**Day 1 : 2 hour : 2 to 4**

**Java :** Java is platform independent and pure object oriented programming language.

1970 🡪C

1980 ->C++

1990 🡪 Python and Java

2000🡪 .net

2010 🡪 JavaScript

2020 🡪 Machine learning and AI, Data Science

Object 🡪 object is any real world entity.

Properties or state . have -🡪

Person

Behaviour . do/does -🡪

Bank

Animal

Car

Customer

Employee

Class 🡪 class is blue print of object or template of object or user defined data type which help to create the object or describe the object.

Syntax to write the class.

class ClassName {

variables or fields.

Method or functions.

}

Class name must be follow pascal naming rules.

1. If class contains one word first letter in upper case. Like Test, Demo, Employee,Customer etc
2. If class container more than one word each word first letter upper case EmployeeDetails, CutomerInfo etc.

Java 1.0 1.2 to 1.8, 9, 18 etc.

From java 11 onward java is not open source.

Data types are divided into two types

1. Primitive : it is use to store only value

8 types

1. byte
2. short
3. int
4. long
5. float
6. double
7. char
8. boolean
9. Non primitive or reference data types : it use to store value as well as reference of another data type.

4 types

1. array
2. class (pre defined or user defined) : string
3. interface ( pre defined or user defined) :
4. enum ( pre defined or user defined)

Classname objectReferenceName = new ClassName();

objectReferenceName.methodName();

types of variable or fields.

In Java variable are divided into 3 types.

1. Instance variable
   1. The variable which declared inside a class but outside a method is known as instance variable.
   2. The instance variable hold default value according to their data types

int family value is 0, float family 0.0, char space, Boolean false, string null.

* 1. Instance variable we can access in that class in all method but method must be non static. Inside non static method of same class we can access directly.

1. Local variable
   1. The variable which declared inside a method is known as local variable.
   2. The local variable doesn’t hold default value we have to initialize.
   3. The scope of the variable within that method where it declared.
2. Static variable

Constructor : constructor is a type of special method which help to create the memory.

Pts

1. Constructor have same name as class itself.
2. Constructor no need to call it will call automatically when we create the object of that class.
3. Constructor doesn’t contains return type not even void also.

Inside a constructor if we want to do any initialization that type of logic we have to write it.

Local variable hide the visibility of instance variable when local variable or parameter variable and instance variable have same name.

This keyword : this is a keyword which refer to current object.

When local variable and instance variable have same name to refer to instance variable we have to use this.instancevariableName;

Parameterized constructor

Day 2 : 1 hour 1 to 2

Encapsulation : Binding or wrapping data/variable and methods/function in a single unit is known as Encapsulation.

Example : class.

class Employee {

int id

String name;

float salary;

void display() {

}

}

By default java follow encapsulation rules.

In Java if we want to achieve encapsulation all instance variable must be private and provide helper or setter method condition if condition satisfies then only set the value for instance variable or don’t set it.

Inheritance : Inheritance is use to inherits or acquire properties and behaviour of old class to new class.

class OldClass { // super class or base class or parent class.

properties and behaviour

}

class NewClass extends OldClass { // sub class or derived class or child class.

properties and behaviour

}

Java provided extends keyword to achieve inheritance.

With help of super class object we can access only its own properties and behaviour

With help of sub class object we can access its own as well as super class properties and behaviour.

Types of inheritance

1. Single inheritance : one super class and one sub class.

class A { }

class B extends A { }

1. Multilevel inheritance : one super class and n number of sub class connected one by one

class A { }

class B extends A{ }

class C extends B{ }

class D extends C { }

1. Hierarchical inheritance : oner super class and n number of sub class directly connected to super class

class A { }

class B extends A{ }

class C extends A{ }

class D extends A{ }

1. Multiple inheritance : more than one super class and one sub class :

class A { }

class B { }

class C extends A ,B { } But java doesn’t support this type of inheritance. In java one class can extends only one class it can’t extends more than one class. In java we can achieve multiple inheritance using interface .

Oops relationship :

Is a relationship : inheritance

Has a relationship

class Employee{

}

class Manager extends Employee{

}

class ProjectManager extends Manager{

}

class Developer extends Employee{

}

class AssociateManager extends Manager {

}

In inheritance super class must be generic and sub class must be specific.

Day 3 : 1 hour 9.15 to 10.15 (24-02-2023)

void add(int a, int b) {

}

void main() {

add(10,20);

}

class Employee {

int id;

String name;

float salary;

void info() {}

}

Employee emp = new Employee();

emp.id

emp.name

emp.salary;

emp.display();

has relationship :

inside one class if we create another class object then we can say it is has a relationship.

3 types of has relationship

Association

class A {

B obj = new B(); 0 or 1 or many

}

class B {

A obj = new A(); 0 or 1 or many

}

Association : inside one class we have to create another class object to achieve association.

Manager is a Employee

Aggregation : it is a type of association but it known as week association.

class Employee {

Address padd = new Address();

Address ladd = new Address();

}

class Address {

city and state

}

Composition : it is a type of association. It is known strong association ie composition.

class Student {

StudentHistory sh = new StudentHistory();

}

class StudentHistory {

}