

**Assignment Cover Sheet**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | | **Module Number and Title** | |
| HND in Computing/HND in Software Engineering | | **Introduction to OOP- SEC4207** | |
| **Student Name & No.** | | **Assessor** | |
| CL/HNDCSE/84/13 | | Ms.Upeka Wijesinghe | |
| **Hand out date** | | | **Submission Date** |
| 29/04/2019 | | | TBC |
| **Assessment type**  Coursework | **Duration/Length of**  **Assessment Type**  Report and Demo | | **Weighting of Assessment**  100% |

|  |  |
| --- | --- |
| **Learner declaration** | |
| I certify that the work submitted for this assignment is my own and research sources are fully acknowledged. | |
| |  |  |  |  | | --- | --- | --- | --- | | **Marks Awarded** | | | | | First assessor | |  | | | IV marks | |  | | | Agreed grade | |  | | | Signature of the assessor |  | Date |  | |

**Feedback Form**

**International College of Business & Technology**

**Module:**

**Student:**

**Assessor:**

**Assignment:**

**Strong features of your work:**

**Areas for improvement:**

**Marks Awarded:**

Contents

[Task 1 3](#_Toc41001778)

[Object Oriented Programming 3](#_Toc41001779)

[Object Oriented Programming Concepts 3](#_Toc41001780)

[Object 3](#_Toc41001781)

[Class 4](#_Toc41001782)

[Inheritance 4](#_Toc41001783)

[Abstraction 5](#_Toc41001784)

[Encapsulation 6](#_Toc41001785)

[Polymorphism 7](#_Toc41001786)

[Task 02 8](#_Toc41001787)

[Use-Case Diagram 8](#_Toc41001788)

[Class Diagram 9](#_Toc41001789)

[Task 3 10](#_Toc41001790)

[10](#_Toc41001791)

[If you enter the correct username, but the wrong password, then you will be prompted with this error message 10](#_Toc41001792)

[11](#_Toc41001793)

[When you enter the correct username and password, you will then be proceeded to the main menu window. 11](#_Toc41001794)

[11](#_Toc41001795)

[12](#_Toc41001796)

[Task 4 15](#_Toc41001797)

[Testing Techniques 15](#_Toc41001798)

[Black Box Testing 15](#_Toc41001799)

[White Box Testing 15](#_Toc41001800)

[Test plan 16](#_Toc41001801)

[Teas Case 18](#_Toc41001802)

[Task 05 32](#_Toc41001803)

[Program Strengths 32](#_Toc41001804)

[Program Weaknesses 32](#_Toc41001805)

[Future Recommendations 32](#_Toc41001806)

[Conclusion 33](#_Toc41001807)

# Task 1

## Object Oriented Programming

The Traditional definition of a programming is: a sequence of instructions to be executed on a computer. In the Object-Oriented Programming (OOP) paradigm, a program that executes is a collection of interacting Objects. In the paradigm, the programs we specify what are in these objects and how these objects behave. The main purpose of object-oriented programing is to simplify the design, programming and most importantly debugging a program. So, to modify a particular data, it is easy to identify which function use to.

## 

## Object Oriented Programming Concepts

### Object

The "thing" that has type, identity, state and behavior

* type - belongs to a "class" of similar objects.
* identity - is a distinct "instance" of a class of object.
* state/attribute - has a set of properties (as known as fields).
* each field can have different values.
* behavior - has "methods" (things that the object knows how to do). may have input parameters, output (or "return") type and has "body" code
* we say we "call" a method on the object.

Each object is an "instance" of that "type" of object.

Each instance has its own values for its attributes.

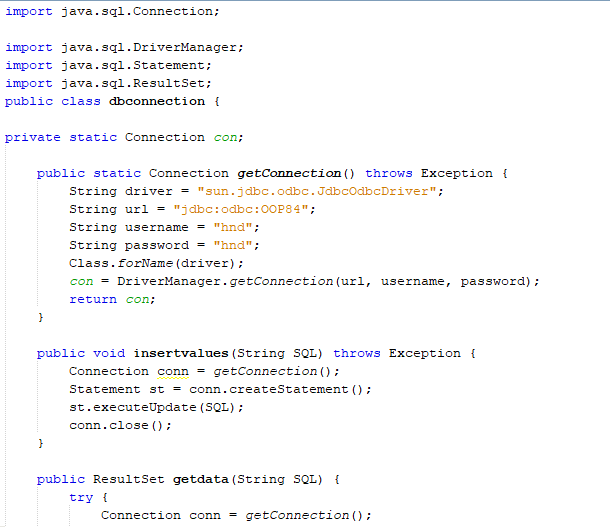
e.g., different accounts can have different balances.

example;

### Class

A class is a template that is used to create objects and methods. A class is one of the defining ideas of object-oriented programming. A class is an entity that determines how an object will behave and what the object will contain. (Guru99, 2018) A class can have subclasses that can inherit characteristics from the class; that is, in relation to each subclass, the class becomes the superclass. Subclasses can also define their own variables and methods that are not part of their superclass.





### Inheritance

Inheritance is a mechanism where one object acquires all the properties and behaviors of a parent object. The idea behind inheritance is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Five types of Inheritance-

1. Single Inheritance (one derived class from one base class)

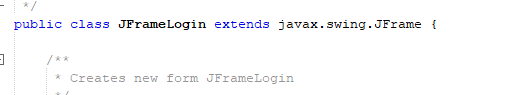
2. Multiple (single derived class from two or more based classes)

3. Hierarchical (multiple...from one base class)

4. Multilevel (derived class from a class what inherited from other)

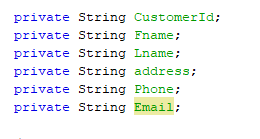
5. Hybrid (combination of Hierarchical and Multilevel inheritance)

example;



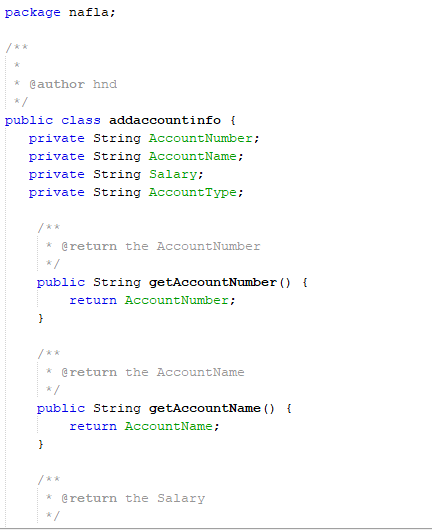
### Abstraction

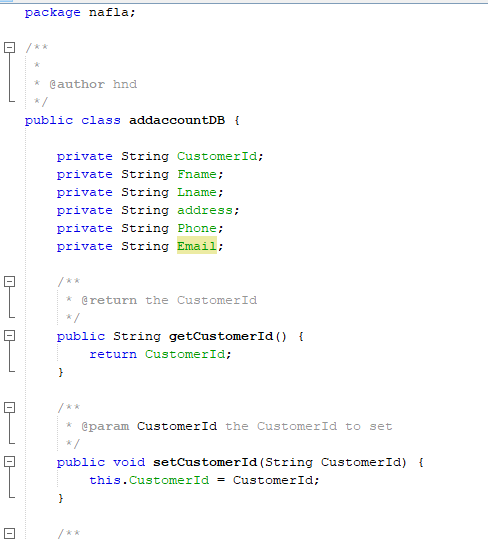
Abstraction is a method which is used to select only the relevant details from a larger pool of data. This method helps to reduce programming complexity and effort. For an example, let’s say that you want to create a banking application. You are supposed to collect all the information regarding your customer. Let’s suppose that you collect these information from your customer:



### Encapsulation

Encapsulation is a mechanism of wrapping variables and the code acting on methods together as a single unit. In encapsulation, the variables of a class will be hidden from other classes. This means that the variables will be private. These variables can be accessed only through the methods of their current class.



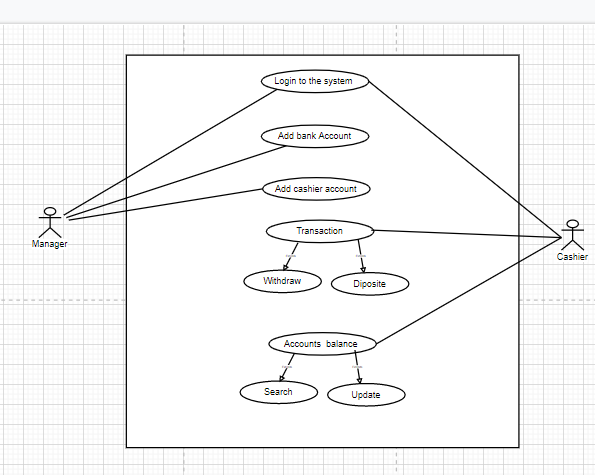


### Polymorphism

Polymorphism is a concept where one name can have many forms. It is the capability of a method to do different things based on the object that it is acting upon. In other words, polymorphism allows you to define one interface and have multiple implementations. There are two versions of polymorphism in Java: (SINGH, 2018)

# Task 02

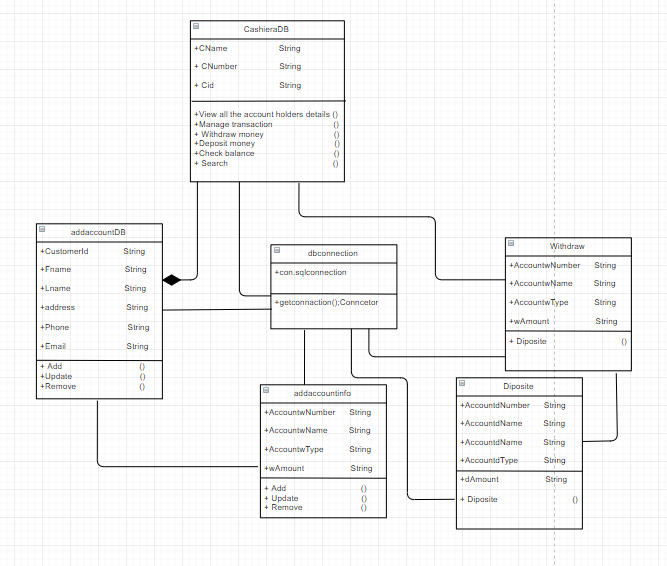
## Use-Case Diagram



Assumptions

* Manager can update Customers bank info as well.
* Manager can view bank info other than the payment details.
* All three roles have to log in to the system to access their powers
* Manager can create bank accounts

## Class Diagram



# Task 3

The code has been attached separately to the document. A basic manual to the program will be written here. When you run the program, you will be first prompted with a login screen. You will have to enter the correct username and the password to proceed. If you enter a wrong username, you will be displayed this error:

# 

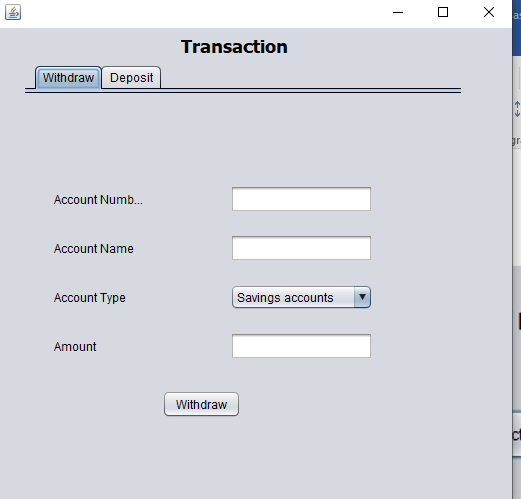
# If you enter the correct username, but the wrong password, then you will be prompted with this error message

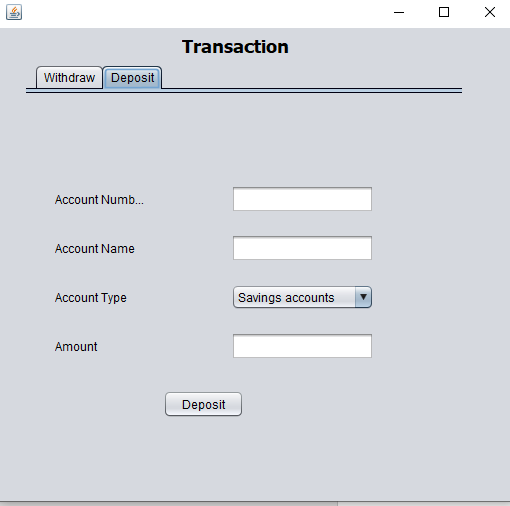
# 

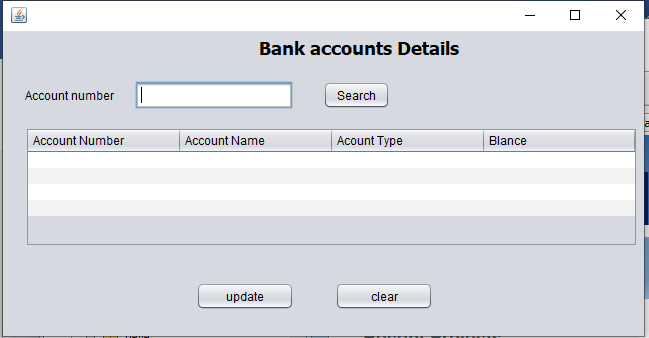
# When you enter the correct username and password, you will then be proceeded to the main menu window.

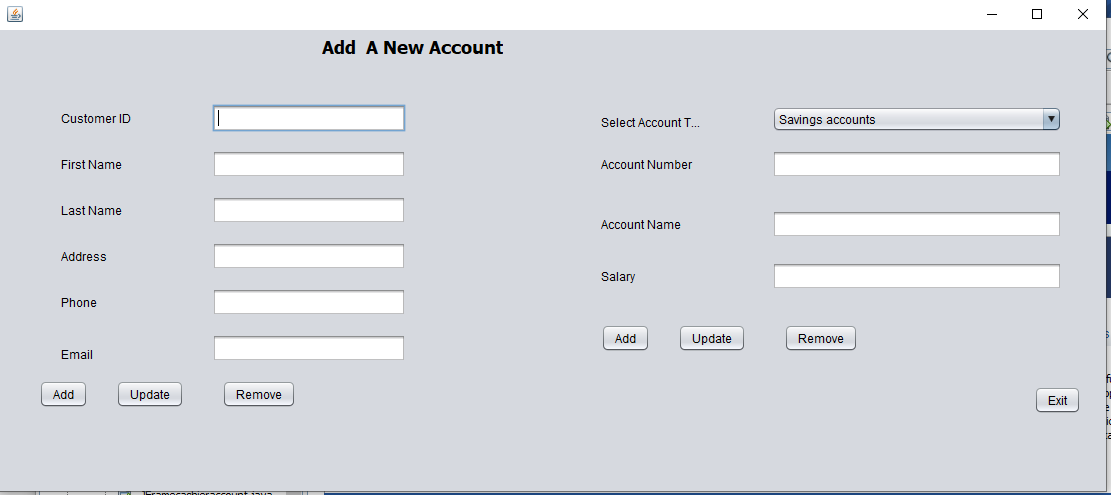
# 

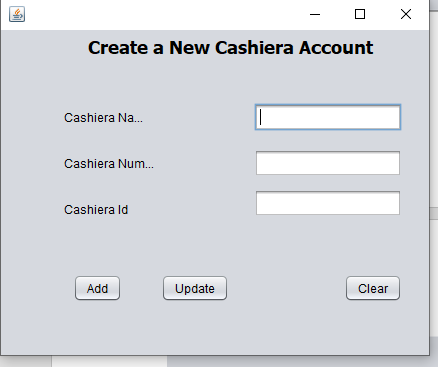
# 











# Task 4

## Testing Techniques

There are various methods to test an application or a software after it is done. Here are a few of them

## Black Box Testing

A software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. Test design techniques include Equivalence partitioning, Boundary Value Analysis, Cause-Effect Graphing. (STF, 2018)

White Box Testing

A software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. Test design techniques include Control flow testing, Data flow testing, Branch testing, Path testing. (STF, 2018)

## Test plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Test*** | | **Test name** | **Changes made** | **Expected result** | **Pass/Fail** |
| 1 | User Login | | Enter the correct username and password. | Display “Login Successful”  Message & Displays Main Menu interface. | Pass |
| 2 | User Login | | Enter an incorrect username or password to the system. | Display error message. | Pass |
| 3 | Main Menu | | Click cashier button | Display cashier interface | Pass |
| 4 | Main Menu | | Click manager button | Display manager interface | Pass |
| 5 | Cashier Menu Transaction | | Cashier Menu Transaction button | Display Account withdraw and deposit interface | Pass |
| 6 | Cashier Transaction  Withdraw money | | Enter the details to the text boxes and click “ withdraw ”  button | Display a message “Withdraw successfully” and data will be save in the data base | pass |
| 7 | Cashier Transaction  Deposited money | | Enter the details to the text boxes and click “Deposited”  button | Display a message “Deposited successfully” and data will be save in the data base | pass |
| 8 | Cashier Transaction  Withdraw money | | Filling the textboxes with blanks | Displays all the error message. | Pass |
| 9 | Cashier Transaction  Deposited money | | Filling the textboxes with blanks | Displays all the error message. | Pass |
| 10 | Cashier Menu Account balance | | Click Cashier Menu Account balance button | Display the bank account details interface | Pass |
| 11 | Cashier Account balance | | Search customer account details using by account number | Display the account details | Pass |
| 12 | Manager add new customer add create bank account | | Enter the details to the text boxes and click “add”  button | Display a message “account details added successfully” and data will be save in the data base | pass |
| 13 | Manager add new customer add create bank account | | Filling the textboxes with blanks | Display the all the error massages | Pass |
| 14 |  | | Filling the textboxes with blanks | Displays an error message | Pass |

## Teas Case

|  |  |
| --- | --- |
| Test Case ID | TC1 |
| Test Objective | Check Student Registration Work correctly |
| Expected Result | Display “Login Successful”  Message & Displays Main Menu interface. |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC2 |
| Test Objective | Enter an incorrect username or password to the system |
| Expected Result | Display error message. |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC3 |
| Test Objective | Click cashier button |
| Expected Result | Display cashier interface |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC4 |
| Test Objective | Click cashier button |
| Expected Result | Display cashier interface |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC5 |
| Test Objective | Cashier Menu Transaction button |
| Expected Result | Display Account withdraw and deposit interface |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC6 |
| Test Objective | Enter the details to the text boxes and click “ withdraw ”  button |
| Expected Result | Display a message “Withdraw successfully” and data will be save in the data base |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC7 |
| Test Objective | Enter the details to the text boxes and click “Deposited”  button |
| Expected Result | Display a message “Deposited successfully” and data will be save in the data base |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC8 |
| Test Objective | Filling the textboxes with blanks |
| Expected Result | Displays all the error message. |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC9 |
| Test Objective | Filling the textboxes with blanks |
| Expected Result | Displays all the error message. |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC10 |
| Test Objective | Click Cashier Menu Account balance button |
| Expected Result | Display the bank account details interface |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC11 |
| Test Objective | Search customer account details using by account number |
| Expected Result | Display the account details |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC12 |
| Test Objective | Enter the details to the text boxes and click “add”  button |
| Expected Result | Display a message “account details added successfully” and data will be save in the data base |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC13 |
| Test Objective | Filling the textboxes with blanks |
| Expected Result | Display the all the error massages |
| Screenshot |  |
| Conclusion | pass |

|  |  |
| --- | --- |
| Test Case ID | TC14 |
| Test Objective | Filling the textboxes with blanks |
| Expected Result | Displays an error message |
| Screenshot |  |
| Conclusion | pass |

# Task 05

## Program Strengths

Even though what we created was a simple application, this program does the job and has its own strengths.

* This program utilizes the simplicity and makes the size of the software small
* No sluggish loading screens
* Straight to the point
* Reliable
* Efficient
* Easy to understand

## Program Weaknesses

There are a few weaknesses in this program.

* Not much appealing
* The text labels are not well aligned on the forms
* Whenever you move a certain window, and then proceeds to another window, the new window will reset its position and appear on the top left corner. This is a bit annoying.
* The password field is not encrypted

## Future Recommendations

There are a lot of changes to be made to this program. Here are a few:

* Make the program more appealing
* Improve the user interface with GUI advancements
* Encrypt the password field
* Show hints or examples on text fields, so that it is easier for the user to understand

## Conclusion

A basic program for City Bank has been developed and for now, I think it serves the basic needs requested by the Bank. The usage of Object Oriented Programming on this project has affected positively. Some of the core concepts of OOP, which are Abstraction, Encapsulation, Inheritance, Polymorphism, etc. have been used in this project. Encapsulation did a big task by wrapping the variables and the code acting on methods together as a single unit. Even though we didn’t use Polymorphism here, there are possibilities to add them. Therefore, in the future, I hope to utilize the concept of polymorphism as well