package bank\_management\_system;

import java.util.InputMismatchException;

import java.util.Scanner;

import java.util.ArrayList;

import java.util.List;

public class Bank {

private List<Customer> customers;

private Scanner scanner;

public Bank() {

this.customers = new ArrayList<>();

this.scanner = new Scanner(System.in);

}

public void start() {

System.out.println("Welcome to the Bank Management System!");

mainMenu();

}

private void mainMenu() {

int choice;

do {

System.out.println("\n--- Main Menu ---");

System.out.println("1. Create New Customer");

System.out.println("2. Manage Existing Customer");

System.out.println("3. Display All Customers");

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

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

try {

choice = scanner.nextInt();

scanner.nextLine(); // Consume newline left-over

switch (choice) {

case 1:

createNewCustomer();

break;

case 2:

manageExistingCustomer();

break;

case 3:

displayAllCustomers();

break;

case 4:

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

break;

default:

System.out.println("Invalid choice. Please enter a number between 1 and 4.");

break;

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Consume the invalid input

choice = 0; // Set choice to 0 to re-enter the loop

} catch (Exception e) {

System.err.println("An unexpected error occurred: " + e.getMessage());

choice = 0; // Set choice to 0 to re-enter the loop

}

} while (choice != 4);

scanner.close(); // Close the scanner when the application exits

}

private void createNewCustomer() {

System.out.println("\n--- Create New Customer ---");

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

String name = scanner.nextLine();

System.out.print("Enter customer address: ");

String address = scanner.nextLine();

System.out.print("Enter customer phone number: ");

String phoneNumber = scanner.nextLine();

Customer newCustomer = new Customer(name, address, phoneNumber);

if (newCustomer.getName().equals("INVALID\_NAME") || newCustomer.getPhoneNumber().equals("INVALID\_PHONE")) {

System.out.println("Customer not created due to invalid name or phone number.");

return; // Go back to the main menu

}

logical duplicates)

for (Customer cust : customers) {

if (cust.getCustomerId().equals(newCustomer.getCustomerId())) {

System.out.println("Error: A customer with this ID already exists. Please try again.");

return;

}

}

customers.add(newCustomer);

System.out.println("Customer " + newCustomer.getName() + " created successfully with ID: " + newCustomer.getCustomerId());

}

private void manageExistingCustomer() {

System.out.println("\n--- Manage Existing Customer ---");

if (customers.isEmpty()) {

System.out.println("No customers registered yet.");

return; // Go back to the main menu

}

System.out.print("Enter Customer ID or Name to manage: ");

String searchInput = scanner.nextLine();

Customer foundCustomer = null;

for (Customer cust : customers) {

if (cust.getCustomerId().equalsIgnoreCase(searchInput) || cust.getName().equalsIgnoreCase(searchInput)) {

foundCustomer = cust;

break;

}

}

if (foundCustomer == null) {

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

return; // Go back to the main menu

}

System.out.println("Managing customer: " + foundCustomer.getName());

customerManagementMenu(foundCustomer);

}

private void customerManagementMenu(Customer customer) {

int choice;

do {

System.out.println("\n--- " + customer.getName() + "'s Menu ---");

System.out.println("1. Display Customer Info");

System.out.println("2. Add New Account");

System.out.println("3. Manage Accounts");

System.out.println("4. Back to Main Menu");

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

try {

choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

customer.displayCustomerInfo();

break;

case 2:

addNewAccount(customer);

break;

case 3:

manageCustomerAccounts(customer);

break;

case 4:

System.out.println("Returning to Main Menu.");

break; // Exit this loop, returning to mainMenu()

default:

System.out.println("Invalid choice. Please enter a number between 1 and 4.");

break;

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Consume invalid input

choice = 0; // Set choice to 0 to re-enter the loop

}

} while (choice != 4);

}

private void addNewAccount(Customer customer) {

System.out.println("\n--- Add New Account for " + customer.getName() + " ---");

if (customer.hasAccount()) {

System.out.println("Error: This customer already has an account. Each customer can only have one account.");

System.out.println("Account cannot be created.");

return; // Go back to customerManagementMenu

}

System.out.println("Select Account Type:");

System.out.println("1. Savings Account");

System.out.println("2. Checking Account");

System.out.println("3. Back to Customer Menu");

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

int typeChoice;

try {

typeChoice = scanner.nextInt();

scanner.nextLine(); // Consume newline

if (typeChoice == 3) {

System.out.println("Returning to Customer Menu.");

return; // Go back to customerManagementMenu

}

if (typeChoice != 1 && typeChoice != 2) { // Validate account type choice

System.out.println("Invalid account type choice.");

return;

}

System.out.print("Enter initial balance: $");

double initialBalance = scanner.nextDouble();

scanner.nextLine(); // Consume newline

Account newAccount = null;

try {

if (typeChoice == 1) {

newAccount = new SavingsAccount(customer.getName(), initialBalance);

} else if (typeChoice == 2) {

newAccount = new CheckingAccount(customer.getName(), initialBalance);

}

if (newAccount != null) {

customer.addAccount(newAccount); // Add the new account to the customer

System.out.println("Account " + newAccount.getAccountNumber() + " created successfully for " + customer.getName());

}

} catch (IllegalArgumentException e) {

System.err.println("Account creation failed: " + e.getMessage());

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a valid number for choice or balance.");

scanner.nextLine(); // Consume invalid input

}

}

private void manageCustomerAccounts(Customer customer) {

System.out.println("\n--- Manage Accounts for " + customer.getName() + " ---");

if (customer.getAccounts().isEmpty()) {

System.out.println("No accounts linked to this customer yet.");

return; // Go back to customerManagementMenu

}

Account selectedAccount = customer.getAccounts().get(0); // Get the first (and only) account

System.out.println("Managing Account: " + selectedAccount.getAccountNumber() + " (Type: " + selectedAccount.getClass().getSimpleName() + ")");

accountOperationsMenu(selectedAccount);

}

private void accountOperationsMenu(Account account) {

int choice;

do {

System.out.println("\n--- Account " + account.getAccountNumber() + " Operations ---");

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

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

System.out.println("3. View Transactions");

System.out.println("4. View Account Info");

System.out.println("5. Apply Interest (Savings Accounts Only)"); // \*\*NEW OPTION: Apply Interest\*\*

System.out.println("6. Back to Customer Account Management");

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

try {

choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

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

double depositAmount = scanner.nextDouble();

scanner.nextLine();

account.deposit(depositAmount);

break;

case 2:

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

double withdrawAmount = scanner.nextDouble();

scanner.nextLine();

account.withdraw(withdrawAmount);

break;

case 3:

System.out.println("\n--- Transactions for " + account.getAccountNumber() + " ---");

if (account.getTransactions().isEmpty()) {

System.out.println("No transactions yet.");

} else {

for (Transaction t : account.getTransactions()) {

System.out.println(t);

}

}

break;

case 4:

account.getAccountInfo();

break;

case 5: // \*\*NEW CASE: Apply Interest\*\*

if (account instanceof SavingsAccount) {

((SavingsAccount) account).applyInterest();

} else {

System.out.println("Interest can only be applied to Savings Accounts.");

}

break;

case 6:

System.out.println("Returning to Customer Account Management.");

break; // Exit this loop, returning to manageCustomerAccounts()

default:

System.out.println("Invalid choice. Please enter a number between 1 and 6.");

break;

}

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter a number.");

scanner.nextLine(); // Consume invalid input

choice = 0; // Set choice to 0 to re-enter the loop

}

} while (choice != 6);

}

private void displayAllCustomers() {

System.out.println("\n--- All Registered Customers ---");

if (customers.isEmpty()) {

System.out.println("No customers registered yet.");

} else {

for (Customer cust : customers) {

cust.displayCustomerInfo();

}

}

System.out.println("--------------------------------");

}

}