Solution Review: Implement an Account Class using Polymorphism

This review provides a detailed analysis to solve the 'Implement an Account Class using Polymorphism' challenge.

WE'LL COVER THE FOLLOWING

Solution

```
// Account Class
class Account {

  protected double balance; // protected variable

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

  // member functions
  public void Deposit(double amount){}
  public void Withdraw(double amount){}
  public void printBalance(){}

}

// Savings class extended from Account class
class Savings extends Account {

  double interestRate = 0.8; // member variable

  public Savings(int balance) { // parametrized contructor
    super(balance); // calling base class constructor
  }

  // Implementation of member functions
  public void Deposit(double amount) {
    balance += amount + (amount * interestRate);
  }

  public void Withdraw(double amount) {
    balance -= amount + (amount * interestRate);
  }
```

```
  public void printBalance() {
    System.out.println("Balance in your saving account: " + balance);
  }

}

// Current class extended from the Account class
class Current extends Account {

  public Current(int balance) { // Parametrized constructor
    super(balance); // calling base class constructor
  }

  // Implementation of public member functions
  public void Deposit(double amount) {
    balance += amount;
  }

  public void Withdraw(double amount) {
    balance -= amount;
  }

  public void printBalance() {
    System.out.println("Balance in your current account: " + balance);
  }

}

class Demo {

  public static void main(String args[]) {
    // creating savings account object
    Account SAccount = new Savings(50000);

    SAccount.Deposit(1000);
    SAccount.printBalance();

    SAccount.Withdraw(3000);
    SAccount.printBalance();

    System.out.println();

    // creating current account object
    Account CAccount = new Current(50000);
    CAccount.Deposit(1000);
    CAccount.printBalance();

    CAccount.Withdraw(3000);
    CAccount.printBalance();
  }

}
```

Explanation

- We have implemented the `Account` class which has the **balance** double variable, and three public methods **Deposit(double amount),**

  Withdraw(double amount) and **printBalance()**

- Implemented `Savings` and `Current` classes extended from the `Account` class through the `extend` keyword

- `Savings` class has private double **interestRate** variable and following methods:

  - `Withdraw(double amount)` deducts *amount* from the *balance* with *interestRate*

  - `Deposit(double amount)` adds *amount* in the *balance* with *interestRate*

  - `printBalance()` displays the balance in the *account*

- `Current` class has following methods:

  - `Withdraw(double amount)` deducts *amount* from *balance*
  - `Deposit(double amount)` adds *amount* in *balance*
  - `printBalance()` displays the balance in the *account*`

- Created *Savings* and *Current* object by calling parametrized constructors of the classes and printed their balance by calling their respective methods