Module no : 11 — Fr Classes and Objects — Date : 23-10-2023.

Introduction to oops :-

Agenda :-

1. What is OOPS?
2. Classes
3. Objects
4. Constructors
5. this keyword
6. Real life example.

Classes & Objects :!

What is OOPS?

Object Oriented Programming Style (OOPS) is a programming style that deals with classes & objects.

The main aim of Object-Oriented programming is to implement real-world entities, for example, object, classes, abstraction, inheritance, polymorphism etc..,

What is a class?

A class is a Blueprint for the object.

What is an Object?

An object is an entity that has a state and some behaviour.

Example:- Building a car: 1st thing we required is a blueprint (Structure) of a car. then how the car should perform (Speed, height, breadth, width). here Blueprint refers to class and performance refers to ~~objects~~. properties & functionalities

OOPS:

Program:

```
package com.company;

class Main{
    // boiler code main function
    public static void main(String[] args) {
        Car c1 = new Car(); // Creating an Object in Main Class
        Car c2 = new Car();

        // Accessing the class properties(Attributes)
        System.out.println("Color of the car1 is : "+c1.color);
        System.out.println("Model of the car1 is : "+c1.model);
        System.out.println("Price of the car2 is : "+c1.price);

        // Manipulating c2:
        c2.model = "Mahindra xuv 800";
        c2.color = "Red";
        c2.price = 5000000;

        System.out.println("Color of the car2 is : "+c2.color);
        System.out.println("Model of the car2 is : "+c2.model);
        System.out.println("Price of the car2 is : "+c2.price);

        // now a single class is used as 2 objects.
        // c1 and c2 both are different objects.
    }
}

// creating a new class
class Car{
    // Properties
    String model = "Mahindra xuv 500";
    String color = "Black";
    int price = 3800000;
    // Functionalities
    void gearType(){
```

```
        System.out.println("Manual/Automatic");
    }
    void lock(){
        System.out.println("Car is locked");
    }
    void unlock(){
        System.out.println("Car is unlocked");
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
Color of the car1 is : Black
Model of the car1 is : Mahindra xuv 500
Price of the car2 is : 3800000
Color of the car2 is : Red
Model of the car2 is : Mahindra xuv 800
Price of the car2 is : 5000000

Process finished with exit code 0
```

What is happening ?

Data Members	Class Methods
model	drive ()
colour	lock ()
price.	unlock

Class Car.

model : Mahindra XUV 500	model : Mahindra XUV 800
Colour : Black	Colour : Red
price : 3800000	price : 5000000
object C1	object C2

Classes and Objects : II :-

calling the functionalities of a class :-

objectname . function in the class ()

Eg :- C1. unlock();

OOPS – getter and setter:

Program:

```
package com.company;

class Main{
    // boiler code main function
    public static void main(String[] args) {
        Car c1 = new Car(); // Creating an Object in Main Class
        Car c2 = new Car();

        // Accessing the class properties(Attributes)
        System.out.println("Color of the car1 is : "+c1.color);
        System.out.println("Model of the car1 is : "+c1.model);
        System.out.println("Price of the car2 is : "+c1.getPrice());//get price

        // Manipulating c2:
        c2.model = "Mahindra xuv 800";
        c2.color = "Red";
        c2.setPrice(5000000);

        System.out.println("Color of the car2 is : "+c2.color);
        System.out.println("Model of the car2 is : "+c2.model);
        System.out.println("Price of the car2 is : "+c2.getPrice());//get price


        // now a single class is used as 2 objects.
        // c1 and c2 both are different objects.

        // calling the functionalities
        c1.lock();
        c2.unlock();

        System.out.println("Car 1 is locked..? "+c1.isLock);
        System.out.println("Car 2 is locked..? "+c2.isLock);
    }
}
```

```java
// creating a new class
class Car{
    // Properties
    String model = "Mahindra xuv 500";
    String color = "Black";
    int price = 3800000;
    boolean isLock = false;
    // Functionalities
    void gearType(){
        System.out.println("Manual/Automatic");
    }
    void lock(){
        isLock = true;
        System.out.println("Car is locked");
    }
    void unlock(){
        isLock = false;
        System.out.println("Car is unlocked");
    }
    // getter for price property
    int getPrice(){
        return price;
    }
    // setter for price property
    void setPrice(int priceValue){
        price = priceValue;
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
Color of the car1 is : Black
Model of the car1 is : Mahindra xuv 500
Price of the car2 is : 3800000
Color of the car2 is : Red
Model of the car2 is : Mahindra xuv 800
Price of the car2 is : 5000000
Car is locked
Car is unlocked
Car 1 is locked..? true
Car 2 is locked..? false

Process finished with exit code 0
```

Constructor :-

A java constructor is special method that is called when an object is instantiated. It has the same name as the class itself and is invoked automatically at the time of object Construction.

Constructor does not have any return type.

Eg :-

```
Cor(){
    System.out.println("Inside the constructor");
}
```

→ We can create Multiple Constructors.

OOPS – Constructor:

Program:

```
package com.company;

class Main{
    // boiler code main function
    public static void main(String[] args) {
        Car c1 = new Car("Mercedes Benz", "Grey", 10000000); // Creating an Object in Main Class
        Car c2 = new Car("Audi", "White", 20000000);
        Car c3 = new Car();
        System.out.println("Car1 : "+c1.color+","+c1.model+","+c1.price); // using constructor 1
        System.out.println("Car2 : "+c2.color+","+c2.model+","+c2.price);
        System.out.println("Car3 : "+c3.color+","+c3.model+","+c3.price); // using constructor 2
    }
}

// creating a new class
class Car{
    // Properties
    String model;
    String color;
    int price;
    boolean isLock = false;
    // Functionalities
    void gearType(){
        System.out.println("Manual/Automatic");
    }
    void lock(){
        isLock = true;
        System.out.println("Car is locked");
    }
    void unlock(){
        isLock = false;
        System.out.println("Car is unlocked");
    }
    // getter for price property
    int getPrice(){
```

```
        return price;
    }
    // setter for price property
    void setPrice(int priceValue){
        price =  priceValue;
    }
    // Constructor with arguments
    Car(String modelName, String colorName, int priceValue){
        model = modelName;
        color = colorName;
        setPrice(priceValue);
    }
    // creating another constructor with no arguments
    Car(){
        model = "Mahindra XUV 500";
        color = "Red";
        price = 3800000;
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
Car1 : Grey,Mercedes Benz,10000000
Car2 : White,Audi,20000000
Car3 : Red,Mahindra XUV 500,3800000

Process finished with exit code 0
```

Example: Complex Numbers :-

Task:- Write a class to store and perform operations on Complex numbers.

let: $x = 2 + 3i$ → imaginary number.

↑ real number

OOPS – Complex Number - Example:

Program:

```
package com.company;

class ComplexNumber{
    int realNumber;
    int imgNumber;

    // constructor
    ComplexNumber(int real, int img){
        realNumber = real;
        imgNumber = img;
    }

    void print(){
        System.out.println(realNumber+" + "+imgNumber+"i");
    }

    ComplexNumber add(ComplexNumber y){
        int sumReal = realNumber + y.realNumber;
        int sumImg = imgNumber + y.imgNumber;
        ComplexNumber res = new ComplexNumber(sumReal, sumImg);
        return res;
    }
}
class Main{
    public static void main(String[] args){
        ComplexNumber x = new ComplexNumber(2,3);
        x.print();

        ComplexNumber y = new ComplexNumber(5,4);
        y.print();

        ComplexNumber z = x.add(y);
        z.print();
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
2 + 3i
5 + 4i
7 + 7i

Process finished with exit code 0
```

this - Keyword :-

In Java, this keyword is used to the current object inside a class method.

Sample program :-

```
class ComplexNumber {
    int real;
    int imaginary;
    // Constructor
    ComplexNumber(int real, int imaginary) {
        this.real = real;
        this.imaginary = imaginary;
    }
}
```

OOPS – toString:

Program:

```
package com.company;

class ComplexNumber{
    int realNumber;
    int imgNumber;

    // constructor
    ComplexNumber(int realNumber, int imgNumber){
        this.realNumber = realNumber;
        this.imgNumber = imgNumber;
    }

    ComplexNumber add(ComplexNumber y){
        int sumReal = this.realNumber + y.realNumber;
        int sumImg = this.imgNumber + y.imgNumber;
        ComplexNumber res = new ComplexNumber(sumReal, sumImg);
        return res;
    }
    public String toString(){
        return this.realNumber+" + "+this.imgNumber+"i";
    }
}
class Main{
    public static void main(String[] args){
        ComplexNumber x = new ComplexNumber(2,3);
        System.out.println("x = "+x);

        ComplexNumber y = new ComplexNumber(5,4);
        System.out.println("y = "+y);
        ComplexNumber z = x.add(y);
        System.out.println("z = "+z);
        //System.out.println("x = "+x); // x = com.company.ComplexNumber@27973e9b is the output after using this
keyword
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
2 + 3i
5 + 4i
7 + 7i

Process finished with exit code 0
```