MOD -02 -> OOPs

Polymorphism

Polymorphism

Polymorphism - Method Overriding

If a subclass provides a method with the same signature (name and parameter) as in its super class, then subclass can override the method of its super class. This process of overriding a super class method by subclass is known as method overriding.

Conditions for method overriding

1. Method in subclass must have same signature (i.e. method name, parameter list and return type have to match exactly) as in its super class.
2. 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.

```java
package polypack;


//Superclass
class Doctor {
    public void treatPatient() {
        System.out.println("Doctor gives general treatment.");
    }
}


//Subclass
class Surgeon extends Doctor {
    @Override
    public void treatPatient() {
        System.out.println("Surgeon performs surgery.");
    }
}
```

```java
public class MethodOverrideDemo {

    public static void main(String[] args) {
        Doctor generalDoctor = new Doctor();
        Surgeon heartSurgeon = new Surgeon();


        System.out.println("=== General Doctor ===");
        generalDoctor.treatPatient();


        System.out.println("\n=== Surgeon ===");
        heartSurgeon.treatPatient();

    }

}
```

Can static method be overridden?

No,

Why?

In Java, static methods belong to the class, not to instances (objects) of the class. Therefore:

- Static methods are resolved at compile time (early binding).
- Overriding relies on runtime polymorphism, which works only with instance methods.
- If you declare a static method with the same signature in a subclass, it is called method hiding, not overriding.

Example for Clarification

```
package polypack;
class Parent {
    static void show() {
        System.out.println("Parent's static show()");
    }
}


class Child extends Parent {
//@Override ->The method show() of type Child must override or implement a supertype
method
    static void show() {
        System.out.println("Child's static show()");
    }
}
```

```java
public class StaticMethodOverridng {
    public static void main(String[] args) {
        Parent.show(); // Output: Parent's static show()
        Child.show(); //Output : Child's static show()
        Parent p=new Child();
        p.show(); // Output: Parent's static show()
    }
}
```

Role of access modifiers in method overriding

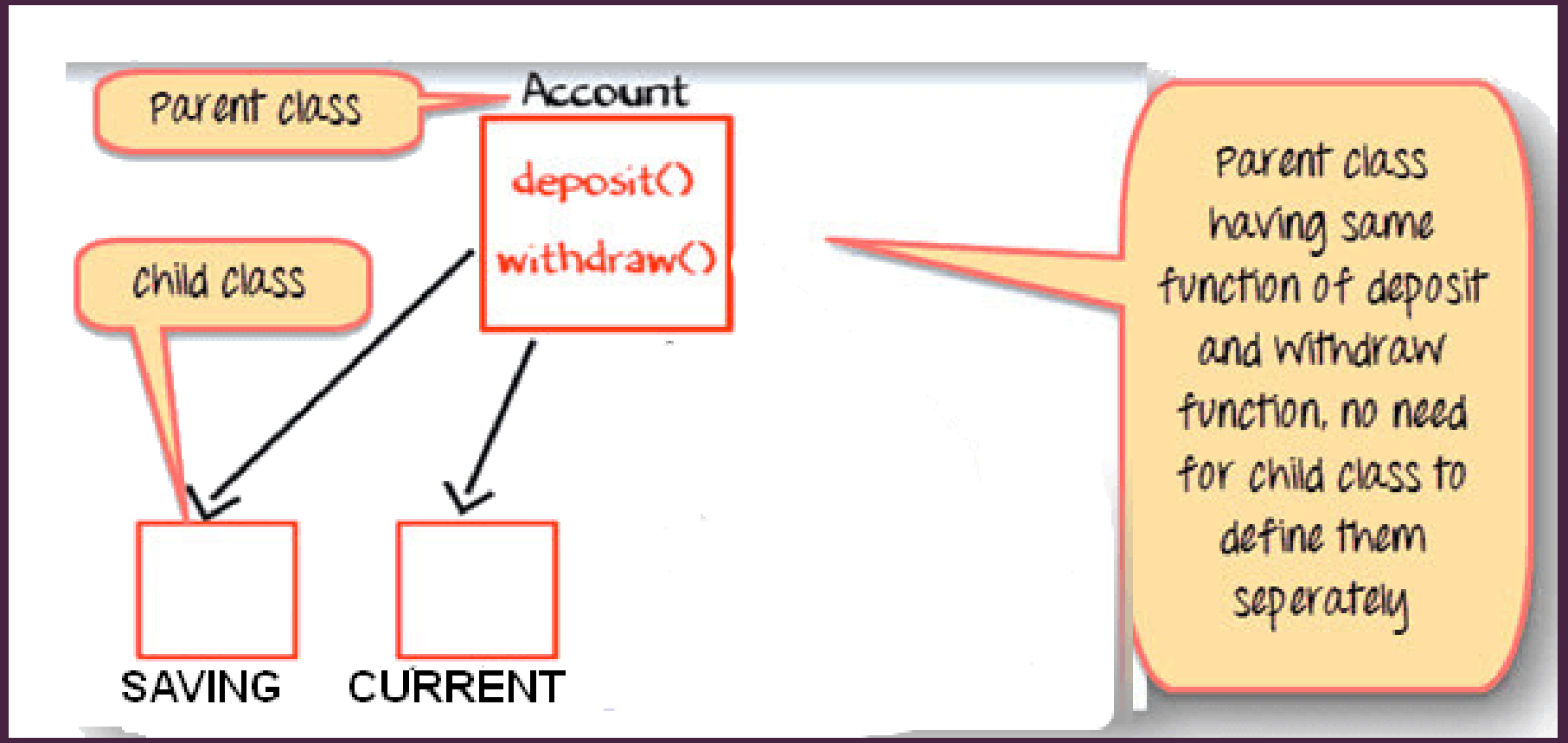
Access modifier of overridden method in subclass can’t be more restrictive than in super class.

Otherwise it will throw an exception.

Java Polymorphism in OOP's with Example

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

The operation of saving and withdraw is same for SAVING and CURRENT accounts. So the inherited methods from Account class will work

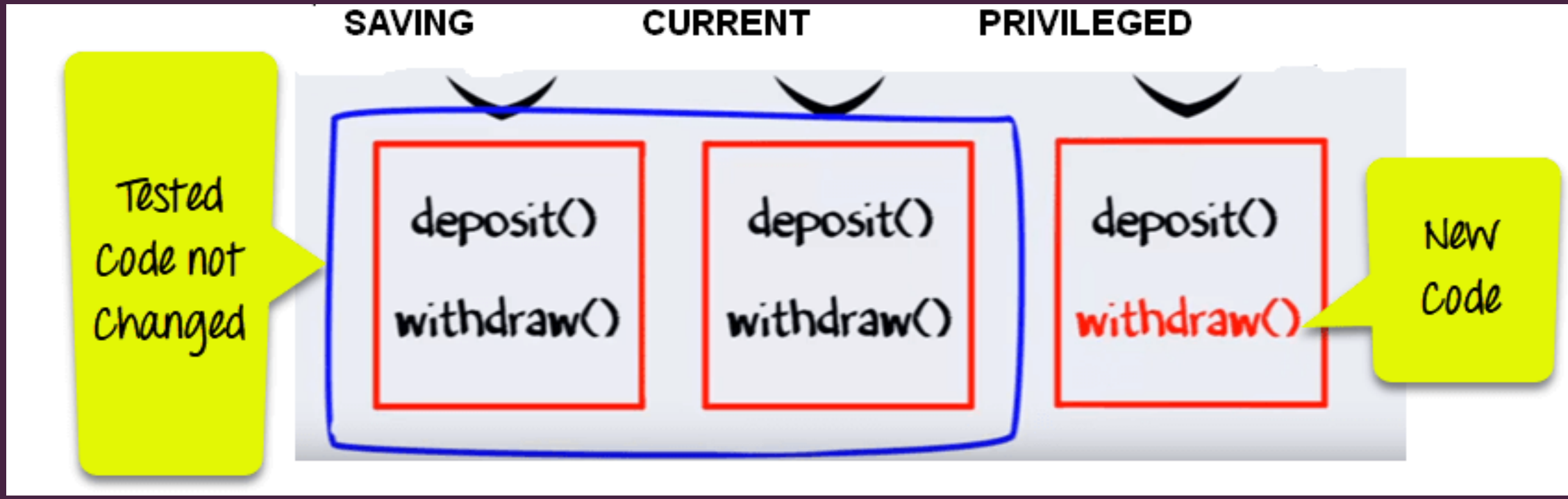

parent class
Account
deposit()
withdraw()
child class
SAVING
CURRENT
Parent class having same function of deposit and withdraw function, no need for child class to define them seperately

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.

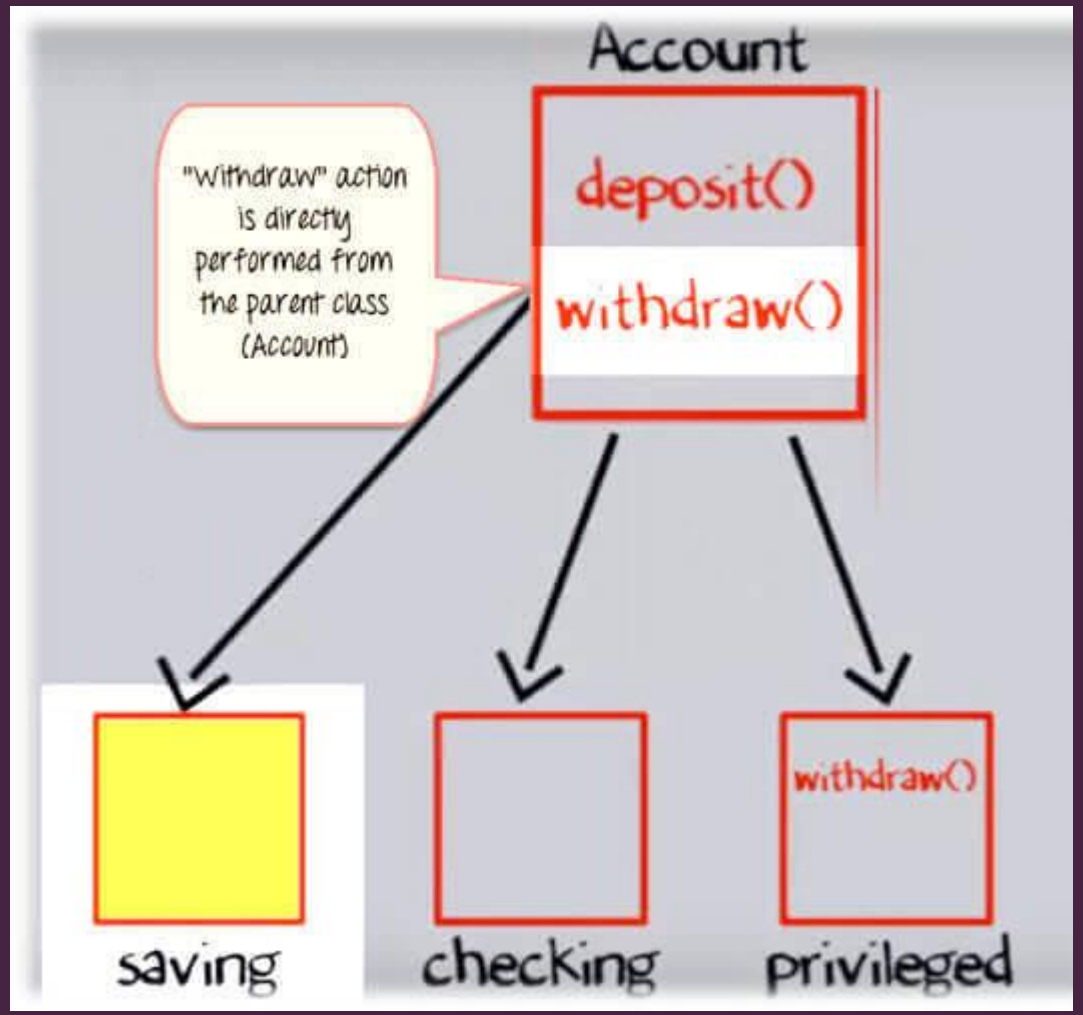
- For a background, overdraft is a facility where you can withdraw an amount more than available the balance in your account.

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


SAVING
CURRENT
PRIVILEGED
Tested Code not Changed
deposit()
withdraw()
deposit()
withdraw()
deposit()
withdraw()
New Code

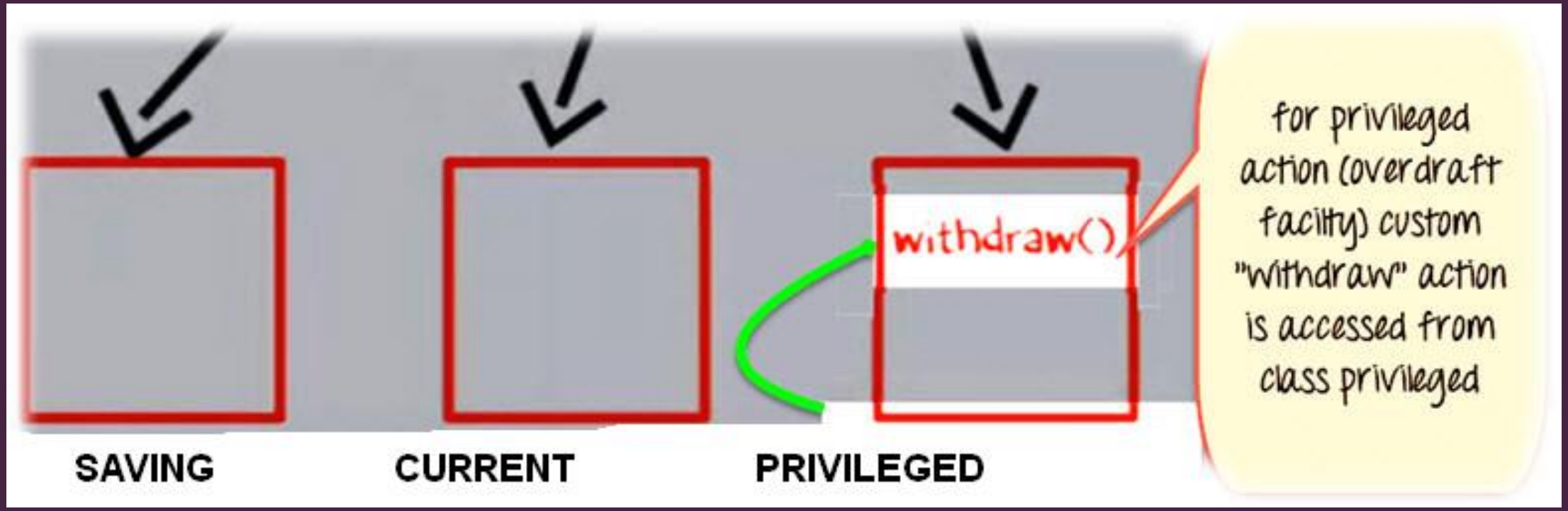
Step 1)

Such that when the "withdrawn" method for saving account is called a method from parent account class is executed.

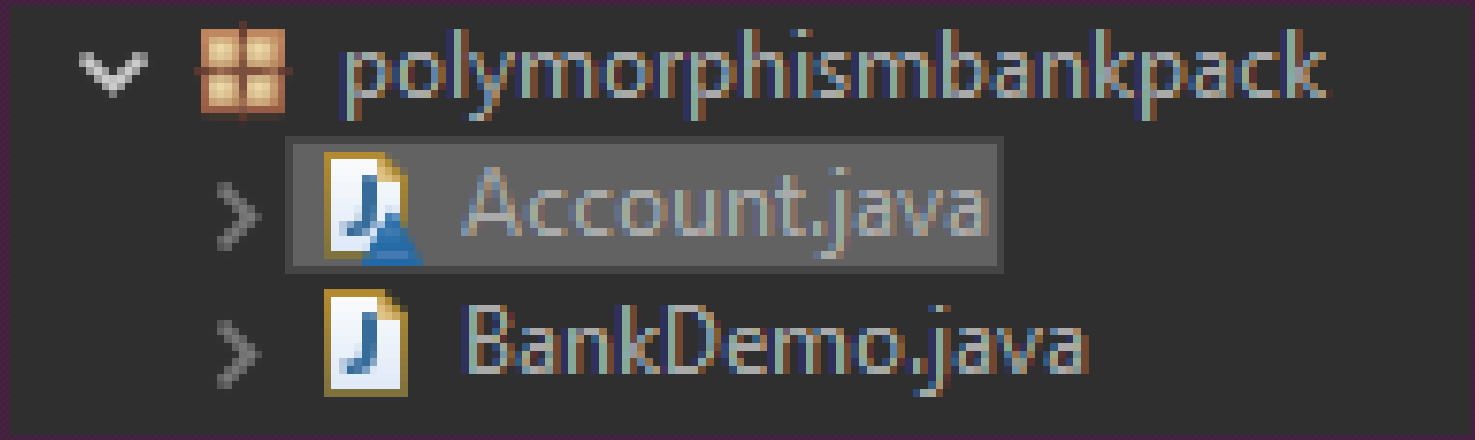


Step 2)

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.



Example for Clarification



```java
package polymorphismbankpack;
//Parent Class
class Account {
    protected double balance;

    public Account(double balance) {
        this.balance = balance;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount + ", Balance: " + balance);
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ", Balance: " + balance);
        } else {
            System.out.println("Insufficient balance!");
        }
    }
}
```

```
//Child Class: SavingsAccount
class SavingsAccount extends Account {
    public SavingsAccount(double balance) {
        super(balance);
    }
}


//Child Class: CurrentAccount
class CurrentAccount extends Account {
    public CurrentAccount(double balance) {
        super(balance);
    }
}
```

```java
//Child Class: PrivilegedAccount with Overdraft
class PrivilegedAccount extends Account {
    private double overdraftLimit = 5000;

    public PrivilegedAccount(double balance) {
        super(balance);
    }

    @Override
    public void withdraw(double amount) {
        if (balance + overdraftLimit >= amount) {
            balance -= amount;
            System.out.println("Privileged Withdrawn: " + amount + ", Balance: " + balance);
        } else {
            System.out.println("Overdraft limit exceeded!");
        }
    }
}
```

```java
package polymorphismbankpack;

public class BankDemo {
    public static void main(String[] args) {
        Account acc1 = new SavingsAccount(2000);
        Account acc2 = new CurrentAccount(3000);
        Account acc3 = new PrivilegedAccount(1000);

        System.out.println("\n=== Saving Account ===");
        acc1.withdraw(1500); // Uses parent method


        System.out.println("\n=== Current Account ===");
        acc2.withdraw(2500); // Uses parent method


        System.out.println("\n=== Privileged Account ===");
        acc3.withdraw(5500); // Uses overridden method in PrivilegedAccount
    }
}
```

Thank you ☺ Happy learning ☺