Dynamic Method Dispatch in Java

1. Dynamic method dispatch is also known as run time polymorphism.

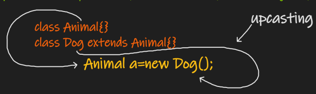
2. It is the process through which a call to an overridden method is resolved at runtime.

3. This technique is used to resolve a call to an overridden method at runtime rather than compile time.

4. 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 :

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



```
class Animal{}
class Dog extends Animal{}
Animal a=new Dog();
```

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 :

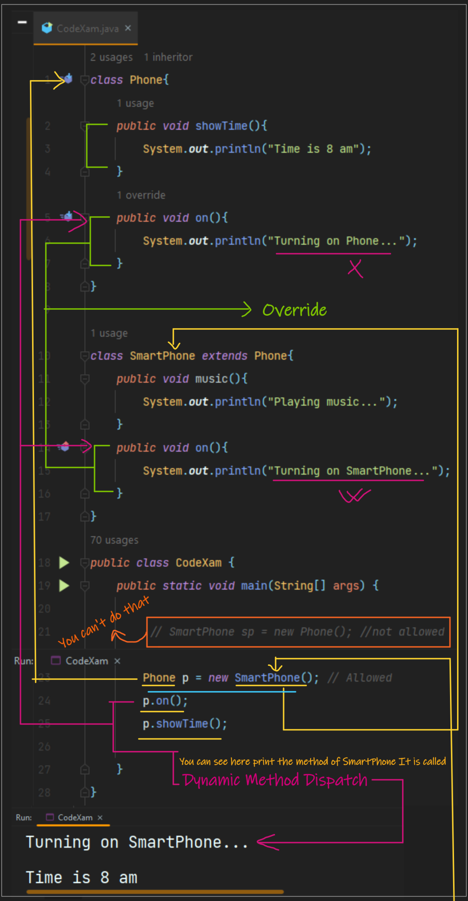
-> In the below code, we've created two classes: Phone & SmartPhone.

-> 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().



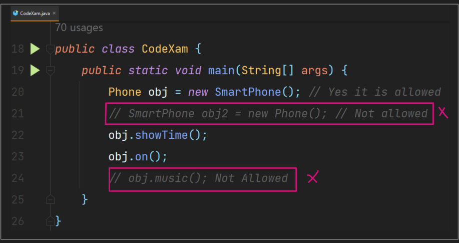
```
class Phone{
    public void showTime(){
        System.out.println("Time is 8 am");
    }
    public void on(){
        System.out.println("Turning on Phone...");
    }
}
class SmartPhone extends Phone{
    public void music(){
        System.out.println("Playing music...");
    }
    public void on(){
        System.out.println("Turning on SmartPhone...");
    }
}
public class CodeXam {
    public static void main(String[] args) {

        // SmartPhone sp = new Phone(); //not allowed

        Phone p = new SmartPhone(); // Allowed
        p.on();
        p.showTime();

    }
}
```

You can see here print the method of SmartPhone It is called Dynamic Method Dispatch

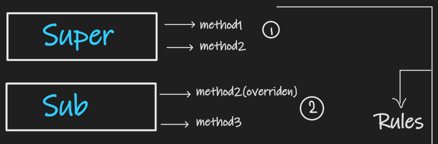
```
Turning on SmartPhone...
Time is 8 am
```

Dynamic method dispatch is also known as run time polymorphism because its called because it depend on run time not compile time that which method will be called

The data members can not achieve the run time polymorphism.



```
public class CodeXam {
    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
    }
}
```



Scenario1 -> Super obj = new Sub() ⟶ Allowed ✓

obj.method2() ⟶ is Called(method of object)

obj.method3() ⟶ Not Allowed X

Scenario2 -> Sub obj = new Super() ⟶ Not Allowed X