BINUS UNIVERSITY INTERNATIONAL

BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual/Group* Work)

Student Information:	Surname	Given Names	Student ID Number
	Sudjoko	Nicholas Audley	2301900321

Course Code	: COMP6510	**Course Name**	: Programming Languages
Class	: L2AC	**Name of Lecturer(s)**	: Jude Joseph Lamug Martinez, MCS
Major	: Computer Science		
Title of Assignment (if any)	: MyBank		
Type of Assignment	: Final Project		
Submission Pattern			
Due Date	: 20/06/2020	**Submission Date**	: 20/06/2020

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I/we* understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I/we* declare that the work contained in this assignment is my/our* own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

I. Project Specification

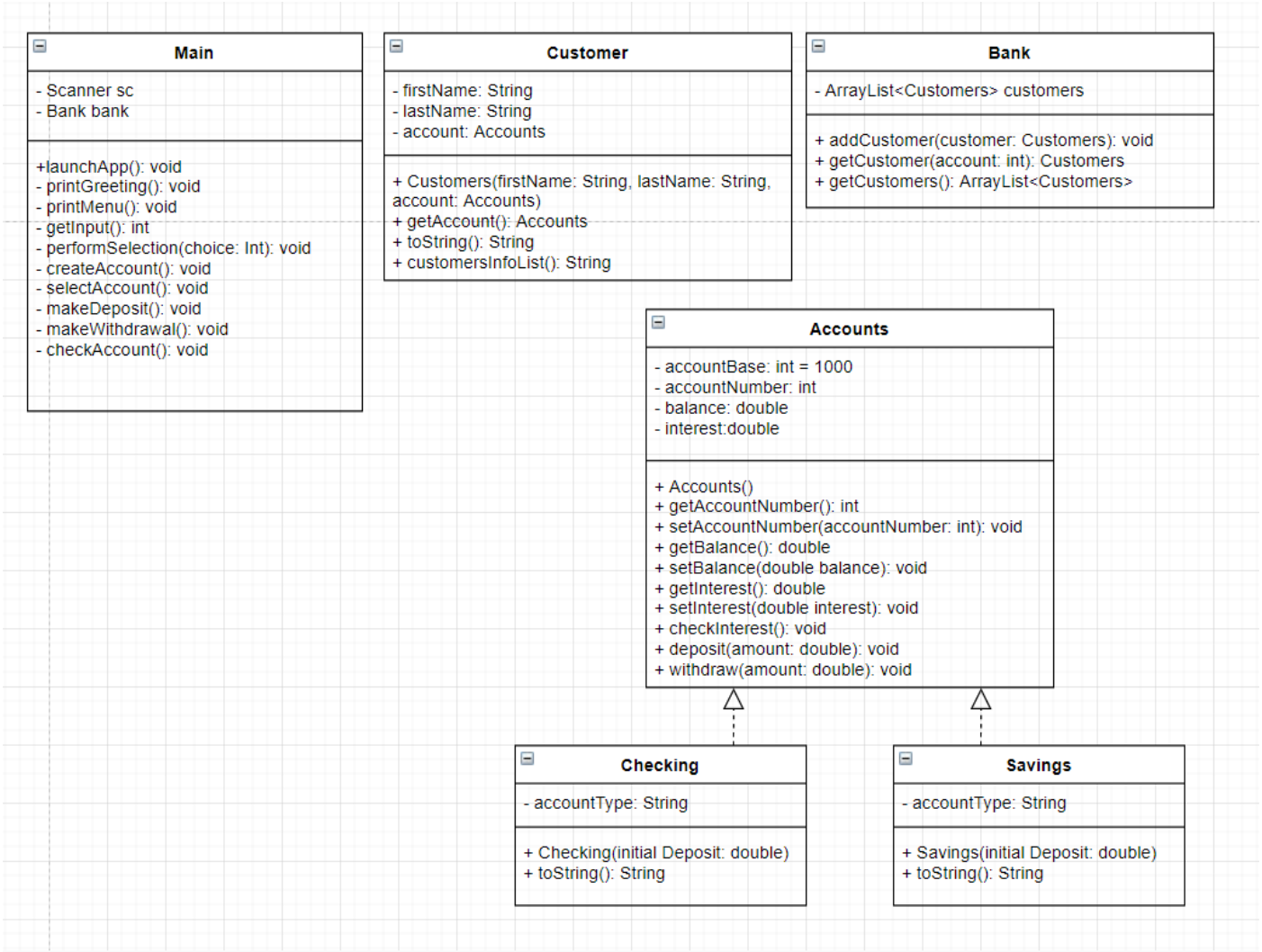
MyBank is a terminal Java application. The intention of this project is to revise what I have learned in the class, as well as teaching me good programming habits and conventions. I believe this project covers almost all the topics I learned in this class this semester.

The application starts by displaying a welcome message to the user, as well as displaying a menu with actions that the user can select. As you can probably tell from the name, MyBank is supposed to be a simulation of a bank. Users can create an account, deposit money, withdraw money, or check their account status.

II. Solution Design

The topic/problem I decided to tackle here is the basic question of how we can make every day life easier by using applications. Of course, there is no way an application this small of a scale will ever be used in a real bank. However, and as an example, a very small business could benefit from ideas of small applications like these, as digital applications have many advantages such as being able to make multiple copies of data or information (backups) really quickly.

Class Diagram:



(Continued in the next page)

I think I have commented my code fairly well so we will just go through the *what* and *how* very quickly in this report
So the program has 6 classes, the first of which is Main.
Here I have usage of primitive data, instance variables, objects, and methods.

```
package MyBank;
import java.util.Scanner;

public class Main {
    // Instance Variables
    Scanner sc = new Scanner(System.in);
    boolean exit = false;
    Bank bank = new Bank();

    public static void main(String[] args) {
        Main screen = new Main();
        screen.launchApp();
    }

    public void launchApp() {
        printGreeting();
        while(!exit) {
            printMenu();
            int choice = getInput();
            performSelection(choice);
        }
    }

    // Method to display a welcome message
    private void printGreeting() {
        System.out.println("Welcome. Thank you for choosing MyBank.");
    }

    // Method to display a menu of selectable actions
    private void printMenu() {
        System.out.println("\nPlease select an action:");
        System.out.println("[1] Create a New Account");
        System.out.println("[2] Make a Deposit");
        System.out.println("[3] Make a Withdrawal");
        System.out.println("[4] Check Account");
        System.out.println("[5] Exit Application");
    }
```

This method here uses the Scanner sc to handle user input. I also used a do-while loop, some exception handling, and some validation checks to make sure the user input is correct.

```
    // Method to receive user inputs
    private int getInput() {
        int choice = 0;
        do {
            System.out.print("\nEnter your choice: ");
            try {
                // Takes user input -> Converts it to Int -> Stores it in choice
                choice = Integer.parseInt(sc.nextLine());
            }
            catch(NumberFormatException e) {
                // Display error message if user input is not a number
                System.out.println("Invalid key press. Please press a number key.");
            }
            if(choice < 1 || choice > 5) {
                // Check if user input is within range
                System.out.println("Invalid key press. Valid number keys: 1 - 5.");
            }
        }
        while(choice < 1 || choice > 5);
        return choice;
    }
```

Here I used a switch to detect what to do after user input

```
// Method to perform action selected by user
private void performSelection(int choice) {
    switch(choice) {
        case 1:
            // Creates an Account
            createAccount();
            break;
        case 2:
            // Makes a Deposit
            makeDeposit();
            break;
        case 3:
            // Makes a Withdrawal
            makeWithdrawal();
            break;
        case 4:
            // Checks Balance
            checkAccount();
            break;
        case 5:
            // Exits the App
            System.out.println("\nThank you for using the MyBank app.");
            System.exit(0);
            break;
        default:
            System.out.println("Error. Please try again.");
    }
}
```

In this method I used the ArrayList from the Bank class, as well as adjusting user input to match array indexes.

```
// Method for Account Selection
private int selectAccount() {
    ArrayList<Customers> customers = bank.getCustomers();
    if(customers.size() <= 0) {
        // Display a message if the bank has 0 customer accounts
        System.out.println("No customers to display. \n");
        return -1;
    }
    // Handling user input for account selection
    System.out.println("Select an account:");
    for(int i = 0; i < customers.size(); i++) {
        // User account list starts with the number [1]
        System.out.println("[" + (i + 1) + "] " + customers.get(i).customersInfoList());
    }
    int account = 0;
    System.out.print("Please enter your selection: ");
    try {
        // However, account indexes start at 0, so -1 to the number inputed by the user
        account = Integer.parseInt(sc.nextLine()) - 1;
    }
    catch(NumberFormatException e) {
        account = -1;
```

Here we have the entirety of the Bank class with the ArrayList.

Main.java | Bank.java | Accounts.java | Customers.java | Checking.java | Savings.java

```
1  package MyBank;
2  import java.util.ArrayList;
3
4  public class Bank {
5      ArrayList<Customers> customers = new ArrayList<Customers>();
6
7      // Create customer account
8      void addCustomer(Customers customer) {
9          customers.add(customer);
10     }
11
12     Customers getCustomer(int account) {
13         return customers.get(account);
14     }
15
16     // List customer accounts
17     ArrayList<Customers> getCustomers() {
18         return customers;
19     }
20 }
```

Here we have the Accounts class
We hold several variables as well as the setters and getters for them

```
public class Accounts {
    private static int accountBase = 1000;
    private int accountNumber;
    private double balance;
    private double interest;

    Accounts() {
        // accountBase is 1000, the first account created will start with this number
        // the next account created will be 1000 + 1 which is 1001
        // and so on...
        accountNumber = accountBase++;
    }

    public int getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(int accountNumber) {
        this.accountNumber = accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public double getInterest() {
        return interest * 100;
    }

    public void setInterest(double interest) {
        this.interest = interest;
    }
```

And then we finally have one of the subclasses that extends the Accounts class
This is the Checking class, the other one is the Savings class

```
public class Checking extends Accounts{
    private static String accountType = "Checking";

    Checking(double initialDeposit) {
        super();
        this.setBalance(initialDeposit);
        this.checkInterest();
    }

    @Override
    public String toString() {
        return "\nAccount Type: " + accountType + " Account\n" +
                "Account Number: " + this.getAccountNumber() + "\n" +
                "Balance: " + this.getBalance() + "\n" +
                "Interest Rate: " + this.getInterest() + "%";
    }
}
```

III. Evidence of Working Program

So as stated previously this program works from the terminal. So all display and user input is from the terminal.