PROJECT

ON

OBJECT ORIENTED PROGRAMMING LANGUAGE

Project Outline:

This project explains the Java code for a simple account management system with two types of accounts: SavingsAccount and CurrentAccount. Both types inherit from a base abstract class called Account.

1. Overall Design

The system utilizes inheritance to achieve code reusability and polymorphism. The abstract class `Account` defines common properties and functionalities for all account types. Specific behaviors like interest calculation are implemented in concrete subclasses (`SavingsAccount` and `CurrentAccount`).

2. Classes

- **Account (Abstract Class):**

  Properties:

  - `accountNumber` (private String): Unique identifier for the account.
  - `accountHolder` (private String): Name of the account holder.
  - `balance` (protected double): Current balance of the account.

  Constructors:

  - `Account(String accountNumber, String accountHolder, double balance)`: Initializes the account with provided details.

  Methods:

  - `getAccountNumber()`: Returns the account number.
  - `getAccountHolder()`: Returns the account holder's name.
  - `getBalance()`: Returns the current account balance.
  - `deposit(double amount)`: Deposits the specified amount into the account (with validation for positive values).
  - `withdraw(double amount)`: Withdraws the specified amount from the account (with validation for positive amounts and sufficient funds).

        - `calculateInterest()`: Abstract method, needs to be implemented by subclasses to define how interest is calculated.
- **SavingsAccount:**
    - Inherits from `Account`.

    Properties:

        - `interestRate` (private double): Interest rate applicable to the account.

    Constructors:

        - `SavingsAccount(String accountNumber, String accountHolder, double balance, double interestRate)`: Initializes the savings account with additional interest rate information.

    Overridden Methods:

        - `calculateInterest()`: Calculates interest based on the current balance and interest rate, adds the interest to the balance, and prints a message with the new balance.
- **CurrentAccount:**

        - Inherits from `Account`.

    Properties:

        - `overdraftLimit` (private double): Maximum amount that can be withdrawn beyond the current balance.

    Constructors:

        - `CurrentAccount(String accountNumber, String accountHolder, double balance, double overdraftLimit)`: Initializes the current account with additional overdraft limit information.

    Overridden Methods:

        - `withdraw(double amount)`: Checks if the withdrawal amount can be covered by the balance and overdraft limit. If sufficient, withdraws the money and updates the balance. Prints an appropriate message otherwise.
        - `calculateInterest()`: Prints a message stating that no interest is earned on current accounts.

3. Main Class

- The `Main` class demonstrates how to create `SavingsAccount` and `CurrentAccount` objects, perform deposits, withdrawals, calculate interest, and display account balances.

Explanation of the Project:

This code utilizes comments within the Java source code to explain the functionality of classes, methods, and properties. However, for a more comprehensive project documentation process, consider including the following:

- **Class Diagrams:** Create UML diagrams to visually represent the classes, their relationships (inheritance), and methods.

- **Detailed Method Descriptions:** Expand on the comments within the code to provide a more detailed explanation of each method's purpose, parameters, return values, and any specific logic implemented.

- **Assumptions and Constraints:** Document any assumptions made during development and limitations of the current implementation.

- **Future Enhancements:** List potential improvements or additional features that could be added to the system in the future.

Code for Bank Account System

```
abstract class Account {
    private String accountNumber;
    private String accountHolder;
    protected double balance;

    public Account(String accountNumber, String accountHolder, double
balance) {
        this.accountNumber = accountNumber;
        this.accountHolder = accountHolder;
        this.balance = balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public String getAccountHolder() {
```

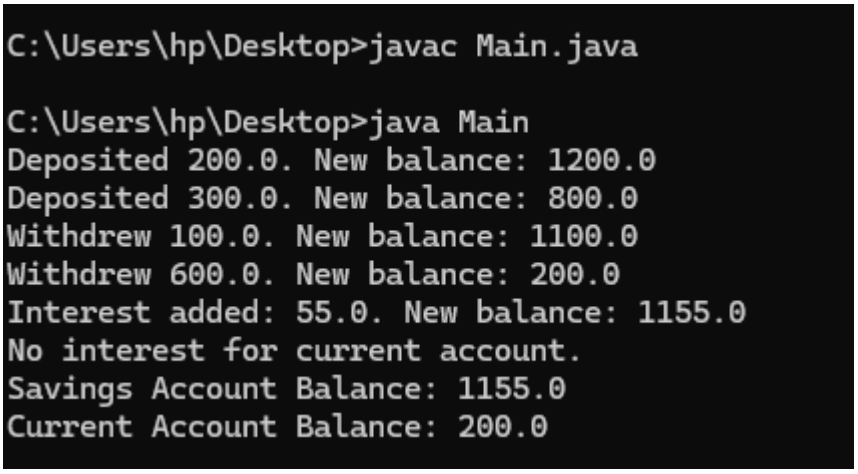
```
        return accountHolder;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited " + amount + ". New balance: " +
balance);
        } else {
            System.out.println("Deposit amount must be positive.");
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrew " + amount + ". New balance: " +
balance);
        } else {
            System.out.println("Insufficient funds or invalid amount.");
        }
```

```
    }

    public abstract void calculateInterest();
}

class SavingsAccount extends Account {
    private double interestRate;

    public SavingsAccount(String accountNumber, String accountHolder, double balance, double interestRate) {
        super(accountNumber, accountHolder, balance);
        this.interestRate = interestRate;
    }

    @Override
    public void calculateInterest() {
        double interest = balance * interestRate / 100;
        balance += interest;
        System.out.println("Interest added: " + interest + ". New balance: " + balance);
    }
}

class CurrentAccount extends Account {
    private double overdraftLimit;
```

```
    public CurrentAccount(String accountNumber, String accountHolder, double balance, double overdraftLimit) {
        super(accountNumber, accountHolder, balance);
        this.overdraftLimit = overdraftLimit;
    }

    @Override
    public void withdraw(double amount) {
        if (amount > 0 && (balance + overdraftLimit >= amount)) {
            balance -= amount;
            System.out.println("Withdrew " + amount + ". New balance: " + balance);
        } else {
            System.out.println("Overdraft limit exceeded or invalid amount.");
        }
    }

    @Override
    public void calculateInterest() {
        System.out.println("No interest for current account.");
    }
}

public class Main {
```

```
    public static void main(String[] args) {

        SavingsAccount savingsAccount = new SavingsAccount("SA123",
"charan", 1000.0, 5.0);

        CurrentAccount currentAccount = new CurrentAccount("CA123",
"keerthi", 500.0, 200.0);



        savingsAccount.deposit(200.0);

        currentAccount.deposit(300.0);



        savingsAccount.withdraw(100.0);

        currentAccount.withdraw(600.0);


        savingsAccount.calculateInterest();

        currentAccount.calculateInterest();



        System.out.println("Savings Account Balance: " +
savingsAccount.getBalance());

        System.out.println("Current Account Balance: " +
currentAccount.getBalance());

}


}
```

Output:


C:\Users\hp\Desktop>javac Main.java
C:\Users\hp\Desktop>java Main
Deposited 200.0. New balance: 1200.0
Deposited 300.0. New balance: 800.0
Withdrew 100.0. New balance: 1100.0
Withdrew 600.0. New balance: 200.0
Interest added: 55.0. New balance: 1155.0
No interest for current account.
Savings Account Balance: 1155.0
Current Account Balance: 200.0