " + number + ", Name: " + name);
    }
public static void main(String[] args) {
    MyClass obj = new MyClass(10, "Alice");
    obj.display();
        } }
```

obj2

Java Programming by Vikas Sir

Access modifiers:

- Access modifiers in Java determine the visibility and accessibility of classes, methods, and variables.
- They help to control how different parts of your code can interact with each other.
- There are four main access modifiers in Java: public, protected, default (no modifier), and private.

Java Programming by Vikas Sir

public:

- **Visibility**: Everywhere.

- **Usage**: Any class, method, or variable declared as public can be accessed from any other class, regardless of which package it belongs to.

```
public class MyClass {
    public int myNumber;
public void myMethod()
    {
        // This method can be accessed from anywhere
    }
            }
```

Java Programming by Vikas Sir

private:

- **Visibility:** Within the same class only.
- **Usage:** Any class, method, or variable declared as private can only be accessed within the class it is declared. It is not visible to other classes.

```
public class MyClass {
    private int myNumber;
    private void myMethod() {
        // This method can only be accessed within MyClass
    }
}
```

Class car {
myNumber = 2;
}

Java Programming by Vikas Sir

protected:

- **Visibility:** Within the same package and subclasses (even if they are in different packages).

- **Usage:** Any class, method, or variable declared as protected can be accessed within its own package and by subclasses in other packages.

```
public class MyClass {
        protected int myNumber;
    protected void myMethod() {
        // This method can be accessed within the same package and subclasses
    }
}
```

Java Programming by Vikas Sir

default (no modifier);

- **Visibility:** Within the same package.

- **Usage:** If no access modifier is specified, the class, method, or variable is accessible only within its own package.

```
class MyClass {
        int myNumber;
    void myMethod()
{
    // This method can be accessed within the same package
  }
    }
```

Java Programming by Vikas Sir

Data Types:-

- data types specify the different sizes and values that can be stored in a variable.

int a; → −1, −2, −3, 4, 5, 6, 7, 0

- There are two main categories of data types in Java: primitive data types and reference (or non-primitive) data types.

⇒ 0, 1, −1, −4

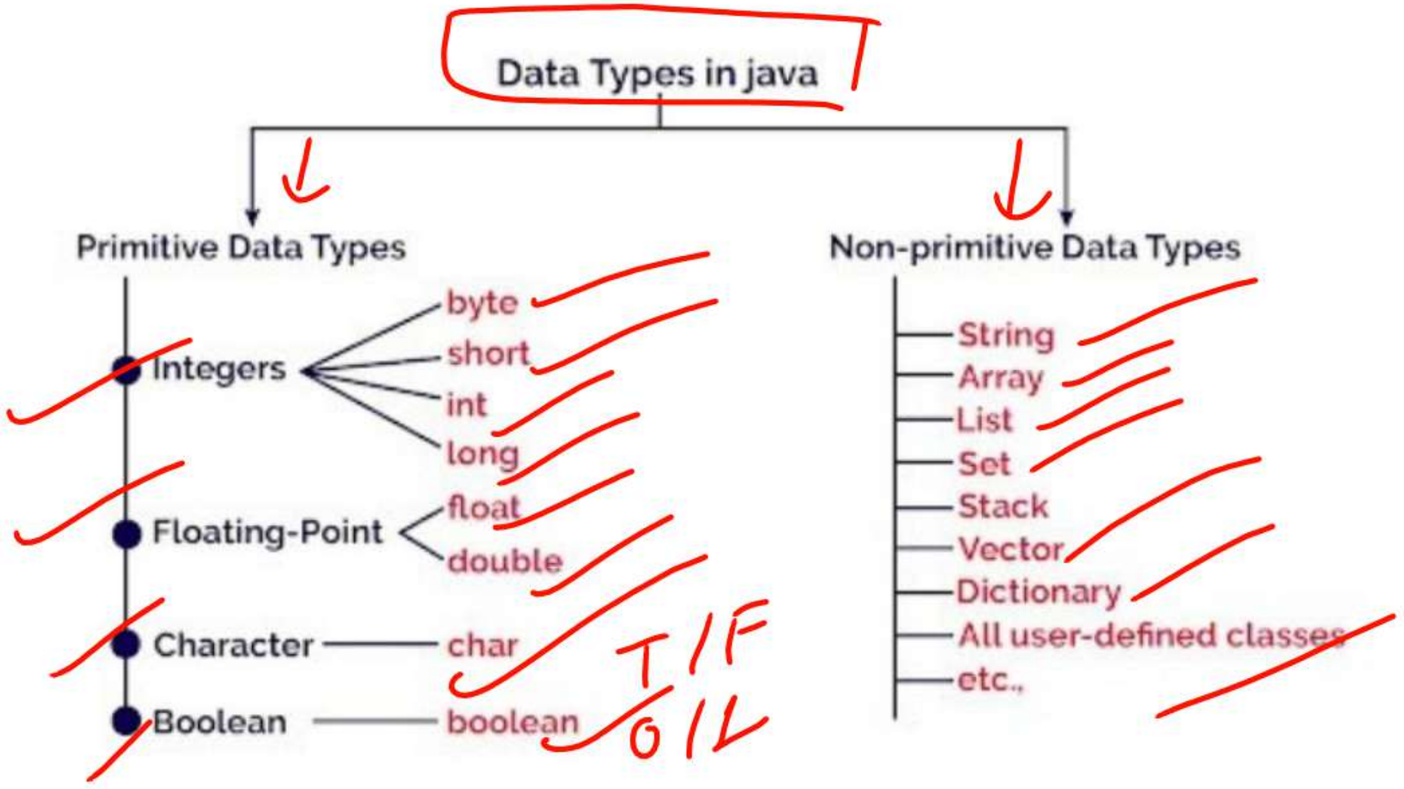
Primitive Data Types

- Primitive data types are the most basic data types available within the Java language. There are eight primitive data types.

Java Programming by Vikas Sir

Reference Data Types

- Reference data types are constructed from primitive data types and refer to objects. These include classes, interfaces, arrays, and enums.



Java Programming by Vikas Sir

Type	Size (in bits)	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2^{31} to $2^{31}-1$
long	64	-2^{63} to $2^{63}-1$
float	32	1.4e-045 to 3.4e+038
double	64	4.9e-324 to 1.8e+308
char	16	0 to 65,535
boolean	1	true or false

1) byte
2
4
8
4
8
2
1

-2^{n-1} to $2^{n-1}-1$

Java Programming by Vikas Sir

OPERATOR IN JAVA :-

- In Java, operators are symbols that perform operations on variables and values.
- They are categorized based on the type of operation they perform.

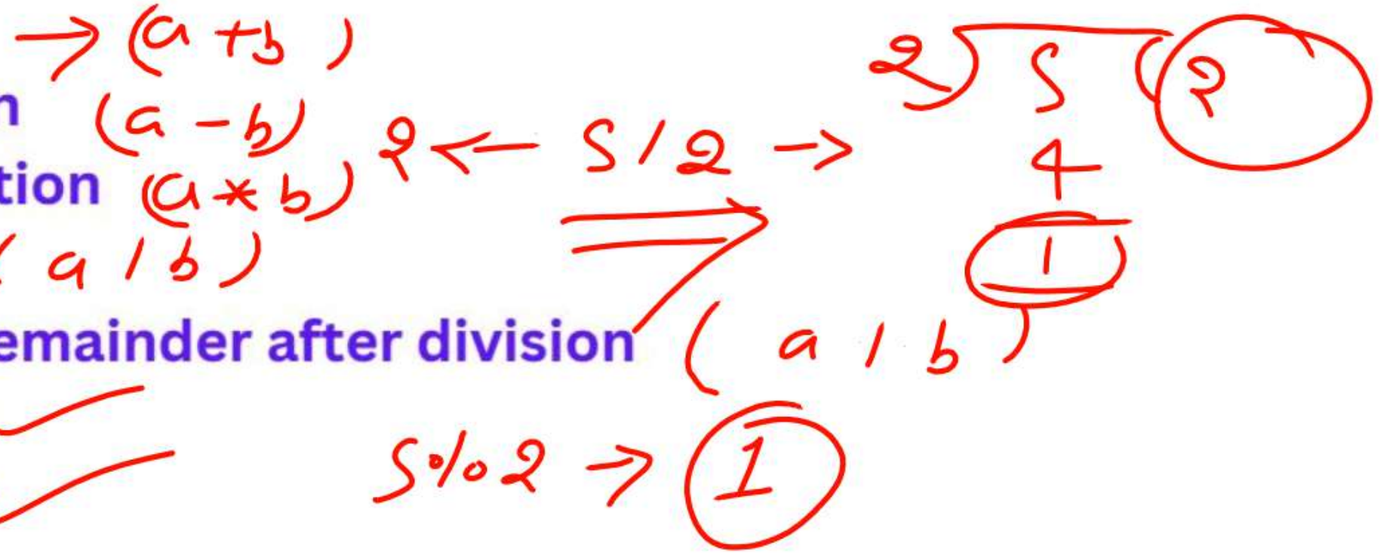
Types Of Operators:

- Arithmetic operators
- Arithmetic operators in Java are used to perform mathematical operations on numeric values, including integers and floating-point numbers.
- Arithmetic operators follow the usual rules of mathematics Of Operator:

Java Programming by Vikas Sir

Common arithmetic operators used in Java:

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus, remainder after division
++	Increment
--	Decrement



- **Relational operators:**
- **Relational operators in Java are used to compare two values and determine the relationship between them.**

Java Programming by Vikas Sir

Relational operators used in Java:

Operator	Meaning
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

a == b

2 == 4 → F / 0

4 == 4 → T / 1

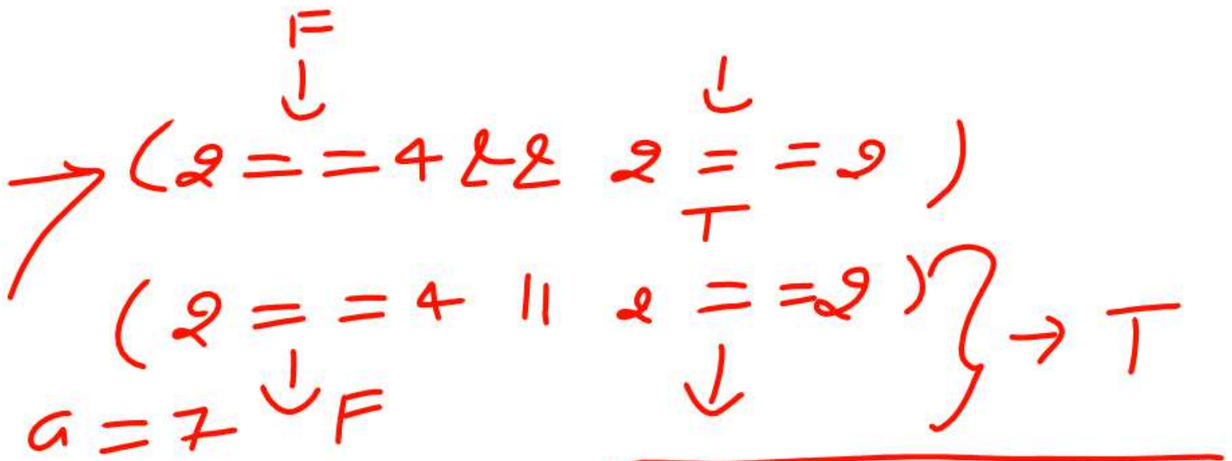
4 != 4 / F

- **Logical operators:**
- Logical operators in Java are used to perform logical operations on boolean values.
- These operators allow you to combine or modify boolean values and make decisions based on the results.

Java Programming by Vikas Sir

Logical operators used in Java:

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT



- **Assignment operators:**

a = 5
a = 7

- Assignment operators in Java are used to assign values to variables.
- These operators combine the assignment of a value with an arithmetic or bitwise operation

Java Programming by Vikas Sir

Assignment operators used in Java:

Operator	Meaning
=	Assignment
+=	Add And assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modules and assign
&=	Bitwise AND and assign
|=	Bitwise OR and assign

a = 5

a += 5 // a = a + 5

a → 10

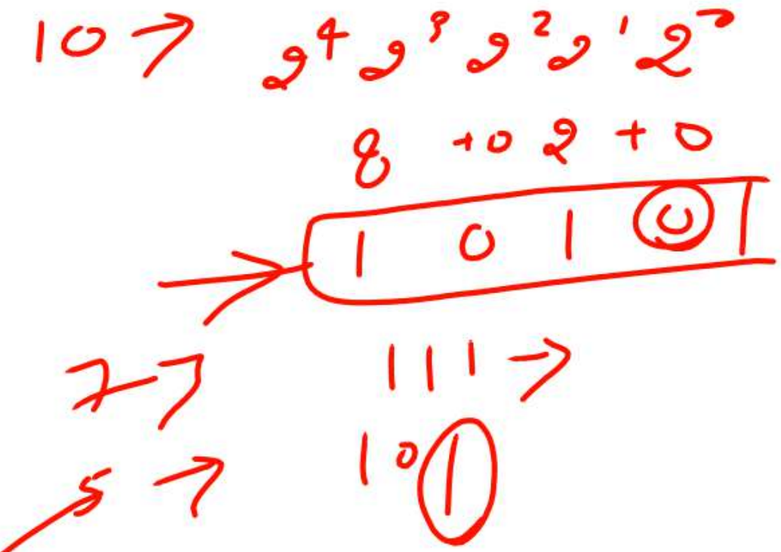
a = a + 5
a = 5 + 5
a = 10

Java Programming by Vikas Sir

- **Bitwise operators:**
- **Bitwise operators in Java are used to perform operations on individual bits of integer types (byte, short, int, long).**

- **These operators treat the values as sequences of binary digits (bits) and manipulate them at the bit level.**

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Left Shift
>>	Right Shift
>>>	Unsigned right shift



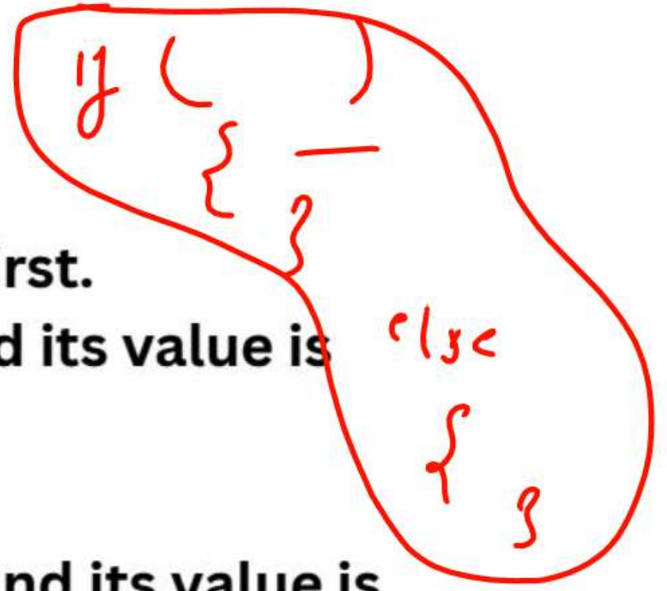
Java Programming by Vikas Sir

- **Conditional Operator:**

- The conditional operator in Java, often referred to as the "ternary Java operator," is a shorthand way of writing an if-else' statement in a single line.

a < b ? P→(a) : P→(b);

syntax: condition? expressionl: expression2



- 'condition' is a boolean expression that is evaluated first.
- If the 'condition' is 'true', 'expression' is executed, and its value is returned.

- If the 'condition' is 'false', 'expression2' is executed, and its value is returned.

Java Programming by Vikas Sir

- **Unary operators:**

- Unary operators in Java are operators that perform operations on a single operand, which can be a variable or an expression.
- They are commonly used in incrementing or decrementing variables.

unary operators in Java

a++
a = 2
a = 2
a++; → 1
P(1) P(3)

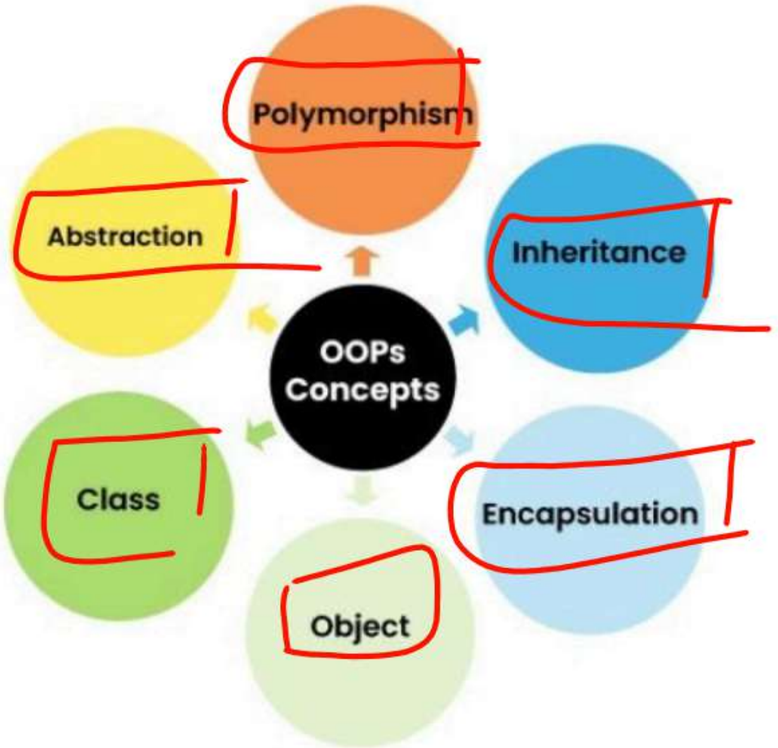
Operator	Meaning
+	Unary plus, used to indicate a positive value
-	Unary minus, used to negate a value
++	Increment
--	Decrement
!	Logical NOT, also used for Boolean negation

Java Programming by Vikas Sir

OOPs(Object-Oriented Programming & System) :-

- Object-Oriented Programming & System (OOPS) concepts in Java helps reduce code complexity and enables the reusability of code.

four main features of (OOP)



Java Programming by Vikas Sir

Objects & Classes:-

- **Objects** are the basic unit of OOPS representing real-life entities. They are invoked with the help of methods.

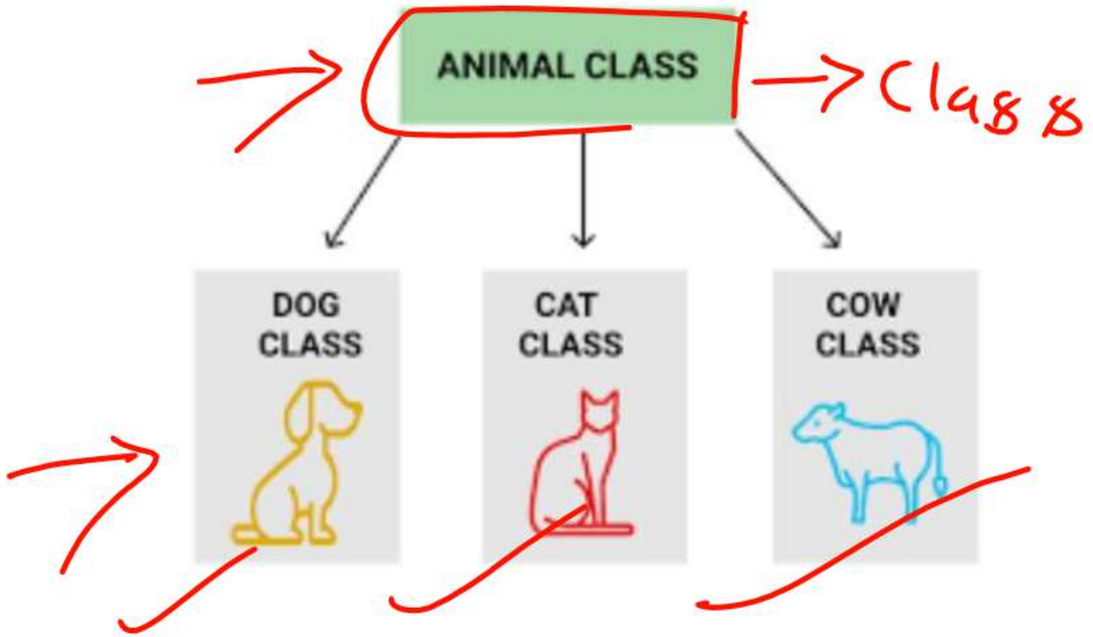
- These methods are declared within a class. Usually, a new keyword is used to create an object of a class in Java.

Abc

Abc obj2 = new Abc();

- **Class:-**
- Class is a predefined or user-defined template from which objects are created. It represents properties/methods that are common to all objects of the same class.

Java Programming by Vikas Sir



Java Programming by Vikas Sir

Abstraction:-

- Abstraction means showing only the relevant details to the end-user and hiding the irrelevant features that serve as a distraction. For example, during an ATM operation.

Encapsulation:

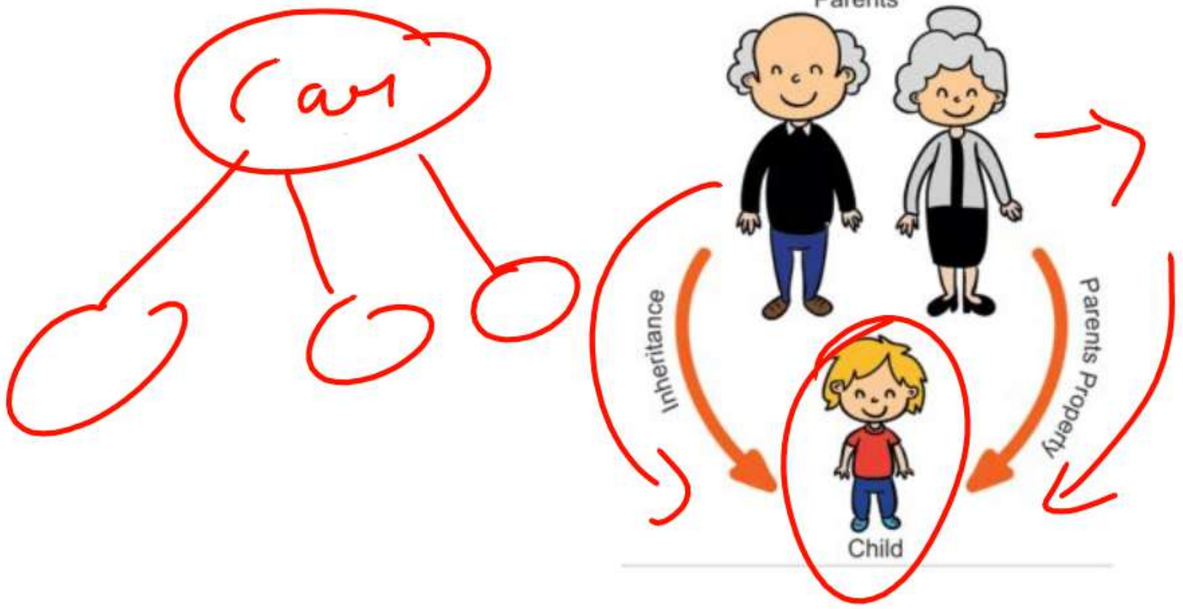
- Bundling the data (attributes) and methods (functions) that operate on the data into a single unit called an object.

- It hides the internal state of the object and only exposes a controlled interface.

Java Programming by Vikas Sir

Inheritance:

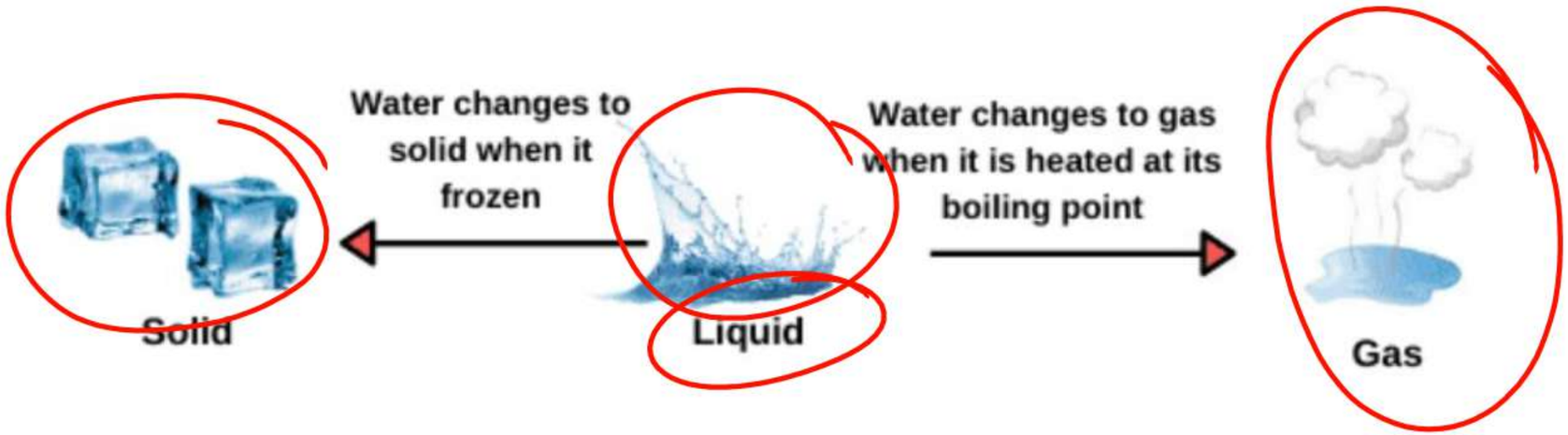
- A mechanism where one class (child class) inherits the attributes and methods from another class (parent class).
- This promotes code reusability.
- Example: Car class inherits from Vehicle class and can use its attributes and methods.



Java Programming by Vikas Sir

Polymorphism:-

- The word “poly” means many and “morphs” means forms, So it means many forms.
- we can define Java Polymorphism as the ability of a message to be displayed in more than one form.
- A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, and an employee.



Java Programming by Vikas Sir

constructors:-

- A constructors in Java is a special method that is used to initialize objects.
- The constructor is called when an object of a class is created.

- Every time an object is created using the new() keyword, at least one constructor is called.

Properties of Constructors:-

- Constructor Should not have any return type
- The access specifier of the constructor can be private, default, protected, public.
- A constructor cannot be declared as final and synchronized.
- A constructor can invoke another constructor of the same class.

Java Programming by Vikas Sir

Example of Constructors:-

```
public class Main {
 int x;
public Main() {
 x = 5;
}
public static void main(String[] args)
{
 Main myObj = new Main();
 System.out.println(myObj.x);
}
}
// Outputs 5
```

Java Programming by Vikas Sir

Types Of Constructors:-

(1)Default Constructor:

- This constructor does not take any parameters. It is used to initialize objects with default values.

```
class MyClass {
    // default constructor is provided by java
}
```

(2)Parameterized Constructor:

- This constructor takes parameters and allows you to initialize objects with specific values.

Java Programming by Vikas Sir

```
class MyClass {
        int value;
MyClass(int val) {
    value = val;
        // initialization code
        }
         }
```

(3)Copy Constructor:

- This constructor creates a new object as a copy of an existing object.
- It typically takes a reference to an object of the same class.

Java Programming by Vikas Sir

```
class MyClass {
  int value;

  // Copy constructor
  MyClass(MyClass obj) {
      value = obj.value;
  }
}

MyClass obj1 = new MyClass(10); // Calls the parameterized constructor
MyClass obj2 = new MyClass(obj1); // Calls the copy constructor
```

Java Programming by Vikas Sir

(4)Private constructor:

- If a constructor is declared as private, then its objects are only accessible from within the declared class.

```
class ClassName {
    private ClassName() {
  // initialization code
        }
        }
```

Java Programming by Vikas Sir

Inheritance:-

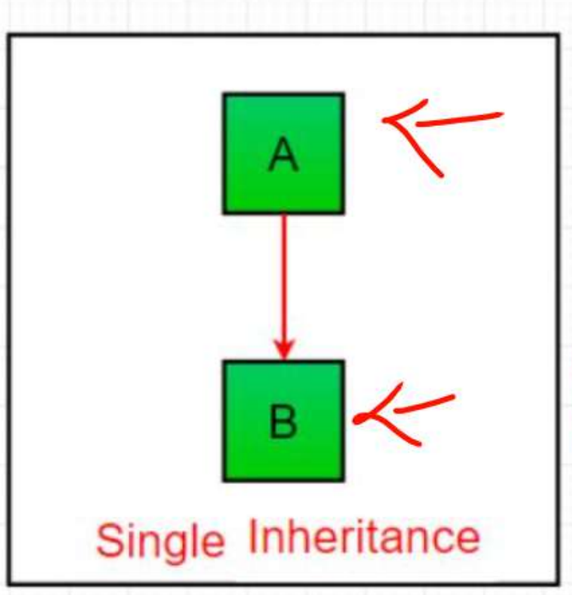
- Inheritance in Java is a fundamental concept that allows one class (called the child or subclass) to inherit features (methods and fields) from another class (called the parent or superclass).
- The child class inherits all the fields and methods of the parent class.
- This means the child class can reuse the code of the parent class without having to rewrite it.
- It promotes code reusability and helps in implementing the "is-a" relationship

Java Programming by Vikas Sir

Types Of inheritance in java:-

Single Inheritance:-

- Single inheritance occurs when a class extends only one superclass. The subclass inherits the properties and behaviors of the superclass.



class B extends A

Java Programming by Vikas Sir

```
class Animal {
 void eat(){
    System.out.println("This animal eats food.");
  }}
class Dog extends Animal {
  void bark() {
    System.out.println("The dog barks.");
  }}
public class SingleInheritance {
  public static void main(String[] args) {
   Dog myDog = new Dog();
  myDog.eat();
    myDog.bark();
  }}
```

myDog →

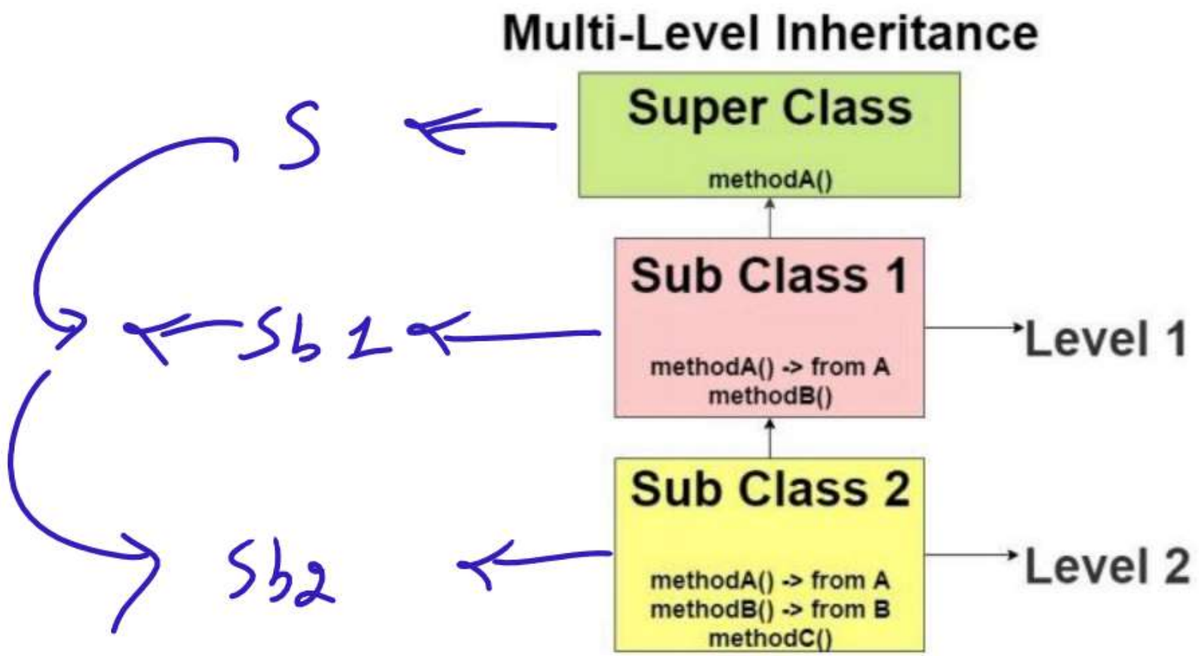
This animal eats food.

The Dog barks.

Java Programming by Vikas Sir

Multi-level Inheritance:-

- Multilevel inheritance in Java is a feature that allows a class to inherit properties and behaviours from another class, which in turn inherits from another class, forming a "chain" of inheritance.



A B C
A, B C, B, A

Java Programming by Vikas Sir

```
class Animal {
 void eat() {
    System.out.println("This animal eats food.");
  }}
class Dog extends Animal {
  void bark() {
    System.out.println("The dog barks.");
  }}
class horse extends Dog{
 void run() {
 System.out.println("The horse is running");
 }
}
```

S

Sub → 1

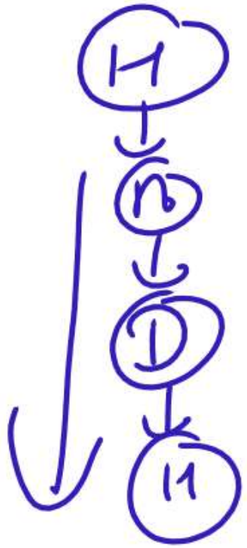
Sub 2 → Dog

H → Dog → animals

→ animals

Java Programming by Vikas Sir

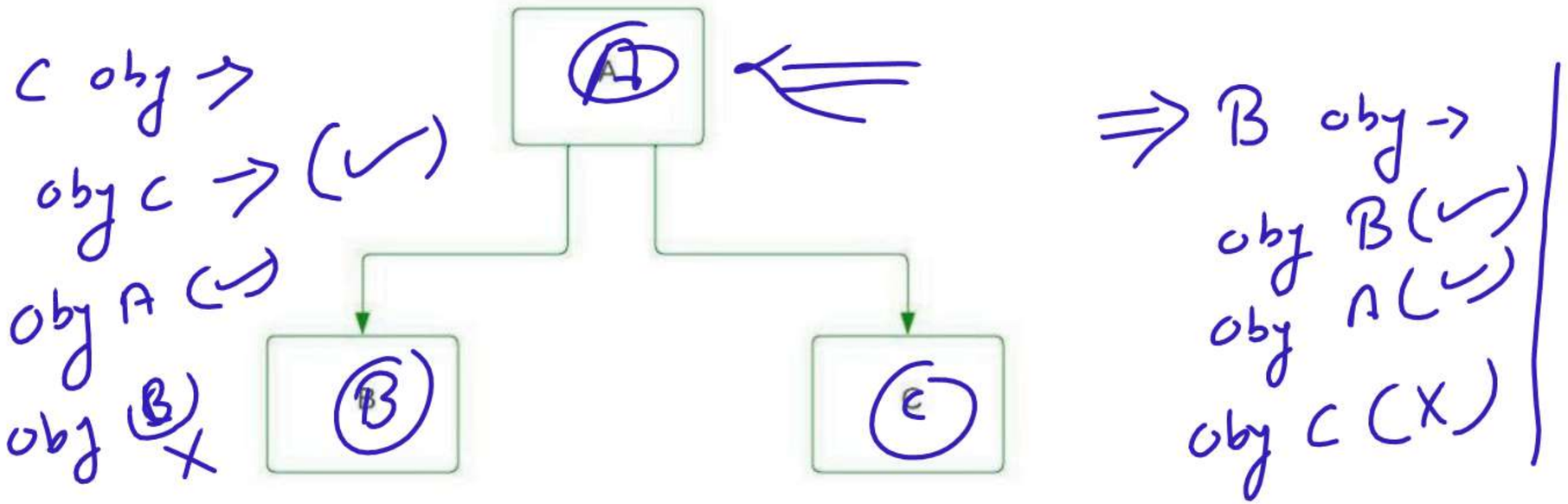
```
public class Multilevel_Inheritance {
   public static void main(String[] args) {
    horse myhorse = new horse();
     myhorse.eat();
     myhorse.bark();
     myhorse.run();
}}
```



Java Programming by Vikas Sir

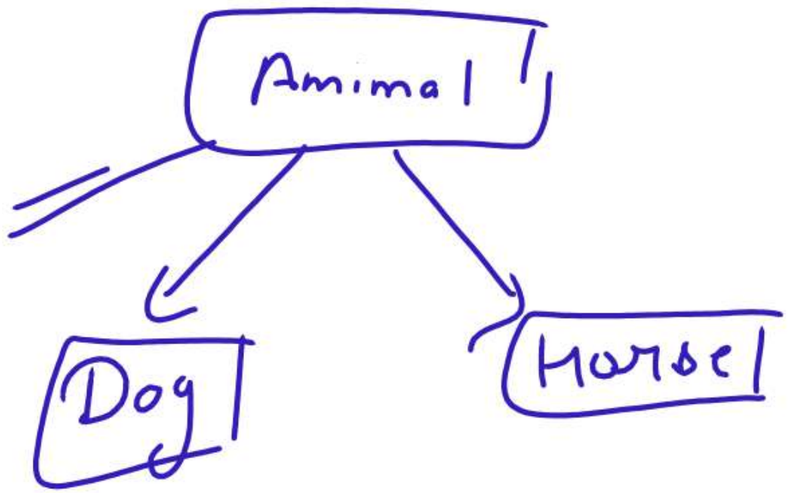
Hierarchical inheritance:

- A type of inheritance in object-oriented programming where multiple derived classes inherit from a single base class.



Java Programming by Vikas Sir

```
class Animal {
 void eat() {
     System.out.println("This animal eats food.");
  }}
class Dog extends Animal{
   void bark() {
     System.out.println("The dog barks.");
   }}
class horse extends Animal{
 void run() {
 System.out.println("The horse is running");
 }
}
```



Java Programming by Vikas Sir

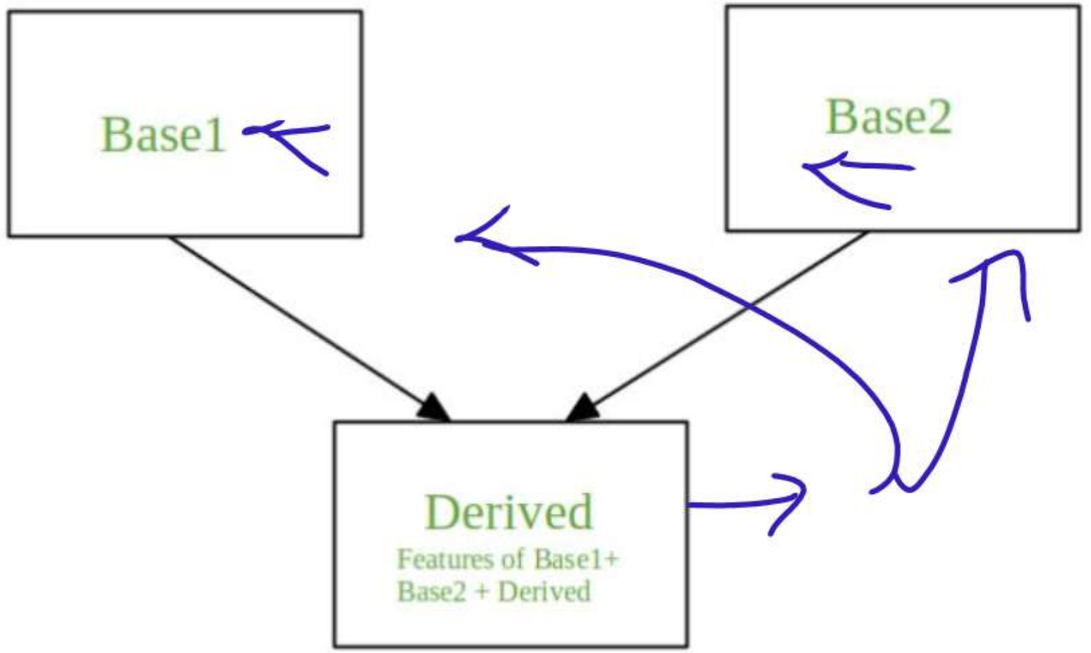
```
public class Hierarchical_Inheritance {
   public static void main(String[] args) {
    horse myhorse = new horse();
     myhorse.eat();
     myhorse.run();
  Dog myobj = new Dog();
     myobj.eat();
     myobj.bark();

   }}
```

Java Programming by Vikas Sir

Multiple Inheritance:

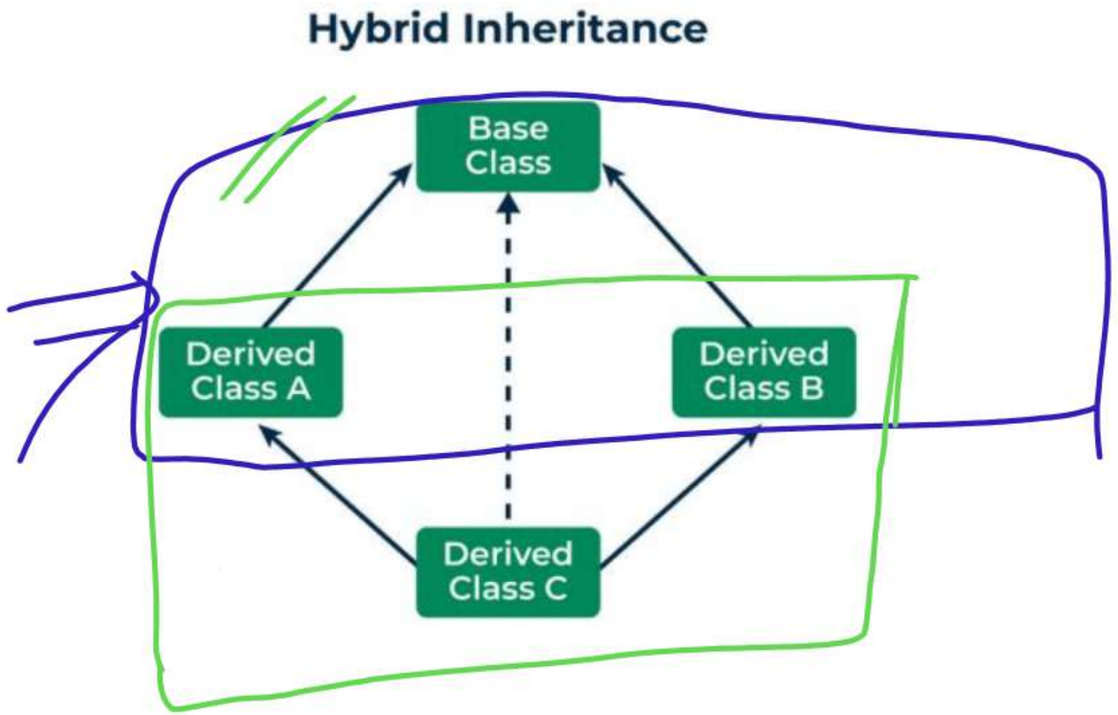
- Java does not support multiple inheritance with classes to avoid complexity and simplify the design, but it can be achieved through interfaces.



Java Programming by Vikas Sir

Hybrid Inheritance:

- A combination of two or more types of inheritance. It can be achieved using interfaces to avoid the diamond problem.



Java Programming by Vikas Sir

Interface In Java:

- The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body.

- It is used to achieve abstraction and multiple inheritances in Java using Interface.
- Java Interface also represents the IS-A relationship.

Syntax for Java Interfaces:

```
interface {
    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

Java Programming by Vikas Sir

Uses of Interfaces in Java:

- It is used to achieve total abstraction.
- java does not support multiple inheritances in the case of class, by using an interface it can achieve multiple inheritances.
- Interfaces are used to implement abstraction.
- Any class can extend only 1 class, but can any class implement an infinite number of interfaces.
- It is also used to achieve loose coupling.

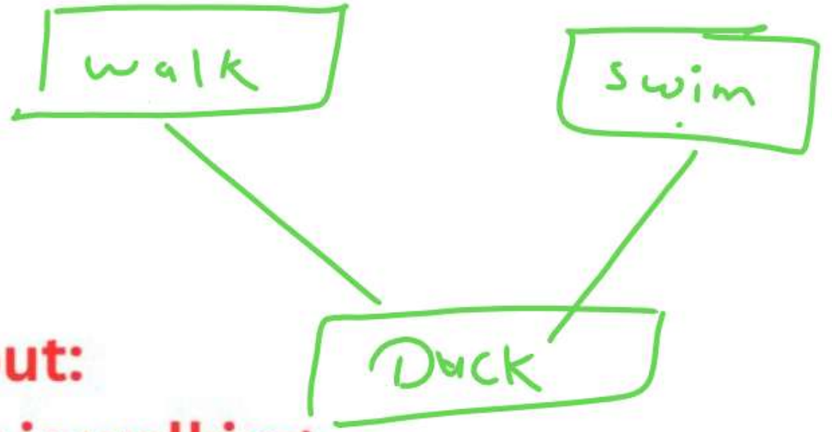
Java Programming by Vikas Sir

Implement Multiple Inheritance by Using Interfaces in Java:

```
Interface Walkable {
 void walk();
}
interface Swimmable {
 void swim();
}
class Duck implements Walkable, Swimmable {
 public void walk()
 {
 System.out.println("Duck is walking.");
 }
```

Java Programming by Vikas Sir

```
public void swim()
 {
 System.out.println("Duck is swimming.");
 } }
class Main {
 public static void main(String[] args)
 {
 Duck duck = new Duck();
 duck.walk();
 duck.swim();
 } }
```



Output:

Duck is walking.

Duck is swimming.

Java Programming by Vikas Sir

Abstract Classes and Methods:

- Data abstraction is the process of hiding certain details and showing only essential information to the user.
- Abstraction can be achieved with either abstract classes or interfaces.

Abstract class:

- is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

Java Programming by Vikas Sir

Abstract method:

- can only be used in an abstract class, and it does not have a body. The
- body is provided by the subclass (inherited from).

Advantage Of Abstract class :

- Inheritance
- Testing
- Abstraction
- Flexibility
- Secure code

Java Programming by Vikas Sir

```
abstract class Animal {
public abstract void animalSound();
}
class Dog extends Animal {
public void animalSound() {
 System.out.println("The Dog is barking ");
}}
class Main {
public static void main(String[] args) {
Dog my = new Dog();
my.animalSound();
}}
```

=> The Dog is barking

Java Programming by Vikas Sir

Polymorphism:-

- The word “poly” means many and “morphs” means forms, So it means many forms.
- we can define Java Polymorphism as the ability of a message to be displayed in more than one form.

Types Of Polymorphism:-

- Compile-time Polymorphism (Method Overloading):

- Compile-time polymorphism is achieved through method overloading. It occurs when multiple methods in the same class have the same name but different parameters (different number of parameters or different types of parameters).

Java Programming by Vikas Sir

Method overloading can be done by changing:

- The number of parameters in two methods. add(a, b), add(a, b, c)
- The data types of the parameters of methods.
  add(int a, int b) : → 10, 20
- The Order of the parameters of methods. add(float a, float b),
  10.2, 11.4

⇒ Add(int a, float b), ⇒ 20.4, 30

⇒ Add(float a, int b)

Java Programming by Vikas Sir

```
class Adder{
 void add(int a, int b){
System.out.println("sum ="+(a+b));
}
void add(int a, int b,int c){
System.out.println("sum="+(a+b+c));
}

public static void main(String[] args){
Adder ad=new Adder();
ad.add(5,6);
ad.add(5,4,7);
}}
```

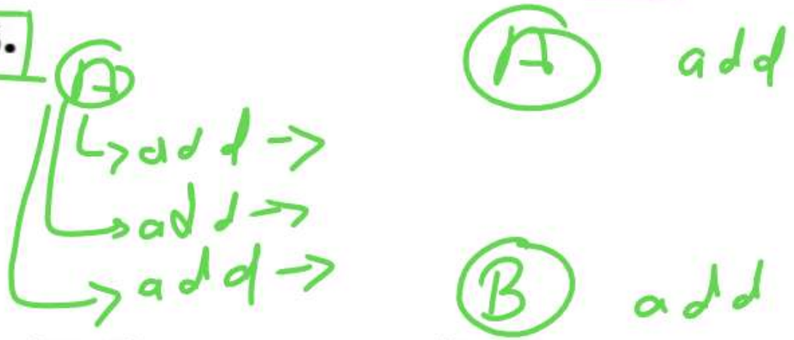
- Sum = 11
- Sum = 16

→ o/p

Java Programming by Vikas Sir

- **Runtime Polymorphism (Method Overriding):**

- Runtime polymorphism is achieved through method overriding. It occurs when a subclass has a method with the same name, return type, and parameters as a method in its superclass.



Rules for Java Method Overriding:

- The method must have the same name as in the parent class.
- The method must have the same parameter as in the parent class.
- There must be an IS-A relationship (inheritance).

```
class Vehicle {
    void move() {
        System.out.println("The vehicle is moving.");
    }}
class Car extends Vehicle {
    void move() {
        System.out.println("The car is driving.");
    }}
public class TestOverriding {
    public static void main(String[] args) {
        Vehicle myVehicle = new Vehicle();
        myVehicle.move();  // Output: The vehicle is moving.
        Car myCar = new Car();
        myCar.move();  // Output: The car is driving.
}}
```

Java Programming by Vikas Sir

Interface

- Interface support multiple inheritance.
- Interface does'n Contains Data Member.
- Interface does'n contains Cunstructors.
- An interface Contains only incomplete member (signature of member).
- Member of interface can not be Static.

Abstract class

- Abstract class does not support multiple inheritance.
- Abstract class contains Data Member.
- Abstract class contains Cunstructors.
- An abstract class Contains both incomplete (abstract) and complete member.
- Only Complete Member of abstract class can be Static

Java Programming by Vikas Sir

Method Overloading

- Multiple methods of same name in single class.
- No need of inheritance, as it is in single class.
- All methods have different signature.
- It's a compile time polymorphism.
- No special keyword used.

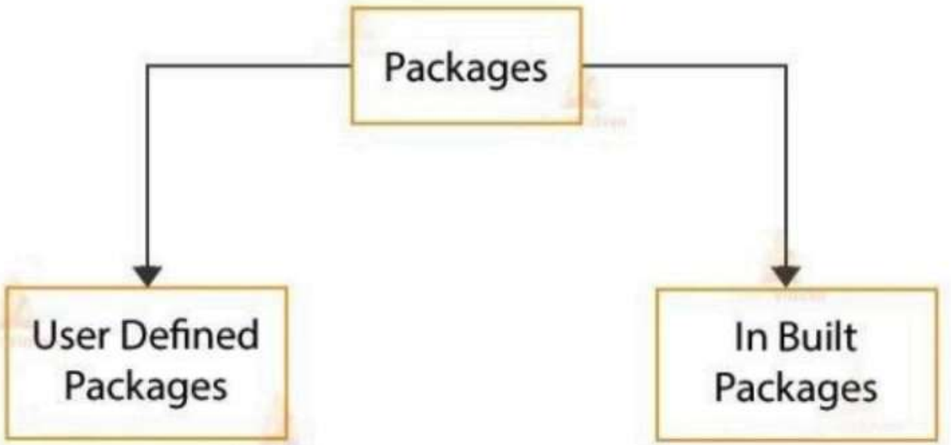
Method Overriding

- Multiple methods of same name in different class.
- Inheritance is used, as it is in different class.
- All methods have same signature.
- It's a run time polymorphism.
- Virtual & override keywords.

Java Programming by Vikas Sir

Package:-

- Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces.



Java Programming by Vikas Sir

Built-in Packages:-

- These packages consist of a large number of classes which are a part of Java API.
- Some of the commonly used built-in packages are:
  java.lang, java.io, java.util. etc

User-defined packages:

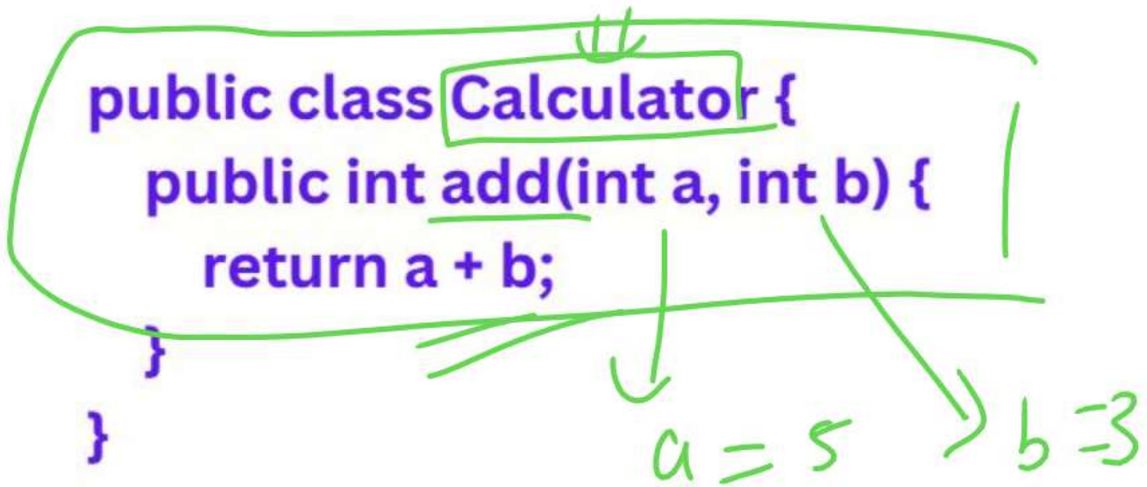
- These are the packages that are defined by the user.
- First we create a directory myPackage (name should be same as the name of the package).
- Then create the MyClass inside the directory with the first statement being the package names.

Java Programming by Vikas Sir

Creating a Package:

```
// Calculator.java
package com.example.math;

public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}
```



Java Programming by Vikas Sir

To use this calculator in another class

```
// Main.java
import com.example.math.Calculator;

public class Main {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        int sum = calc.add(5, 3);
        System.out.println("Sum: " + sum);
    }
}
```

⇒ flow controll stmt.
→ Array

Sum: 8

int arr[score];

Java Programming by Vikas Sir

CLASSPATH IN JAVA

- The classpath in Java is an environment variable that tells the Java compiler (javac) and the Java runtime (java) where to find user-defined classes and packages.

- It is essential when running Java programs that depend on external classes or libraries.

Java Programming by Vikas Sir

Setting the CLASSPATH in Java:

1. Temporary Setting of CLASSPATH

FOR WINDOWS `set CLASSPATH=path1;path2;path3`

Linux/Mac Terminal: `export CLASSPATH=path1:path2:path3`

2. Permanent Setting of CLASSPATH

On Windows:

- Right-click on My Computer or This PC and select Properties.
- Click on Advanced system settings.

Java Programming by Vikas Sir

- Under System variables, click New to add a new CLASSPATH or select an existing CLASSPATH variable and click Edit.

- Enter the paths to your libraries and classes. Separate each path with a semicolon (‘ ; ’).

EXAMPLE:-

.;C:\Program Files\Java\jdk1.8.0_231\lib;C:\myJavaProjects\lib

(.) refers to the current directory.

Java Programming by Vikas Sir

Regular Import

- Uesd to import classes or package.
- Syntax: import.
- package.ClassName;
- Enables you to use the class name without package prefix.
- Example: java import. util.List;

Static Import

- Multiple methods of same name in single class.
- Syntax: import static.
- package.ClassName.*;
- Allow you to use static members without the class name prefix.
- java import static java.lang.Math.*;

Thanks For

Watching

Semester Adda