**Miner Final Projects of Java With DS**

**2. Create a Bank Account Management System using OOP concepts.**

**Bank Account Management System:**

**1. Class Definition:** Create classes for Account, Customer, and

Bank.

**2. Attributes and Methods:** Include attributes and methods for

managing accounts, transactions, and customer information.

**3. Account Types:** Implement various account types using

inheritance if necessary.

**4. Transaction Handling:** Develop methods for deposits,

withdrawals, and account transfers.

**5. Customer Management:** Maintain customer records, including

personal information and linked accounts

PROGRAM IMPLEMENTATION

* Creating a complete Java code for a bank account management system is quite extensive, but I can provide you with a simplified version that demonstrates the basic structure using classes and methods.

**PROGRAM**

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

// Enum for account types

enum AccountType {

SAVINGS,

CURRENT

}

// Customer class to store personal information

class Customer {

private String customerId;

private String name;

private String address;

// Constructor

public Customer(String customerId, String name, String address) {

this.customerId = customerId;

this.name = name;

this.address = address;

}

// Getters and setters

public String getCustomerId() {

return customerId;

}

public String getName() {

return name;

}

public String getAddress() {

return address;

}

}

// Account class as a base class

class Account {

private String accountId;

private Customer customer;

private double balance;

// Constructor

public Account(String accountId, Customer customer, double balance) {

this.accountId = accountId;

this.customer = customer;

this.balance = balance;

}

// Getters and setters

public String getAccountId() {

return accountId;

}

public Customer getCustomer() {

return customer;

}

public double getBalance() {

return balance;

}

// Deposit method

public void deposit(double amount) {

balance += amount;

System.out.println("Deposit successful. New balance: " + balance);

}

// Withdrawal method

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawal successful. New balance: " + balance);

} else {

System.out.println("Insufficient funds");

}

}

// Transfer method

public void transfer(Account destinationAccount, double amount) {

if (amount <= balance) {

balance -= amount;

destinationAccount.deposit(amount);

System.out.println("Transfer successful. New balance: " + balance);

} else {

System.out.println("Insufficient funds for transfer");

}

}

}

// SavingsAccount class extending Account

class SavingsAccount extends Account {

private double interestRate;

// Constructor

public SavingsAccount(String accountId, Customer customer, double balance, double interestRate) {

super(accountId, customer, balance);

this.interestRate = interestRate;

}

// Getter for interestRate

public double getInterestRate() {

return interestRate;

}

}

// CurrentAccount class extending Account

class CurrentAccount extends Account {

private double overdraftLimit;

// Constructor

public CurrentAccount(String accountId, Customer customer, double balance, double overdraftLimit) {

super(accountId, customer, balance);

this.overdraftLimit = overdraftLimit;

}

// Getter for overdraftLimit

public double getOverdraftLimit() {

return overdraftLimit;

}

}

// Bank class to manage accounts and customers

class Bank {

private Map<String, Customer> customers;

private Map<String, Account> accounts;

// Constructor

public Bank() {

this.customers = new HashMap<>();

this.accounts = new HashMap<>();

}

// Add customer to the bank

public void addCustomer(Customer customer) {

customers.put(customer.getCustomerId(), customer);

}

// Create savings account

public void createSavingsAccount(String accountId, String customerId, double balance, double interestRate) {

Customer customer = customers.get(customerId);

if (customer != null) {

SavingsAccount savingsAccount = new SavingsAccount(accountId, customer, balance, interestRate);

accounts.put(accountId, savingsAccount);

System.out.println("Savings account created successfully");

} else {

System.out.println("Customer not found");

}

}

// Create current account

public void createCurrentAccount(String accountId, String customerId, double balance, double overdraftLimit) {

Customer customer = customers.get(customerId);

if (customer != null) {

CurrentAccount currentAccount = new CurrentAccount(accountId, customer, balance, overdraftLimit);

accounts.put(accountId, currentAccount);

System.out.println("Current account created successfully");

} else {

System.out.println("Customer not found");

}

}

// Get account balance

public double getAccountBalance(String accountId) {

Account account = accounts.get(accountId);

if (account != null) {

return account.getBalance();

} else {

System.out.println("Account not found");

return -1;

}

}

// Perform deposit

public void performDeposit(String accountId, double amount) {

Account account = accounts.get(accountId);

if (account != null) {

account.deposit(amount);

} else {

System.out.println("Account not found");

}

}

// Perform withdrawal

public void performWithdrawal(String accountId, double amount) {

Account account = accounts.get(accountId);

if (account != null) {

account.withdraw(amount);

} else {

System.out.println("Account not found");

}

}

// Perform transfer

public void performTransfer(String sourceAccountId, String destinationAccountId, double amount) {

Account sourceAccount = accounts.get(sourceAccountId);

Account destinationAccount = accounts.get(destinationAccountId);

if (sourceAccount != null && destinationAccount != null) {

sourceAccount.transfer(destinationAccount, amount);

} else {

System.out.println("One or more accounts not found");

}

}

}

// Example of using the Bank and related classes

public class BankApp {

public static void main(String[] args) {

Bank bank = new Bank();

Customer customer1 = new Customer("C1001", "John Doe", "123 Main St");

bank.addCustomer(customer1);

bank.createSavingsAccount("S10001", "C1001", 1000.0, 0.02);

bank.createCurrentAccount("C10001", "C1001", 500.0, 100.0);

System.out.println("Savings Account Balance: " + bank.getAccountBalance("S10001"));

System.out.println("Current Account Balance: " + bank.getAccountBalance("C10001"));

bank.performDeposit("S10001", 500.0);

bank.performWithdrawal("C10001", 200.0);

System.out.println("Savings Account Balance: " + bank.getAccountBalance("S10001"));

System.out.println("Current Account Balance: " + bank.getAccountBalance("C10001"));

bank.performTransfer("S10001", "C10001", 300.0);

System.out.println("Savings Account Balance: " + bank.getAccountBalance("S10001"));

System.out.println("Current Account Balance: " + bank.getAccountBalance("C10001"));

}

}

**OUTPUT**

Savings account created successfully

Current account created successfully

Savings Account Balance: 1000.0

Current Account Balance: 500.0

Deposit successful. New balance: 1500.0

Withdrawal successful. New balance: 300.0

Savings Account Balance: 1500.0

Current Account Balance: 300.0

Deposit successful. New balance: 600.0

Transfer successful. New balance: 1200.0

Savings Account Balance: 1200.0

Current Account Balance: 600.0

This example demonstrates the basic structure of a bank account management system with a simple command-line interface. It includes the creation of customers, different types of accounts, and performing transactions. The execution process is provided in the main method of the BankApp class. Feel free to modify and expand this code based on your specific requirements.

Top of Form