Java module 3

Exercises Day 3 (A)

1 - Encapsulation	Basic Bank Account Class

Instructions

Create a simple BankAccount class to handle deposit and withdrawal operations, ensuring that the account balance cannot directly be altered from outside the class.

Add the following main method to test your BankAccount class:

```java
public static void main(String[] args) {
   BankAccount account = new BankAccount(200);
   account.deposit(150);
   System.out.println(account); // Should show updated
balance
   account.withdraw(100);
   System.out.println(account); // Should show updated
balance after withdrawal
}
```

Expected output

```
Account Balance: $350.00
Account Balance: $250.00
```

Solution

```java
public class BankAccount {

    private double balance;

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

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
        }
    }

    @Override
    public String toString() {
        return "Account Balance: $" + String.format("%.2f",
balance);
    }
}
```

2 -	Extending BankAccount with SavingsAccount

Instructions

Extend your previous BankAccount class to create a new class called SavingsAccount.
The SavingsAccount class should have a new feature: interest accumulation. When creating a new SavingsAccount, we should provide the account's initial balance and the interest rate that will be applied to the account. The SavingsAccount class should also offer a method to apply the interest, this method will calculate the interest and will add it to the account's current balance.

Add the following main method to test your SavingsAccount class:

```java
public static void main(String[] args) {
   SavingsAccount savingsAccount = new
SavingsAccount(1000, 5); // 5% interest rate
   System.out.println(savingsAccount); // Initial state
   savingsAccount.applyInterest(); // Apply interest
   System.out.println(savingsAccount); // After interest
is applied
   savingsAccount.withdraw(200);
   System.out.println(savingsAccount); // After the
withdraw

}
```

Expected output

```
Savings Account Balance: $1000.00, Interest Rate: 5.00%
Savings Account Balance: $1050.00, Interest Rate: 5.00%
Savings Account Balance: $850.00, Interest Rate: 5.00%
```

Solution
BankAccount.java

```java
public class BankAccount {

    private double balance;

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

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
        }
    }

    @Override
    public String toString() {
        return "Account Balance: $" + String.format("%.2f",
balance);
    }
}
```

Solution SavingsAccount.java

```java
public class SavingsAccount extends BankAccount {
    private double interestRate;

    // Constructor
    public SavingsAccount(double initialBalance, double
interestRate) {
        super(initialBalance); // Initialize balance
through the superclass constructor
        this.interestRate = interestRate;
    }

    // Method to apply interest
    public void applyInterest() {
        double interest = getBalance() * interestRate /
100;
        deposit(interest); // Reuse the deposit method to
add interest to the balance
    }

    // Override toString method
    @Override
    public String toString() {
        return "Savings Account Balance: $" +
String.format("%.2f", getBalance()) +
               ", Interest Rate: " + String.format("%.2f",
interestRate) + "%";
    }
}
```