**Day 3: 17-10-2025: SDLC**

1. Non primitive data types or reference data types: it is use to store value as well as reference of another data types.

In Java reference data types mainly divided into 4 types.

* 1. array
  2. class : pre defined or user defined
  3. interface : pre defined or user defined
  4. enum : pre defined or user defined

array : array is known as reference data types which is use to store more than one value of same data types or homogeneous elements.

Syntax

Array declaration

datatype arrayName[];

int []abc;

int xyz[];

int [] mno[];

int[] mmm[];

array declaration with initialization

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

array value start with index position 0.

Creating memory for array

datatype arrayname[]=new datatype[size];

int []num1=new int[10];

num1[0]=100;

num1[1]=200;

we can take the value using Scanner class.

enhanced loop or for each loop

syntax

for(datatype variableName : arrayName) {

S.O.P(variableName);

}

2-D array

int num[][]=new int[row][cols];

Jagged Array : In Jagged array of array can be different types.

Syntax

datatype arrayname[][]=new datatype[row\_size][];

OOPs using Java

Object Oriented Programming system

object : any real world entity

property or state -🡪 have -🡪 variables / fields

Person

Behaviour -🡪 do/does -🡪 methods / functions

Bank

Car

Employee

Customer

Product

class : blue print of object or template of object.

this keyword

this keyword is use to refer current object.

1. When local variable and instance variable have same name local variable hide the visibility of instance variable.

**this.instancevariabname=localvariableName**

types of variable or fields

1. instance variable:
   1. the variable which declared inside a class but outside a method including main method is known as instance variable.
   2. instance variable hold default value according to their data types. Int family –0, float family 0.0, char🡪space, Boolean 🡪 false, string 🡪null.
   3. Instance variable we can access within that class method but method must be non static.
2. local variable
   1. the variable which declared inside a method including main method is known as local variable.
   2. local variable doesn’t hold default value we need to initialized mandatory while using that variable.
   3. scope within that method where it declared.
3. static variable

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

1. Constructor have same name as class itself.
2. Constructor no return type not even void also. If we write void it consider as method not constructor.
3. Constructor no need to call it will call automatically whenever we create the object.
4. By default jvm provide default constructor and default constructor always empty constructor.
5. If we write explicitly empty or parameter constructor jvm doesn’t provide default constructor.
6. Constructor mainly use to do initialization.

In the life of the object if you want to perform any task only one time that type of task you need to write inside a constructor. If you want to perform more than one time that type of task you need to write inside a method. empty constructor always common task for all object. parameterized constructor dynamic value with specific task.

Copy constructor : You copy one object property into another object’s property. But Java doesn’t support this concept directly. We need to create copy constructor concept manually.

**Encapsulation:** Binding or wrapping data (variables) and code(methods) in a single unit is known as encapsulation.

Example : class.

We can’t access any property and behaviour of that class without creating the object.

**Java Bean class :**

Java Bean is a normal class which follow few rules while creating. It is also known as container class.

1. Class must be public.
2. All variable must be private
3. For each variable we need to provide setter and getter method.
4. Setter method is use to set the value. The method name start with prefix set followed by variable name.
5. Getter method is use to get the value. The method name start with prefix get followed by variable name.

Java Bean class also known as pure encapsulation class.

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

class OldClass { super class or base class or parent class

Property

Behaviour

}

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

Property

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 classes connected one by one.

class A {}

class B extends A {}

class C extends B{}

class D extends C{}

1. Hierarchical inheritance

One super class and n number of sub classes connected directly 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 {} we can’t extends more than one class at time. Java doesn’t support multiple inheritance. This type of inheritance support using interface.

**OOPs relationship**

1. Manager/Developer Is a Employee
2. Has a

class Employee{

id,name,salary // primitive property

readEmp()

disEmp();

}

class Manager extends Employee{

numberOfEmp

Address add = new Address(); // complex property

readMgr()

disMgr()

}

class ProjectManager extends Manager{

projectName;

readPmgr()

disPmgr();

}

class Developer extends Employee {

technology;

readDev()

disDev()

}

class Address{

city

state

Scanner sc = new Scanner

readAdd(){}

disAdd() {}

}

**Has a relationship**

1. Association
2. Aggregation
3. Composition

class A {

B obj = new B(); zero, 1 or many

}

class B {

A obj = new A(); zero, 1 or many

}

Association means either any one of the class you need to create class object ie 1 or many.

class Manager {

Address add = new Address(); 1 or many

}

class Address {

Manager mgr = new Manager(); not good.

}

Manager has a Address one or many.

It is a type of association but weak association. Weak association is known as aggregate.

class Students {

StudentHistory sh = new StudentHistory();

}

class StudentHistory {

}

It is a type of association. It is known as strong association because both class dependents upon each others. Strong association is known as composition.