Lecture 33

Static & Instance Methods

Static Method :- Class-level methods

- belong to class rather than any specific instance of class
- defined with static keywords
- Since they belong to the class, they can be called directly using the class name without creating an object
- A static method can access only other static data (variables) and static methods. It can not access instance variables or instance methods directly (without an object reference).

e.g.:-
```
public class Demo {
    String name = "Jenny";
    public static void greet() {
        System.out.println("Hi" + name);
    }
}
```

Here name is instance variable

gives error

⇒ But if we make the name variable static then we can access it directly. or if we create object of the class to access name then also it will work fine.

e.g.:-
```
public class Demo {
    String name = "Jenny";
    public static void greet() {
        Demo obj = new Demo();
        System.out.println("Hi" + obj.name);
    }
}
```

⇒ it will work fine

Usage:- Static methods are used for operations that do not depend on instance variables of the class. They perform tasks that are general to the class itself rather than to any individual instance. A common example is utility methods

e.g:-

```
static int square (int num) {
    return num * num;
}
static int add (int a, int b) {
    return a+b;
}
```

→ Suppose we have a class for simple string operations like reversing a string or converting a string to uppercase. These operations don't depend on any instance data and can be made static.

e.g:-

```
public class StringUtils {
    static String reverse (String str) {
        return new StringBuilder(str).reverse().toString();
    }
    static String toUpperCase (String str) {
        return str.toUpperCase();
    }
}
```

So static methods are useful for utility classes, shared resources & functions that are general to the entire class.

⇒ Let's understand use of static methods with real life analogies

e.g:- a shared calculator in an office

→ In Java this would be similar to having a Calculator class with static methods like Add, Subtract, Multiply & Divide.

```
public class Calculator {
    static int add(int a, int b) {
        return a+b;
    }
    static int subtract(int a, int b) {
        return a-b;
    }
}
```

These methods don't depend on any particular calculator. These are general-purpose operations, so we can use them directly from the class without creating individual calculators.

⇒ **Second example is of Bank's Exchange Rate Service.**

Imagine a bank providing the current exchange rate between USD & EUR. The rate is universal for all customers. So there is no need to create separate service for each customer requesting the rate.

Instance Methods – Instance methods belongs to specific instance/object of the class.

- These are object-level methods
- They can access & modify instance variables
- To call an instance method, we need to create object of the class
- They can access both instance & static variables and can call both static & instance methods.

→ Instance methods are used when we need to work with the data stored in a specific object.

Invocation :- objectreference.methodName()

eg:-

```
public class Demo{
    String empName;
    int empId;
    void setInfo(String name, int id){
        empName = name;
        empId = id;
    }
    void getInfo(){
        System.out.println("Hi " +empName);
        System.out.println("Your empId is: " +empId);
    }
public static void main(String[] args){
    Demo obj = new Demo();
    obj.setInfo("Jenny", 101);
    obj.getInfo();
    obj.setInfo("Payal", 102);
    obj.getInfo();
    }
}
```

these attributes are representing the state of the object

* So here as you can see the state of the object has been updated. Initially state was Null (No name & no empid) then we updated it to first set of values ("Jenny", 101) & then another set of values ("Payal", 102)

And we can also update a single value.

e.g:-

```
void updateDepartment(String dept){
void updateName(String name){
    empName = name;
}
```

→ To call this we can write:-

```
obj.updateName("Jiya");
obj.getInfo();
```

A class can contain all the static methods, all the instance methods or a mix of both static and instance methods