Project Report:

Critical Thinking 1 - Bank Account

Alejandro Ricciardi

Colorado State University Global

CSC372: Programming 2

Professor: Dr. Vanessa Cooper

June 16, 2024

Project Report:

Critical Thinking 1 - Bank Account

This documentation is part of the Critical Thinking 1 Assignment from CSC372: Programming 2 at Colorado State University Global. This Project Report is an overview of the program's functionality and testing scenarios including console output screenshots. The program is coded in Java JDK-21; and is named Critical Thinking 1 (Bank Account). The program is composed of a Main class, BankAccount class, and a CheckingAccount class that extends the BankAccount.

The Assignment Direction

Option #1: Implementing a Superclass Bank Account
Part 1: Implement a superclass BankAccount that has the following fields and methods.

Fields:
string firstName
string lastName
int accountID
double balance

Methods:
constructor(): initialize balance to zero
deposit() - will accept a single value double parameter; the parameter value is added to the existing balance
withdrawal() - accepts a single value double dollar amount; the parameter value is subtracted from the existing balance
Setters and getters for firstName, lastName, and accountID
getBalance() getter to return the balance
accountSummary() - prints all account information
Part 2: Implement a CheckingAccount class that inherits from the BankAccount class, that:

Has an interest rate attribute
Allows overdraft withdrawals and charges a $30 fee
Methods:

processWithdrawal() - will display a negative balance that includes a $30 overdraft fee and denotes that a fee has been accessed
displayAccount() - should display all superclass attributes and provides an additional interest rate
Ensure that your program has the two required classes and a test class.

Place each Java class into a separate Java source file.

Students must use appropriate version control for all programmatic assignments created. GIT repositories should be established and screen captures of repositories submitted with each assignment.

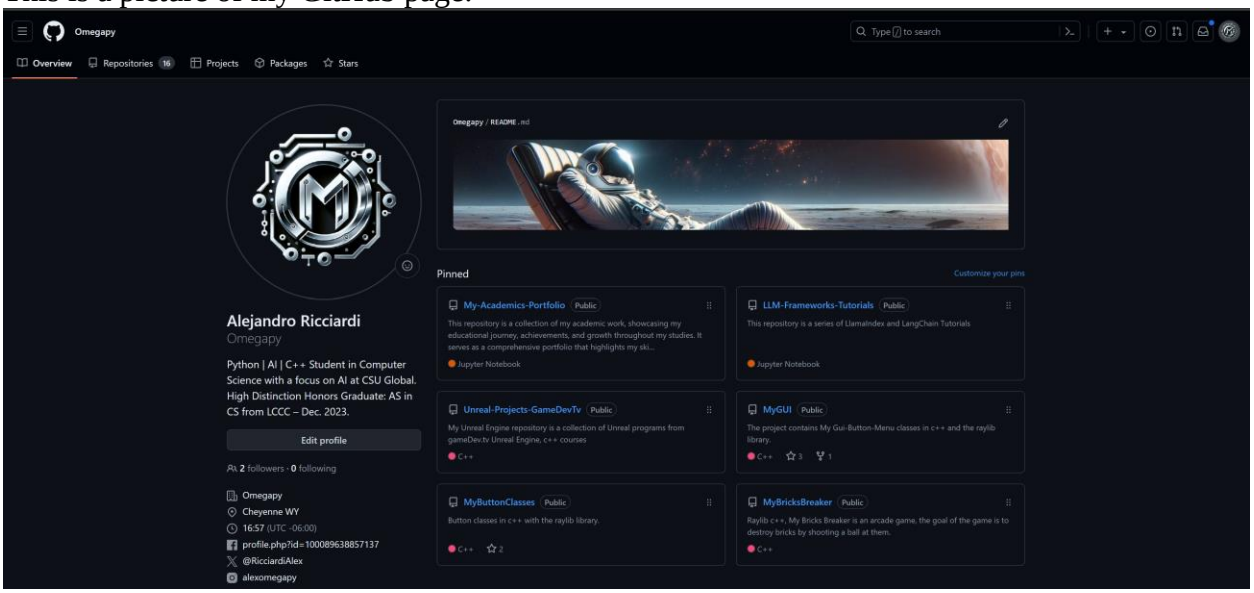
Submit screenshots of your program's execution and output. Include all appropriate source code in a zip file.

⚠️ **My notes:**

- The program implements illegal argument exception checks that are not part of this assignment requirement. Consequently, the illegal argument exception checks screenshot were not added to this document. However, you are welcome to run tests for it.
- Added requirement to the Account ID, it needs to be a 8 digits positive nonzero integer,
- A not-required-assignment withdrawalCash() method similar to the CheckingAccount.processWithdrawal() method was added to the BankAccount class. It will not allow the user to withdrawal cash if the account funds are not sufficient.
- The Main class tests the program functionality.
- **For the source code please see Main.java, BankAccount. Java, and CheckingAccoun.java files.**

Git Repository

This is a picture of my GitHub page:



I use GitHub as my Distributed Version Control System (DVCS), the following is a link to my GitHub, Omegapy.
My GitHub repository that is used to store this assignment is named My-Academics-Portfolio and the link to this specific assignment is:
https://github.com/Omegapy/My-Academics-Portfolio/tree/main/Programming-2-CSC372/Critical-Thinking-1

The BankAccount Class

The BankAccount class represents a basic bank account with fields for first name, last name, account ID, and balance. It provides methods to deposit and withdraw funds and does not allow cash overdrafts in comparison its extended CheckingAccount class allows with an overdraft fee the checks overdraft.

The CheckingAccount Class

The CheckingAccount class represents a checking account that is a child of the BankAccount class. It includes an interest rate and allows for overdraft withdrawals but it applies an overdraft fee.

The Main Class

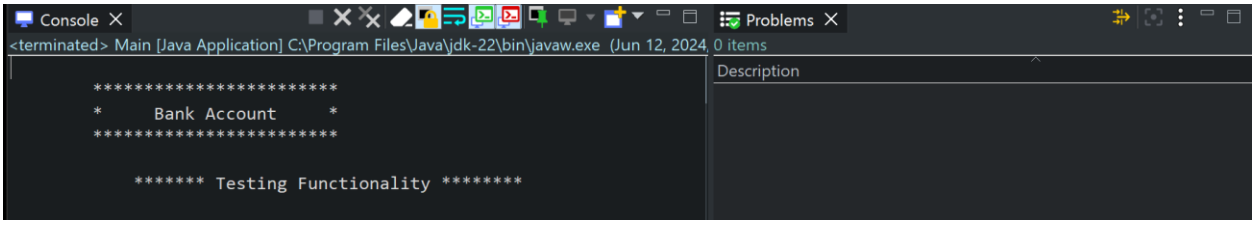
The main method tests the functionality of the BankAccount and CheckingAccount classes. It creates instances using various constructors, sets values using setters, and retrieves values using getters. It also tests deposit and withdrawal methods, including cash and checks overdrafts in the BankAccount and CheckingAccount classes.

More next page.

Screenshots: Program Functionality and Testing Scenarios

Figure 1

Welcome Screen



Note: Eclipse Console output from the program and Problem window showing no items.

Figure 2

BankAccount Class Functionality

```
******* Creating Bank Accounts

*** Default Constructor - bankAccount1 BankAccount object
The BankAccount object was created successfully!

---- Bank Account Summary ----
First Name: Unknown
Last Name: Unknown
Account ID: 00000001
Balance: $0.00

--- setting values in bankAccount1
The First Name was set successfully!
The Last Name was set successfully!
The Accound ID was set successfully!
First Name: John
Last Name: Doe
Account ID: 12563478
Balance: $0.00

--- deposite $500 and cash withdrawal $100 in bankAccount1
The Deposite was added successfully!
The Cash Withdrawal was successful!

---- Bank Account Summary ----
First Name: John
Last Name: Doe
Account ID: 12563478
Balance: $400.00

      ***************

*** constructor-2 with first name and last name - bankAccount2 BankAccount object
The BankAccount object was created successfully!

---- Bank Account Summary ----
First Name: Jane
Last Name: Smith
Account ID: 00000002
Balance: $0.00

      ***************
```

Continues next page

```
*** constructor-3 with first name, last name, and balance -  bankAccount3 BankAccount object
The BankAccount object was created successfully!

---- Bank Account Summary ----
First Name: Alice
Last Name: Johnson
Account ID: 00000003
Balance: $2000.00

--- checking getters with bankAccount3
Getting first name: Alice
Getting last name: Johnson
Getting account ID: 00000003
Getting Balance: $2000.00

**********************************************************
```

Figure 3

CheckingAccount Class Functionality

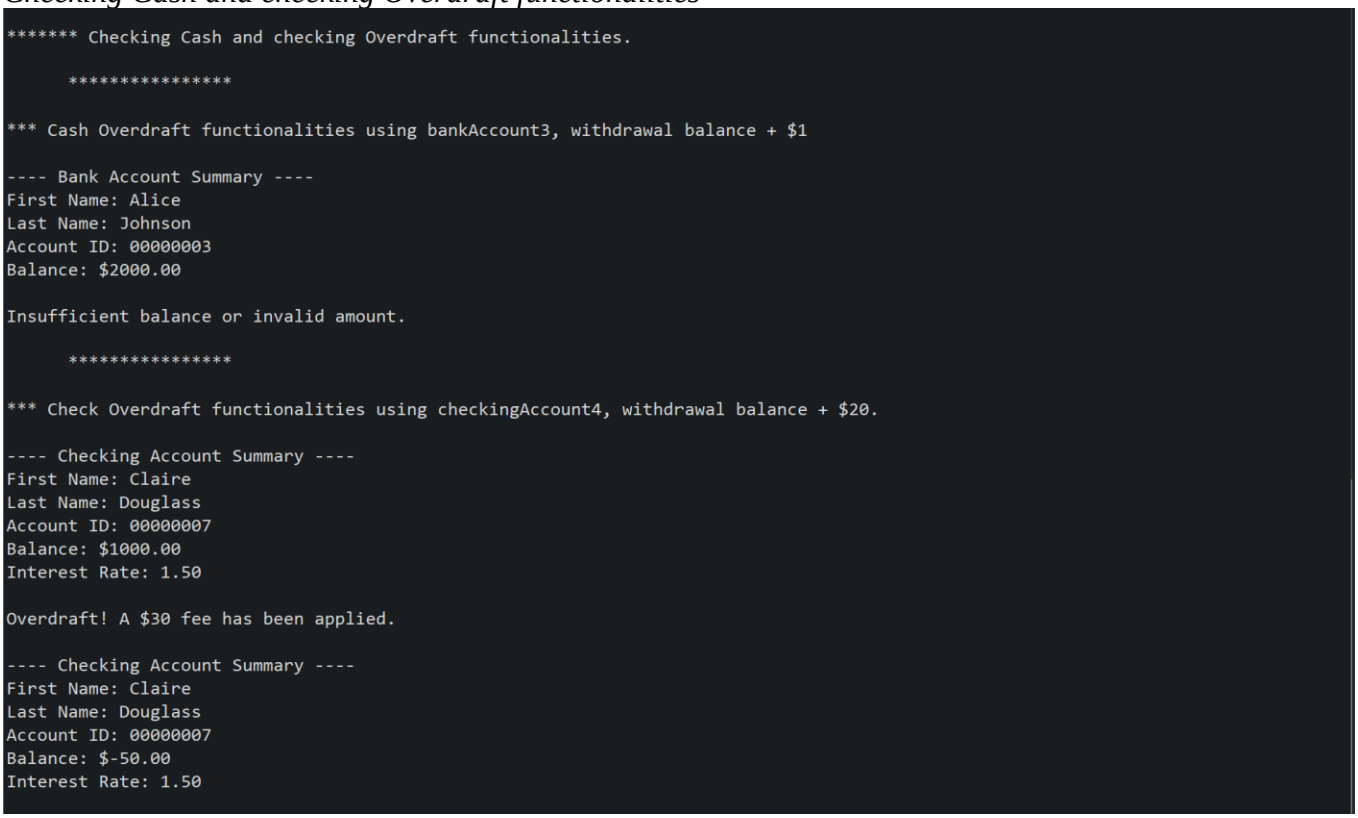
```
******* Creating Checking Accounts

*** Default Constructor - checkingAccount1 CheckingAccount object
The BankAccount object was created successfully!
The BankAccount-CheckingAccount object was created successfully!

---- Checking Account Summary ----
First Name: Unknown
Last Name: Unknown
Account ID: 00000004
Balance: $0.00
Interest Rate: -1.00

--- setting values in checkingAccount1
The First Name was set successfully!
The Last Name was set successfully!
The Accound ID was set successfully!
The Interest Rate was set successfully!

---- Checking Account Summary ----
First Name: Tom
Last Name: Hart
Account ID: 78901234
Balance: $0.00
Interest Rate: 1.50

--- deposite $500 and cash check withdrawal $100 in checkingAccount1
The Deposite was added successfully!
The Withdrawal was successfully!

---- Bank Account Summary ----
First Name: Tom
Last Name: Hart
Account ID: 78901234
Balance: $400.00

      ***************

*** constructor-1 with first name and last name but no balance or interest rate - checkingAccount2 CheckingAccount object
The BankAccount object was created successfully!
The BankAccount-CheckingAccount object was created successfully!

---- Checking Account Summary ----
First Name: Janet
Last Name: Lock
Account ID: 00000005
Balance: $0.00
Interest Rate: -1.00

      ***************
```

Continues next page

```
*** constructor-2 with first name, last name, and interest rate but no balance - checkingAccount3 CheckingAccount object
The BankAccount object was created successfully!
The BankAccount-CheckingAccount object was created successfully!

---- Checking Account Summary ----
First Name: Greg
Last Name: Martin
Account ID: 00000006
Balance: $0.00
Interest Rate: 1.50

      ***************

*** constructor-3 with first name, last name, balance, and interest rate - checkingAccount4 CheckingAccount object
The BankAccount object was created successfully!
The BankAccount-CheckingAccount object was created successfully!

--- checking getters with checkingAccount4
Getting first name: Claire
Getting last name: Douglass
Getting account ID: 00000007
Getting Balance: $1000.00
Getting Interest Rate: 1.50

      ***************

*** constructor-4 with account ID but no interest rate- checkingAccount5 CheckingAccount object

---- Bank Account Summary ----
First Name: Jane
Last Name: Smith
Account ID: 00000002
Balance: $0.00
The BankAccount object was created successfully!
The CheckingAccount object was created successfully!

---- Checking Account Summary ----
First Name: Jane
Last Name: Smith
Account ID: 00000002
Balance: $0.00
Interest Rate: -1.00

      ***************

*** constructor-5 with account ID and interest rate- checkingAccount6 CheckingAccount object

---- Bank Account Summary ----
First Name: Alice
Last Name: Johnson
Account ID: 00000003
Balance: $2000.00

The BankAccount object was created successfully!
The CheckingAccount object was created successfully!

---- Checking Account Summary ----
First Name: Alice
Last Name: Johnson
Account ID: 00000003
Balance: $2000.00
Interest Rate: 1.50

*********************************************************
```

More next page

Figure 4
Checking Cash and checking Overdraft functionalities



As shown in Figure 1 through Figure 4 the functionalities tests run without any issues, displaying the correct outputs as expected.