Dynamic Method Dispatch in Java

- Dynamic method dispatch is also known as run time polymorphism.
- It is the process through which a call to an overridden method is resolved at runtime.
- This technique is used to resolve a call to an overridden method at runtime rather than compile time.
- To properly understand Dynamic method dispatch in Java, it is important to understand the concept of upcasting because dynamic method dispatch is based on upcasting.

Upcasting :

- It is a technique in which a superclass reference variable refers to the object of the subclass.

Example :

```
class Animal{}
class Dog extends Animal{}
```

Copy

```
Animal a=new Dog();//upcasting
```

Copy

In the above example, we've created two classes, named Animal(superclass) & Dog(subclass). While creating the object 'a', we've taken the reference variable of the parent class(Animal), and the object created is of child class(Dog).

Example to demonstrate the use of Dynamic method dispatch :

- In the below code, we've created two classes: **Phone & SmartPhone**.
- The **Phone** is the parent class and the **SmartPhone** is the child class.
- The method **on()** of the parent class is overridden inside the child class.
- Inside the main() method, we've created an object **obj** of the **Smartphone()** class by taking the reference of the **Phone()** class.
- When **obj.on()** will be executed, it will call the **on()** method of the **SmartPhone()** class because the reference variable obj is pointing towards the object of class **SmartPhone()**.

```
class Phone{
    public void showTime(){
```

```
        System.out.println("Time is 8 am");
    }
    public void on(){
        System.out.println("Turning on Phone...");
    }
}

class SmartPhone extends Phone{
    public void music(){
        System.out.println("Playing music...");
    }
    public void on(){
        System.out.println("Turning on SmartPhone...");
    }
}
public class CWH {
    public static void main(String[] args) {

        Phone obj = new SmartPhone(); // Yes it is allowed
        // SmartPhone obj2 = new Phone(); // Not allowed

        obj.showTime();
        obj.on();
        // obj.music(); Not Allowed


    }
}
```

Copy

Output :

```
Time is 8 am
Turning on SmartPhone...
```

Copy

Note: The data members can not achieve the run time polymorphism.

Code as described/written in the video :

```
package com.company;
class Phone{
    public void showTime(){
        System.out.println("Time is 8 am");
    }
    public void on(){
        System.out.println("Turning on Phone...");
    }
}

class SmartPhone extends Phone{
    public void music(){
        System.out.println("Playing music...");
    }
    public void on(){
        System.out.println("Turning on SmartPhone...");
    }
}
public class cwh_49_dynamic_method_dispatch {
    public static void main(String[] args) {
        // Phone obj = new Phone(); // Allowed
        // SmartPhone smobj = new SmartPhone(); // Allowed
        // obj.name();
```

```
        Phone obj = new SmartPhone(); // Yes it is allowed
        // SmartPhone obj2 = new Phone(); // Not allowed

        obj.showTime();
        obj.on();
        // obj.music(); Not Allowed


    }
}
```