

Banking App

Ethan Chiu

501170506

Toronto Metropolitan University

COE528 - 011

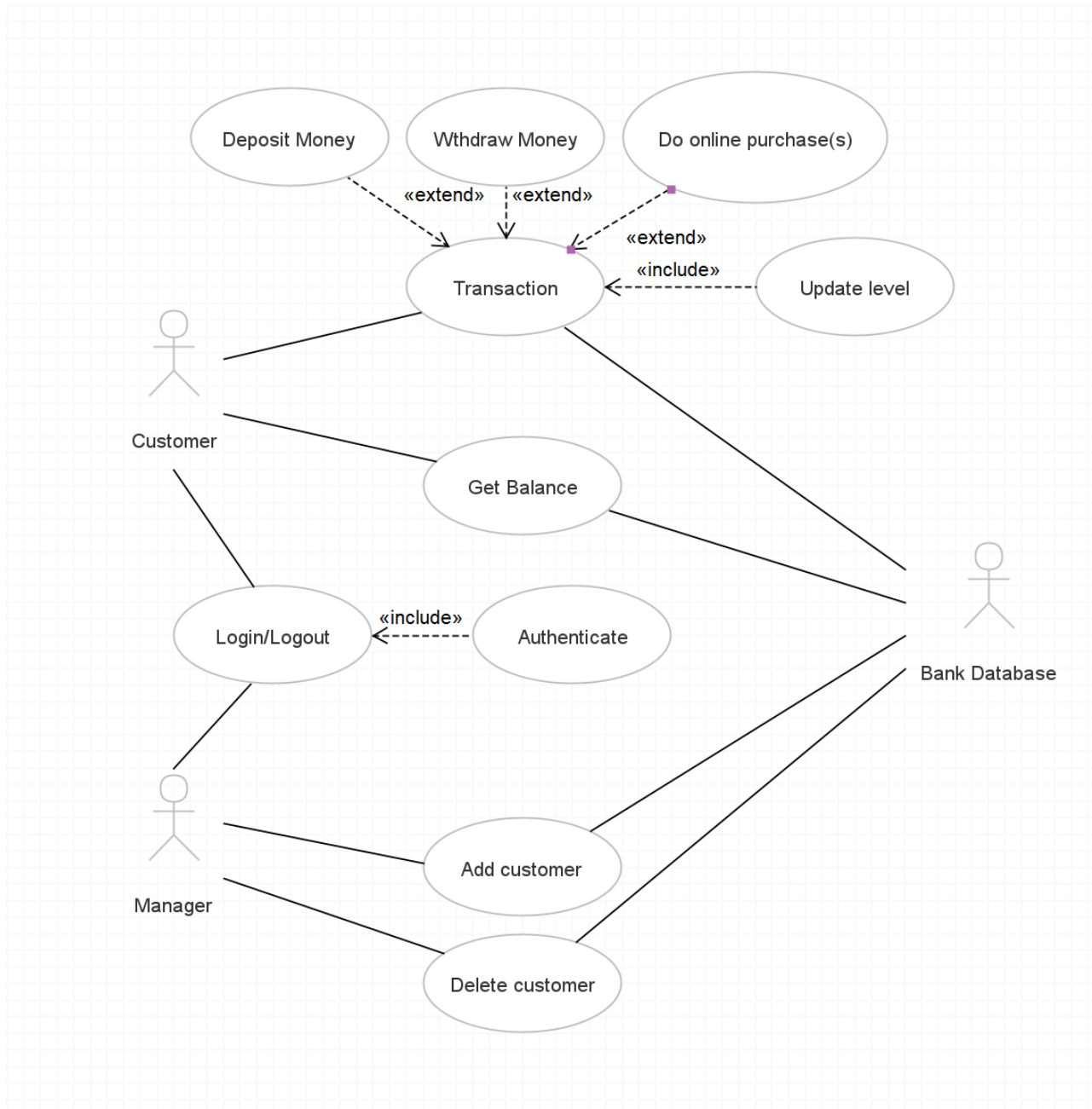
Prof. Boujemaa Guermazi

24th March 2024

Overview

The report for The Bank App Project is presented herein. It consists of sections explaining different parts of the final project for the COE 528 course. This report consists of a Use Case Diagram, Class Diagram, Descriptions of the Case and Class Diagrams, mention of a class with full javadocs, an indication of where the State design pattern was used and any references used.

Use Case Diagram



```

classDiagram
    class FileHandler {
        +String role
        +String username
        +String password
        +void updateBal(String, double)
        +double getBal(String)
    }
    class ManagerController {
        +TextField password
        +TextField username
        +Label status
        +void addCustomer()
        +void logOut()
        +void removeCustomer()
    }
    class CustomerInterfaceController {
        +FileHandler file
        +Label currentBal
        +void switchToDeposit()
        +void initialize()
        +void switchToWithdraw()
        +void switchToPurchase()
        +void logOut()
    }
    class LoginController {
        +TextField username
        +PasswordField password
        +Label status
        +BankAccount bankAccount
        +void login()
    }
    class PurchaseController {
        +Label newBal
        +Label currentBal
        +TextField value
        +Label currentFee
        +Label status
        +void back()
        +void initialize()
        +void purchase()
    }
    class BankAccount {
        +double balance
        +Level lvl
        +FileHandler file
        +String account
        +boolean repOk()
        +double getBal()
        +String toString()
        +boolean deposit(double)
        +boolean purchase(double)
        +void updateLevel()
        +boolean withdraw(double)
    }
    class Silver {
        +BankAccount bankAccount
        +int fee()
    }
    class Gold {
        +BankAccount bankAccount
        +int fee()
    }
    class Platinum {
        +BankAccount bankAccount
        +int fee()
    }
    class Level {
        +BankAccount bankAccount
        +int fee()
    }
    class Account {
        +BankAccount bankAccount
        +int fee()
    }
    class App {
        +App()
        +Scene scene
        +Parent loadFXML(String)
        +void start(Stage)
        +void setRoot(String)
        +void main(String[])
    }

    FileHandler "1" -- "1" ManagerController
    ManagerController "1" -- "1" CustomerInterfaceController
    CustomerInterfaceController "1" -- "1" LoginController
    LoginController "1" -- "1" PurchaseController
    PurchaseController "1" -- "1" BankAccount
    BankAccount "1" -- "1" Silver
    BankAccount "1" -- "1" Gold
    BankAccount "1" -- "1" Platinum
    BankAccount "1" -- "1" Level
    BankAccount "1" -- "1" Account
    App "1" -- "1" BankAccount
  
```

The diagram illustrates the architecture of a banking application. It features several classes with their attributes and methods, and their relationships:

- FileHandler**: Attributes include `role` (String), `username` (String), and `password` (String). Methods include `updateBal(String, double)` (void) and `getBal(String)` (double).
- ManagerController**: Attributes include `password` (TextField), `username` (TextField), and `status` (Label). Methods include `addCustomer()` (void), `logOut()` (void), and `removeCustomer()` (void).
- CustomerInterfaceController**: Attributes include `file` (FileHandler) and `currentBal` (Label). Methods include `switchToDeposit()` (void), `initialize()` (void), `switchToWithdraw()` (void), `switchToPurchase()` (void), and `logOut()` (void).
- LoginController**: Attributes include `username` (TextField), `password` (PasswordField), `status` (Label), and `bankAccount` (BankAccount). Method is `login()` (void).
- PurchaseController**: Attributes include `newBal` (Label), `currentBal` (Label), `value` (TextField), `currentFee` (Label), `status` (Label), and `bankAccount` (BankAccount). Methods include `back()` (void), `initialize()` (void), and `purchase()` (void).
- BankAccount**: Attributes include `balance` (double), `lvl` (Level), `file` (FileHandler), `account` (String), `repOk` (boolean), `getBal` (double), `toString` (String), `deposit` (double) (boolean), `purchase` (double) (boolean), `updateLevel` (void), and `withdraw` (double) (boolean).
- Silver**, **Gold**, **Platinum**, **Level**, and **Account**: These are subclasses of **BankAccount**. They inherit the `fee()` method (int) and have a `bankAccount` (BankAccount) attribute.
- App**: Attributes include `scene` (Scene). Methods include `loadFXML(String)` (Parent), `start(Stage)` (void), `setRoot(String)` (void), and `main(String[])` (void).

Relationships are shown with solid lines (association) and dashed lines (generalization). Multiplicities are indicated at the ends of the association lines.

Description:

The Use Case diagram shows users' interactions with the Banking App system. The Bank App uses a Use Case Diagram to illustrate the Manager and Customer interactions. The Manager can log in, log out, add customers, and delete customers. The customer can log in, log out and perform transactions. The user also contains a level which updates depending on the amount of money they have in their account.

Name	Get Balance
Participating actors	Customer, Bank Database
Entry Conditions	Customer exists within the database
Flow of events	<ol style="list-style-type: none">1. The customer logs into the account2. The Bank Database returns the account balance
Exit conditions	The customer logs out or performs a transaction
Quality Reqts.	None

Description:

This class diagram shows the structure and relationships within the Banking application. The diagram shows various classes with their instance variables as well as the methods they contain. Relationships between classes, such as associations, aggregations, compositions, and inheritances, show how they interact within the system. This diagram shows a visual representation of the system's architecture and design.

Selected Class for Point Number 2:

The class “BankAccount” has been chosen to address point number 2. As per point number 2, this class has the necessary clauses, abstraction function, the rep invariant and implementations as mentioned in point number 2.

State Design Pattern in UML Class Diagram:

As seen in the UML class diagram the classes BankAccount, Level, Silver, Gold and Platinum make up the state design pattern. Where Silver, Gold and Platinum are possible states that level can be, seen in the class diagram.

References