**EXERCISE - Account Balance**

**Description**:

Create a program that represents Account.

Program allows to deposit/withdraw/getBalance from Account.

Expected Input:

* A double value for deposit/withdraw operations (use Scanner class)

Expected Output:

* Printed on console the current balance of the account and after deposit/withdraw operations

**Instructions**:

1. Create a class called 'Account' with a private field 'balance'
2. In the constructor for the Account class, set the initial balance of the account to 0.
3. Create 'deposit' method that takes a double value as a parameter and update the balance of the account by adding the value to the balance.
4. Create 'withdraw' method that takes a double value as a parameter and update the balance of the account by subtracting the value from the balance, but **only**if the withdrawal amount is less than the current balance.
5. Create 'getBalance' method that returns the current balance of the account.
6. In the main method, create an instance of the Account class, print out the initial balance, make a deposit and withdrawal, and print out the updated balance.

Hints:

* Make sure to test your code with different input values to ensure it's working as expected.
* Remember that 'this' keyword refers to the current instance of an object.
* Keep in mind that in this example, the balance is private and can only be accessed by the deposit, withdraw and getBalance methods, which prevent unauthorized access or manipulation of the balance.
* Remember that encapsulation is an important principle of OOP that helps to hide the implementation details of a class and restrict access to the internal state of an object. Simpler way: Encapsulation is a security measure that allows hiding how a program works and controlling who and how can use it. It's like locking a piggy bank so that no one else has access to it except you.

Example Solution below:

--------------------------------------------------------------------------------------------------------

1. class Account {
2. private double balance;
4. public Account() {
5. this.balance = 0;
6. }
8. public double getBalance() {
9. return balance;
10. }
12. public void deposit(double amount) {
13. this.balance += amount;
14. }
16. public void withdraw(double amount) {
17. if (amount <= this.balance) {
18. this.balance -= amount;
19. } else {
20. System.out.println("You don't have enough funds to withdraw");
21. }
22. }
23. }
25. public class Main {
26. public static void main(String[] args) {
27. Scanner scanner = new Scanner(System.in);
29. Account account = new Account();
30. System.out.println("Initial balance: " + account.getBalance());
32. System.out.print("Enter the deposit amount: ");
33. double depositAmount = scanner.nextDouble();
35. account.deposit(depositAmount);
37. System.out.println("Balance after deposit: " + account.getBalance());
39. System.out.print("Enter the withdraw amount: ");
40. double withdrawAmount = scanner.nextDouble();
42. account.withdraw(withdrawAmount);
44. System.out.println("Balance after withdrawal: " + account.getBalance());
45. }
46. }