**JFSD: A-Z of Back-end and Database Development**

**Day 2 : 14 Jul. 24**

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

Pts

1. Constructor have same name as class in self in java (But in JavaScript function name must be constructor).
2. Constructor no need to call it will call automatically whenever we create the object.
3. Constructor no return type not even void.
4. If we doesn’t write any constructor by default empty constructor provided by java compiler.

Constructor is use to do some initialization whenever memory get created…

Parameterized method and Parameterized constructor

this is keyword which is use to refer to current object.

one of the use of this

1. When instance variable and local variable have same name then using this we can refer to instance variable.

Constructor vs Methods

In the life of the object if we want to perform any task only one time that type of task we need to write inside empty or parameterized constructor.

Empty for all object same logic or same data set. Parameter for different object different value or different logic.

In the life of the object if we want to perform any task more than one time that type of task we need to write inside a method.

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

Example : class itself by default follow Encapsulation rules.

If we want our variable as secure use private keyword. when variable is private we can’t assign the value through object from outside class.

JavaBean class

1. class must be public
2. All variable part of that class must be private
3. For each variable we need to provide setter and getter methods.
4. Setter method is use to set the value
5. Getter method is use to get the value.

public class Customer {

private int cid;

private String cname;

public void setCid(int cid) {

this.cid = cid;

}

public int getCid() {

return this.cid;

}

}

In Java JavaBean class is known as pure encapsulation class.

Inheritance :

Inheritance is use to inherit the properties and behaviour of old class to new class.

Syntax

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

property

behaviour

}

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

property

behaviour

}

With help of sub class object we can access its own property and behaviour as well as super class property and behaviour. Using super class we can access only its own property 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 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 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{ } Java doesn’t support this type of inheritance. Using interface we can achieve multiple inheritance.

OOPs relationship

Is a relationship

Manager is a Employee

Developer is a Employee

ProjectMananger is a Manager

Has a relationship

Inside one class we are creating the object of another class is known as has a relationship.

Employee has a Address.

class Employee {

id,name,salary,Scanner object primitive property

readEmp

disEmp

}

class Manager extends Employee{

numberOfEmp

Address add = new Address(); complex property

readMgr() read,numberofemp

disMgr() : disp numberofemp

}

class ProjectManager extends Manager{

clientName

readPmgr()

disPmgr();

}

class Developer extends Employee{

projectName

readDev

disDev

}

class Address {

city and state , Scanner

readAdd()

disAdd()

}