J. Java Notes

तुम्हारा Result Decide नहीं करता है,
तुम Loser हो कि नहीं तुम्हारी कोशिस Decide करती है।

M T W T F S S
Page No.:
Date:
YOUVA

OOP's

⇒ Object Oriented programming language of Java Programming which is used for designing a program using Classes and Objects.
Oop's can also be characterized as data controlling for accessing the code.

⇒ function = Method define in the Class Called function.
or
⇒ Any function define in the Class Called ~~function~~ Method.

= Class Pen { // Define the Pen blue pen
String color;
String type; // ballpoint, gel

// Creat A Method
public void Write () {
Sysout (" Writing Something ");
}
// Creat Method Colour Bring
public void Printcolor () {
Sysout (this.color);
// this called the Current properties.
}
}

// Create Public Class and Main Method
Public class oops {
main () {
// Create Class Method Object using new Keyword
Pen pen1 = new Pen ();
pen1.color = "blue";
// (.) dot operator to Assign the properties

```
pen1.type = "gel";
// Create ob one More Object

Pen pen2 = new Pen();
pen2.color = "black";
pen2.type = "ballpoint";

// Called the function/Method Using
   (.) dot operator.
pen1.printColor();
pen2.printColor();
```

output — blue, black

→ Classes does not any Return type like - int, boolean...

```
class Student {
    String name;
    int age;
// Create function
public void printInfo() {
    Sysout(this.name);
    Sysout(this.age);
}
}

public class Oops {
    public static void main(String args[]){
        Student S1 = new student();
        S1.name = "Shradha";
        S1.age = "22";
        S1.PrintInfo();
    }
}
```

Note ⇒ define one line—

Student sz = new Student ();

⇒ new ⇒ Keyword Jaise hum new Keyword lagaya So Memory Heap ke Andar Space Allocate Ho Jayegi. // Is Space ke Andar Hamari puri Object Jake Store ho Jayegi.

⇒ Student () ⇒ basically "()" paranthisis lagte hai Jaha par function Hote hai.
└ basically this is A function, Special type of function in Java. Called Constructor

⇒ Constructor =
that mins kuchh Construct Karna, Yo kuchh banana.

= Java ke Andar Constructor ka kaam hota hai Java ke Object ko Construct Karna

⇒ Java have A three type Constructor—
(1) ↳ Kitte object ko hum 3 type se bana Sakte hain.

(I) Non - Parameterized Constructor ⇒
↳ Vo Constructor Inside Not declired the parameters.

Note ⇒ Constructor Same As Class Name hoga
⇒ Constructor does not Return any things.
⇒ One obj ke liye Constructor only one baar hi call hoga.

Ex -
```
student () {
    Sysout ("Non / default Constructor");
}
```

Note ⇒ Agar hum Default Constructor Nahi Write Karte hain to Java Itself bana deta hai.

* (2) = Parameterized Constructor

```
Student (String name , int age) {
    this. name = name;
    this. age = age;
}

// Public Class and Main Method
public class Oops {
    main () {
    Student s2 = new Student ("shradha", 22);
    s2. printInfo ();
    }
}
```

⇒ (3) Copy Constructor

↳ this is Concept of c++ lang.

↳ Work is To copy one obj value and asign Other Obj that is Copy Constructor.

```
Student ( Student s2) {
    this. name = s2. name;
    this. age = s2. age;
}
// Ek default Constructor define Karna hog
   Student s1 Ke liye.
```

Note ⇒ Unlike language like C++, Java has no Destructor. Instead, Java has efficient Garbage Collector that deallocates memory automatically.

```
Student () {

}

// Called Copy - Constructor

public class Oops {
    main () {
        Student S1 = new Student () {
            S1.name = "Shradha";
            S1.age = 22;

        Student S2 = new Student (S1);
        S2.printInfo ();
```

Note = Not define Any properties Is S2, But Asign the hole define S1 properties.

⇒ Oops four Concepts —

1. Polymorphism
2. Inheritance
3. Encapsulation
4. Abstraction

Polymorphism — Polymorphism is the ability to present the same Interface for differing underlying form (data types). With polymorphism, each of these classes will have different underlying data. Precisely, Poly means "many" and morphism means "forms".

OR

⇒ Ability of an object to take multiple forms is also called polymorphism.

Prog⇒
```
class Student {
    String name;
    int age;
//all function ___ Same name_

public void printInfo (String name) {
        Sysout (name);
    }
public void printInfo ( int age) {
        Sysout ( age);
    }
public void printInfo ( String name , int age) {
    Sysout ( name + " " + age);
    }
}
```

Note⇒ Here I'm Write the three function But all the function Names Are Same (printInfo). Is Implimention Ko Polymorphism Kahte hain.

Note= here Same function (name) ko baar-baar Used Kiya Alag-Alag Purpose ke liye.

Note⇒ In Sabhi Concept ko Java Ke Andar function Overloading Kahte hai

```
public class OOps {
    main () {
    Student Sz = new Student ();
    Sz. name = "Aman";
    Sz. age = 24;

//  Sz. printInfo ( Sz.age);
    Sz. printInfo (Sz. name , Sz. age);
    }
}
```

Imp

⇒ Type of Polymorphism

① Compile Time Polymorphism (Static)
② Runtime Polymorphism (Dynamic)

① Compile Time Polymorphism :—

The polymorphism which is implemented at the Compile Time is known as Compile-time polymorphism.
Ex— Method Overloading

⇒ Method Overloading— Method overloading is a technique which allows you to have more than one function with the same function name but with different functionality.

② Runtime Polymorphism— It is Dynamic Polymorphism.
Ex— Function Overriding

Function Overriding— Function Overriding means when the child class contains the method which is already present in the parent class. Here, the child class overrides the method of the parent class.

Note⇒ In case of function overriding, Parent and Child classes both contain the same function with a different definition. The call to the function is determined at Runtime is known as Runtime polymorphism.

#(2) Inheritance — Inheritance is a process in which one object acquires all the properties and Behaviors of it's parent object Automatically. In such a way, you can Reuse Extend or modify the Attributes and behaviors which are defined in other classes

⇒ In Java, the class which Inherits the Members of another class is called Derived class and the class whose Members are Inherited is called base class.

Imp Note ⇒ Jab Ak class ki properties or Method ko Koi dusari Class Inherit ✗ kar leti hai or le leti hai, It's called Inheritance.

⇒ Used Inheritance, To Re-Usebility banati hai

Ex =

```
class Shape {
    String color;
}
class Triangle extends Shape {
}

public class Oops {
    main () {
        Triangle t1 = new Triangle ();
        t1.color = "Red";
    }
}
```

Derived class → Base class

⇒ Type of Inheritance—

① Single Inheritance ⇒ When one Class Inherits Another Class, It is known as Single level Inheritance.

→ Base Class / Parent Class / Super class

Prog = class Shape {
public void area () {
Sysout (" displays Area of Shape ");
}
}

→ Derived class / child class / sub class

class Triangle extends Shape {

public void area (int h, int b) {
Sysout (('1/2') * b * h);
}
}

```
public class oops {
    main () {
        Triangle Tr = new Triangle();
        Tr.area(2,3);
```

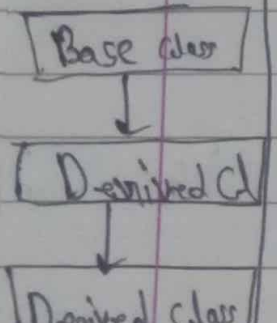
output — 6

② Multilevel Inheritance — Multilevel Inheritance is a process of deriving a Class from another derived class

Note → Mtlb AK se jyada level hamne Create Kar liya

Prog ⇒ class Shape {
public void area () {
Sysout (" displays Area of Shape ");
}
}

class Triangle extends Shape {
public void area (int h, int b) {
Sysout ((1/2) * b * h);
}
}

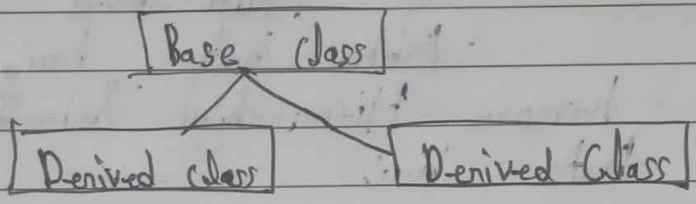


→ Triangle ki Sabhi side Equal hoti hai

```
class Equilateral Triangle extends Triangle {
    public void area (int h, int b) {
        Sysout ((1/2)*b*h);
    }
}
```

= Expl. Here Equilateral Triangle ~~ka~~ Triangle ka derived class banega.

③ Hierarchial Inheritance = Hierarchical Inheritance is defined as the process of deriving more than one class from a base class.

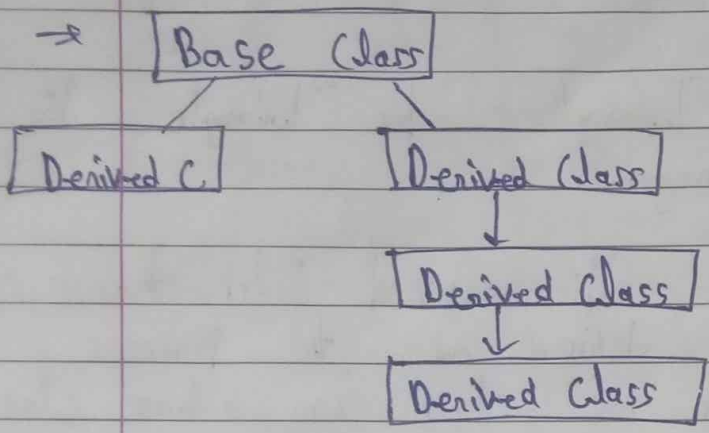


Note = Ak Base Class ko Multiple child class / derived class ne ~~kiya hai~~ Inherit kar rakha hai ~~tdat~~ called is Hierarchical Inheritance

Prg-

```
class Shape {
    public void area () {
        Sysout (" display Area of Shape");
    }
}
class Triangle extends Shape {
    public void area (int h, int b) {
        Sysout (" " + (1/2)*b*h);
    }
}
class Circle extends Shape {
    public void area (int r) {
        Sysout ((3.14)*r*r);
    }
}
```

④ Hybrid Inheritance = Hybrid Inheritance is a Combination of Simple, multiple Inheritance hierarchical Inheritance.

→ 

⇒ Iss hamare different-2 type ke Inheritance Ak hi Jagah dekhe ko Mill jate hain. Mtlb Kissi Branch Me hamara Single-level Inheritance chal Raha hota hai, Kissi me Multi-level Inheritance chal Raha hota hai, Aur Kahi par hamara Hierarchial Inheritance chal Raha hota hai.

Package In Java

= Package is a group of Similar types of Classes, Interfaces and Sub-packages.

= Packages can be built-in or User defined.

⇒ Packages that mins Ak Cartage Jiya Usme Releated Saman ko Store Kar diya. Yah Yani Ak Jaise Saman Ka..

= like- Ak packages pen Ka — Issme all pen Store
Ak package pencil Ka — All pencil store
etc

= Issi Tarah Code hame logically Write Karna hota hai, that mins code hamara Idhar-Udhar faila Nahi hona chahiye.

→ that mins Kaun si cheese Kiske liye Accessible hogi

Page No.: Date: YOUVA

Access Modifiers in Java → 4 types

Public

① # ~~Public~~ = The access level of a public Modifier is everywhere. It can be accessed from within the class, Outside the class, within the package and outside the package.

~~pvg~~ package bank;

class Account {
    public String name;
}

Public that mins Koi bhi Account Class Ka object bana liya vo Name ko Access Kar Sakta hai, ye Information

2. Public → Hamare Puri Class ke mein Koi bhi Access Kar Sakta hai, Sath ke Sath Koi Dusare Package Me bhi Access Kar Sakta hai.

② Default ⇒ The access level of a default modifier is only within the package. It cannot be Accessed from OutSide the package.

⇒ default access Modifier hame write nahi Karna hota hai. Agar Kissi properties or function ke Aage Kuch bhi nahi lagate ho Default type Ka hota hai.

Eg 2

class Account {
    String name; // Default
}

(3) Protected — The Access level of a protected Modifier is within the package and OutSide the package through Child class.
If you do not make the child class, It cannot be accessed from outside the package.

Note ⇒ Protected → that mintes Usse apne Package me Saari Cheeye Accessible hogi Sath ke Sath Dusre Package me karna hai To Only Sub-Class hi kar Sakti hai.

Prg=
```
package bank;
    class Account {
        protected String email;
    }
public class Bank {
    main() {
        Account acc1 = new Account();
        acc1.email = "raviKumar@gmail.com";
```

(4) Private ⇒ The Access level of a private modifier is only within the Class.
It Can Not be Accessed from outside the Class.

Note ⇒ Private Ko Class ke bahar koi bhi Access Nahi kar Sakta.

```
class Account {
    private String password;
}
public class Bank {
    main() {
        Account acc1 = new Account();
```

acct.password = "R117";

that is Not Accessible for Bank Class.

⇒ So Private modifier ko Access ~~kar~~ Karne ke liye Java ke Andar Used Used— getter & Setter.

getter = that mins Private Cheej ki Information Jena.

setter ⇒ that mins Uss Private Cheej ki koi Value Set Kar diji ye.

Prog—

```
class Account {
    private String password;
    // getter / Setter
    // Create getter function
    public String getPassword() {
        return this.password;
    }
    public void setPassword (String Pass) {
        this.password = Pass;
    }
}

public class Bank {
    main () {
        Account acct = new Account();
        acct.setPassword ("R117");
        Sysout (acct.getPassword());
    }
}
```

#(3) Encapsulation =

Meaning of Encapsulation is, To make sure that "Sensitive" data is "hidden" from User-

* Declare Class Variable / Attribute as Private.
* Provides public Get & Set Methods to Access and Update the value of private Variable.

⇒ Data hiding ⇒ A language feature to Restrict Access to Members of an objects, reducing the negative effect due to dependencies-
e.g. — "protected", "private" feature in Java.

Note ⇒ Encapsulation- Mtlb Aap Data Aur Usske function ko Combine Karde Aap Usske Ak hi Entity Me.
Yani Jab bhi Aap Ak Class banate hai — To Aaap Karte hain — To Data Aur Usske Function
Data - Yani hamari Class ki Properties,
Function — Yani Usme likhe huye Method
Unn Dano ko Combine Kar ke Hamne Ak Unit ke Andar Store Kar diya Iss Unit ko Hmmne Java ke Andar name diya hai Class
Ab Inn Sabhi ko Hum Encapsulation kahte hai, English me Mtlb Kissi cheez ko Encapsulate Kar dena Yani Rakh dena.

Note ⇒ Encapsulation se Data hiding ka Concept possible ho pata hai: Data hiding = that Mins kuchh data hai Jo User ko dikhana hai kuchh chhupa hai. with the help of Access Modifier

④ Abstraction ⇒ We try to obtain an abstract view, Model or structure of a real life problem, and reduce its Unnecessary details. With definition of properties of problems, Including the data which are affected and the operations which are Identified the model abstracted from problems can be a standards solution to this type of problems.

⇒ In Simple Terms, It is hiding the Unnecessary details & showing only the essential parts/ functionalities to the User.

Data binding ⇒ Data binding is a process of binding the application UI and Business Logic. Any Change made in the Business Logic will Reflect directly to the application UI.

Note ⇒ Abstract ⇒ that mins Important cheeje User ko dikha dena Aur Non-Important cheeje User se Hide kar dena

Note ⇒ Abstraction And Data hiding ⇒ Data hiding is the process of protecting members of Class from unintended changes whereas, Abstraction is hiding the implementation details and showing only Important / Useful part to the user.

= Abstraction is Achieved in 2 way's —

① - Abstract class

↳ 1-Way ⇒ abstract Keyward Used Karen, abstract classess banaye, function banaye, properties banaye.

② - Interfaces (Pure Abstraction)

Discuss

Ans → 1st create Animal - aur Animal ke liye hum ak Blue-print bana lenge. Uske baad different Type's of Animal ko Create kar lenge.

Note → Inheritance

Ak Class ka Blue-print banane ka middle hota hai Usko Base Class. Inheritance me.
Aur Ak Object banane ka blue-print hota hai Uski Class.

⇒
```
class Animal {
    public void walk(){
    }
}
class Horse extends Animal {
    public void walk() {
        Sysout("walk in 4 legs");
    }
}
class Chicken extends Animal {
    public void walk() {
        Sysout("walk in 2 legs");
    }
}
```

Question ⇒ Jab bhi Hum Horse bana rahe honge ya chicken bana rahe honge To Usske liye Animal (Class me kya) (hamne) likha tha dekhne ki jaroorat hai. Because Animal Class ki jitti bhi properties hai Vo to Horse class And chicken Class me aa hi rahi hain. To User ke pass ye Right ho ki Vo Animal ko bhi Walk kara sake.

⇒ Yaha pe Ak Tarah Se Animal Class E-Relevant hai, Usski chez hame User ~~Animal~~ ko dikhane ki jaroorat Nahi hai.

⇒ Waise bhi Animal ko koi bhi Relevant Object To banayenge Nahi kyo, kyoki Animal koi Spesific Animal To hai hi Nahi jiska hame Object banana pade. Hum Horse se New Horse Create kar sakte hain, Chicken se New chicken Create kar Sakte hain. Animal ko koi type hi nahi hai.

Page No.: YOUVA
Date:

↳ Ye only ek Concept hai, Animal Ak only Concept Not exits.

To

```
abstract class Animal {
    abstract void walk();
}
```

Note:- Abstract class ke Andar kuchh function or properties ka abstract bana sakte hain. Jaise walk() function ko koi Kaam Nahi hai only Exits karta hai—

abstract void walk(); fun Abstract bana diya To
↳ Implementation write karne ki jaroorat nahi

⇒ Mtlb Ak walk type ka function hoga To har type ke Animal me Exist hoga.

```
class Horse extends Animal {
    public void walk() {
        Sysout ("walk in 4 legs");
    }
}

public class oops {
    main() {
        Horse horse = new Horse();
        horse.walk();
// Error  Animal animal = new Animal();
//        animal.walk()
//  ↳ throw the Error Runtime Error.
```

⇒ Prop

⇒ Properties Abstract Class—

(i) An abstract class must be declared with an abstract keyword.
(ii) It can have abstract and non-abstract methods
↳ Ex—

```
abstract class Animal {
    abstract void walk();
    public void eat() {
        Sysout("Animal eats");
    }
}

class Horse extends Animal {
    public void walk() {
        Sysout("walk on 4 legs");
    }
}

public class oops {
    main() {
    Horse horse = new Horse();
    horse.walk();
    horse.eat();
```

Non-abstract Method

(iii) It cannot be Instantiated // Non-Create a Object

(iv) It can have Constructors and Static methods also.

Eg—

```
abstract class Animal {
    abstract void walk();
// Constructors
Animal() {
    Sysout("You are About to create an Animal.");
}
class Horse extends Animal {
```

Called Process - (1) Static (2) Heredar (3) Method (4) Constructor

Inst

// Constructor

```
Horse () {
    Sys.Out ("Wow you have created a Horse!");
}

public class oops {
    main () {
        Horse horse = new Horse ();
    }
}
```

Exp Jaise Object bana than Constructor Called —
output : You are about to create an Animal."
wow, you have created a Horse

Exp= Mild Jb bhi hum Derived Class ko koi Object Created karte hai to 1st Base Class ko Constructor Called hoga than Derived Class ka.

Note ⇒ Iss process ko Java ke Andar Constructor Chaning Kahte hain.

(25)= Interfaces (Pure Abstraction)

Exp ⇒ Abstract Class ⇒ Me Non-abstract Method write kiya — .Ak eat() function banaya jise Horse Class bhi Usse kar raha, Aur Chiken Class bhi Usse kar raha. Jaki eat() function abstract Nahi hai phir phi Java Ne Allow kar diya, So ye pure Abstraction Nahi hai.

Note= pure Abstraction ka kaam/Mild Saari Useless Information ko Hide kar degi.
Aur Ja Saari Usefull Information User ke liye Use dikha degi.

⇒ So ye Ada hai with the help of Interfaces.

Note ⇒ Interface, Class Jaise hi Hote hai. Bss Unme kuchh fixed Properties Hoti hai—

```
Prog—   interface Animal {
            public void walk ();
//Constructure.  Animal () {          // Interface ke Constructors nahi hote.
                }                      // Interface cannot have Constructors.
            // function
            void eat () {
            }
```

Exp ⇒ Interface Animal ke Andar kai bhi Non-abstract function nahi hona chahiye. Nahi function ki Implementation honi chahiye.

```
Prog=   interface Animal {
            public walk ();
        }
        class Horse implements Animal {
            public void walk () {
                sysout (" walks on 4 legs");
            }
        }
        public class oops {
            main () {
                Horse horse = new Horse ();
                horse.walk ();
            }
        }
```

→ Properties —

① All the fields in interfaces are public, Static and final by default.

Ex= interface Animal {
    int eyes = 2;
    public void walk();
}

Exp= eyes = Ki value / Sabhi ke liye Same hogi Isse Hum change Nahi Kar Sake.
Static = Yahi Sabhi ke liye Same hogi
final ⇒ final rahegi Value Hamesha fix hogi. ~~&~~ Aur public hogi.

② All Methods are public & abstract by default.

③ ~~It~~ Interfaces support the functionality of "Multiple Inheritance".

Ex =
interface Animal {
    void walk();
}
→ mins only eats plants
interface Herbivore {
}
⇒ Multiple Inheritance
class Horse implements Animal, Herbivore {
    public void walk() {
        Sysout("walks on 4 legs");
    }
}
public class Cops {
    main() {
        Horse horse = new Horse();
        horse.walk();
    }
}

Exp⇒ Multiple Inheritance =

→ Horse ek Animal bhi hai Aur Herbivore bhi "ki only plants eats karta hai".
So Is Tarike se Dono Class ki properties Ek Saath Le li / Inherit kar li Horse Ne.

Imp Note= that is Called Multiple Inheritance.
Jo Java ke Andar Classes se Nahi Hota interfaces se hota hai.

Static Keyword

Exp=
⇒ Vo Cheej Jo Sabke liye Accesible hai = "public ki Tarah Nahi"
= Static hum Unn properties ko batate hain Jo Class ke liye common ho.

Ex—

```
class Student {
    String name;
    // String school;
```

↳ Socho Agar To har / Sabhi Student ke liye School To Ek hi hoga.
so—

```
static String school;
```

↳ Mtlb Student Class ke Jitne bhi object banenge School Ka Name Same hoga.

```
public class oops {
    main () {
        Student.school = "IET Acts";
    }
}
```

Exp= Abhi Tak hum Student Class ko Object 1st Create Karte the. phir (.) dot Karke cheejo ko Access Karte the.

Imp

⇒ But - Jise bhi ~~Sai~~ hum Static banate hai—
• static method/function, static variable Unko Hum Class ke Name se Access kar Sakte hain.

Note⇒ Maan Jiske KL ko School ka Name change ho gaya. To Student.school = "IET Ads Pune" Saare ke Saare Student ka Name Ak Saath hi change ho Jayega.

Note= Static hum Unn properties / Cheej ka banate hai Jinko hum chahte hain Vo Sabhi ke liye Common hai

```
// function / method
public static void changeSchool() {
    school = "Dr. RMLAU";
}
```

⇒ 4 Types Used in Static keyword—
① Static Keyword is Used any properties Eg = static string sch
② Static Keyword is Used in function ⇒ public static void changeSchool()
③ an block ke Saamne Used kar Sakte hain
④ on Used in Nested Classes

Note⇒= Memory me → Static cheejo ko Only Ak baar hi Memory Assign hoti hai.
Aur Jo object ki chije hoti hain Unko Memory baar-baar di Jaati hai.

Note⇒ oops ka main work hai Cheejo ko Systmeticallly Organized Karna. Kuchh ~~Achhe~~ Achhe Concept ko Implement Karna.