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
  - Explanation

Solution

```java
// Account Class
class Account {

  protected double balance; //

  public Account(double balanc
    this.balance = balance;
  }

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

}

// Savings class extended from
class Savings extends Account

  double interestRate = 0.8; /

  public Savings(int balance)
    super(balance); // calling
  }

  // Implementation of member
  public void Deposit(double a
    balance += amount + (amoun
  }

  public void Withdraw(double
```

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