Day-9

* Polymorphism
* **What is Polymorphism in Java?**

**Polymorphism** in Java occurs when there are one or more classes or objects related to each other by inheritance. It is the ability of an object to take many forms. Inheritance lets users inherit attributes and methods, and polymorphism uses these methods to perform different tasks. So, the goal is communication, but the approach is different.

For example, you have a smartphone for communication. The communication mode you choose could be anything. It can be a call, a text message, a picture message, mail, etc. So, the goal is common that is communication, but their approach is different. This is called **Polymorphism.** Now, we will learn Polymorphism in Java with example.

**Java Polymorphism in OOPs with Example**

We have one parent class, ‘Account’ with function of deposit and withdraw. Account has 2 child classes

The operation of deposit and withdraw is same for Saving and Checking accounts. So the inherited methods from Account class will work.



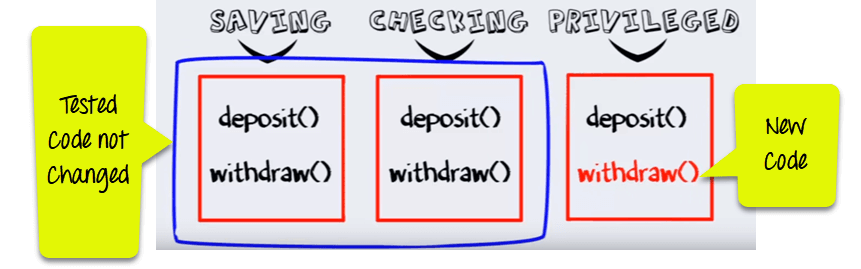
Java Polymorphism Example

### Change in Software Requirement

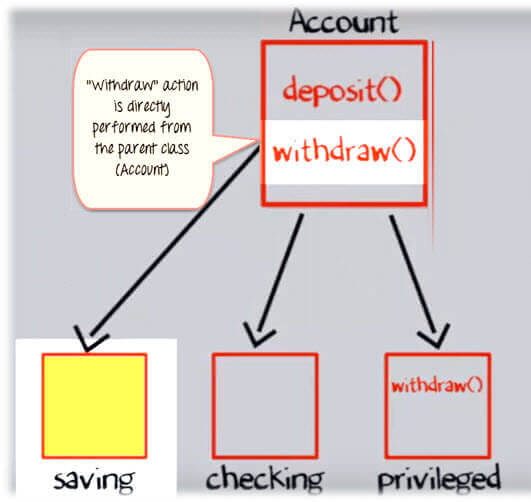
There is a change in the requirement specification, something that is so common in the software industry. You are supposed to add functionality privileged Banking Account with Overdraft Facility.

For a background, overdraft is a facility where you can withdraw an amount more than available the balance in your account.

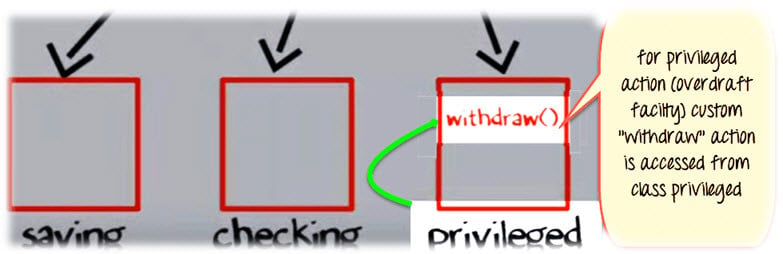
So, withdraw method for privileged needs to implemented afresh. But you do not change the tested piece of code in Savings and Checking account. This is advantage of OOPS



**Step 1)** Such that when the “withdrawn” method for saving account is called a method from parent account class is executed



**Step 2)**But when the “Withdraw” method for the privileged account (overdraft facility) is called withdraw method defined in the privileged class is executed. This is **Polymorphism in OOPs.**



## Method Overriding in Java

Method Overriding is redefining a super class method in a sub class.

### Rules for Method Overriding

* The method signature i.e. method name, parameter list and return type have to match exactly.
* The overridden method can widen the accessibility but not narrow it, i.e. if it is private in the base class, the child class can make it public but not vice versa.

**Difference between Overloading and Overriding**

| **Method Overloading** | **Method Overriding** |
| --- | --- |
| Method overloading is in the same class, where more than one method have the same name but different signatures. | Method overriding is when one of the methods in the super class is redefined in the sub-class. In this case, the signature of the method remains the same. |
| **Ex:**  void sum (int a , int b);  void sum (int a , int b, int c);  void sum (float a, double b); | **Ex:**  class X{  public int sum(){  // some code  }  }  class Y extends X{  public int sum(){  //overridden method  //signature is same  }  } |

* **Encapsulation**

## What is Encapsulation in Java?

**Encapsulation in Java** is a mechanism to wrap up variables(data) and methods(code) together as a single unit. It is the process of hiding information details and protecting data and behaviour of the object. It is one of the four important OOP concepts. The encapsulate class is easy to test, so it is also better for unit testing.

**Learn Encapsulation with an Example**

To understand what is encapsulation in detail consider the following bank account class with deposit and show balance methods

class Account {

private int account\_number;

private int account\_balance;

public void show Data() {

//code to show data

}

public void deposit(int a) {

if (a < 0) {

//show error

} else

account\_balance = account\_balance + a;

}

}