import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class ATMSystem {

private static List<User> users = new ArrayList<>();

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("Welcome to the ATM System");

System.out.println("1. Register");

System.out.println("2. Login");

System.out.println("3. Exit");

System.out.print("Select an option: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:

registerUser(scanner);

break;

case 2:

User user = loginUser(scanner);

if (user != null) {

performATMOperations(user);

} else {

System.out.println("Login failed. Invalid credentials.");

}

break;

case 3:

System.out.println("Thank you for using the ATM. Goodbye!");

scanner.close();

System.exit(0);

default:

System.out.println("Invalid choice. Please try again.");

}

}

}

private static void registerUser(Scanner scanner) {

System.out.print("Enter your name: ");

String name = scanner.next();

System.out.print("Enter your user ID: ");

String userId = scanner.next();

System.out.print("Enter your user account number: ");

String accountNumber = scanner.next();

System.out.print("Create a password: ");

String password = scanner.next();

User newUser = new User(userId, name, accountNumber, password);

users.add(newUser);

System.out.println("Registration successful.");

}

private static User loginUser(Scanner scanner) {

System.out.print("Enter your user ID: ");

String userId = scanner.next();

System.out.print("Enter your password: ");

String password = scanner.next();

for (User user : users) {

if (user.getUserId().equals(userId) && user.getPassword().equals(password)) {

System.out.println("Login successful.");

return user;

}

}

return null;

}

private static void performATMOperations(User user) {

ATM atm = new ATM(user);

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("ATM Operations:");

System.out.println("1. View Transaction History");

System.out.println("2. Withdraw");

System.out.println("3. Deposit");

System.out.println("4. Transfer");

System.out.println("5. Logout");

System.out.print("Select an option: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:

atm.viewTransactionHistory();

break;

case 2:

System.out.print("Enter the amount to withdraw: ");

double withdrawAmount = scanner.nextDouble();

atm.withdraw(withdrawAmount);

break;

case 3:

System.out.print("Enter the amount to deposit: ");

double depositAmount = scanner.nextDouble();

atm.deposit(depositAmount);

break;

case 4:

System.out.print("Enter recipient's user account number: ");

String recipientAccountNumber = scanner.next();

User recipient = getUserByAccountNumber(recipientAccountNumber);

if (recipient != null) {

System.out.print("Enter the amount to transfer: ");

double transferAmount = scanner.nextDouble();

atm.transfer(recipient, transferAmount);

} else {

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

}

break;

case 5:

System.out.println("Logout successful.");

return;

default:

System.out.println("Invalid choice. Please try again.");

}

}

}

private static User getUserByAccountNumber(String accountNumber) {

for (User user : users) {

if (user.getAccountNumber().equals(accountNumber)) {

return user;

}

}

return null;

}

}

class User {

private String userId;

private String name;

private String accountNumber;

private String password;

private double balance;

private List<Transaction> transactionHistory;

public User(String userId, String name, String accountNumber, String password) {

this.userId = userId;

this.name = name;

this.accountNumber = accountNumber;

this.password = password;

this.balance = 0.0;

this.transactionHistory = new ArrayList<>();

}

public String getUserId() {

return userId;

}

public String getName() {

return name;

}

public String getAccountNumber() {

return accountNumber;

}

public String getPassword() {

return password;

}

public double getBalance() {

return balance;

}

public List<Transaction> getTransactionHistory() {

return transactionHistory;

}

public void setBalance(double balance) {

this.balance = balance;

}

// Other methods for viewing transaction history, withdrawing, depositing, and transferring

}

class Transaction {

private String type;

private double amount;

private LocalDateTime dateTime;

public Transaction(String type, double amount) {

this.type = type;

this.amount = amount;

this.dateTime = LocalDateTime.now();

}

public String getType() {

return type;

}

public double getAmount() {

return amount;

}

public LocalDateTime getDateTime() {

return dateTime;

}

}

class ATM {

private User currentUser;

public ATM(User user) {

this.currentUser = user;

}

public void viewTransactionHistory() {

System.out.println("Transaction History:");

for (Transaction transaction : currentUser.getTransactionHistory()) {

System.out.println(

transaction.getType() + " " + transaction.getAmount() + " at " + transaction.getDateTime());

}

}

public void withdraw(double amount) {

if (amount <= 0) {

System.out.println("Invalid amount for withdrawal.");

return;

}

if (amount <= currentUser.getBalance()) {

currentUser.setBalance(currentUser.getBalance() - amount);

currentUser.getTransactionHistory().add(new Transaction("Withdrawal", amount));

System.out.println("Withdrawal successful. Current balance: " + currentUser.getBalance());

} else {

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

}

}

public void deposit(double amount) {

if (amount <= 0) {

System.out.println("Invalid amount for deposit.");

return;

}

currentUser.setBalance(currentUser.getBalance() + amount);

currentUser.getTransactionHistory().add(new Transaction("Deposit", amount));

System.out.println("Deposit successful. Current balance: " + currentUser.getBalance());

}

public void transfer(User recipient, double amount) {

if (amount <= 0) {

System.out.println("Invalid amount for transfer.");

return;

}

if (amount <= currentUser.getBalance()) {

currentUser.setBalance(currentUser.getBalance() - amount);

recipient.setBalance(recipient.getBalance() + amount);

currentUser.getTransactionHistory().add(new Transaction("Transfer to " + recipient.getName(), -amount));

recipient.getTransactionHistory().add(new Transaction("Transfer from " + currentUser.getName(), amount));

System.out.println("Transfer successful. Current balance: " + currentUser.getBalance());

} else {

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

}

}

}