Challenge 3: Implement an Account Class using Polymorphism

In this challenge, we'll implement an Account class along with two derived classes, Savings and Current.

We'll cover the following

Problem Statement

Write a code that has:

- A **parent** *class* named `Account`.
    - Inside it *define*:
        - a protected double member `balance`
        - `public void Withdraw(double amount)`
        - `public void Deposit(double amount)`
        - `public printBalance()`
- Then, there are **two derived** *classes*
    - `Savings` class
        - has a *private* member `interestRate` set to **0.8**
        - `Withdraw(double amount)` deducts *amount* from *balance* with *interestRate*
            - `Deposit(double amount)` adds *amount* in *balance* with

interestRate

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

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

Input

- In the `Savings` class, `balance` is set to **50000** in the parametrized constructor
- In the `Current` class, `balance` is set to **50000** in the parametrized constructor

Output

Balance before withdrawing from the savings account Balance after withdrawing from the savings account

Balance before withdrawing from the current account Balance after withdrawing from the current account

Sample Input

```
// 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();
```

Sample Output

```
    Balance in your saving account: 51800.0
    Balance in your saving account: 46400.0

    Balance in your current account: 51000.0
    Balance in your current account: 48000.0
```

Coding Exercise

First, take a close look and design a step-by-step algorithm before jumping to the implementation. This problem is designed for your practice, so initially try to solve it on your own. If you get stuck, you can always refer to the solution provided in the solution review. Good Luck!

```
// Write classes code here

class demo {

  public static void main(String args[]) {
    // make instances of classes here
    // call their traits functions here

  }

}
```

The solution will be explained in the next lesson.