**Project #3**

CSCI6620.81

**Online Banking System**

Spring 2018

Instructor: Dr. Yongming Tang  
**by**

Yelyzaveta Rachenko [ ID: 1744706]

Dhara Patel [ ID: 1820905]

Nidhi Patel [ID: 1824696]

Zhrieh Alkawas [ ID: 1722323]

Due Date: 05/01/18



**Fairleigh Dickinson University**

//Part C (-5), Part D (-3). You scored 92/100.

|  |  |
| --- | --- |
| LIST OF TABLES |  |
| TITLE | **PAGE NO.** |
| Functional Model |  |
| Sign Up |  |
| Table:1 Sign Up successful scenario | 12 |
| Table:2 Sign Up unsuccessful scenario | 12 |
| Table:3 Sign Up cancel scenario | 13 |
| Table:4 Sign Up initial use case | 13 |
| Table:5 Sign Up refine use case | 14 |
| Login |  |
| Table:6 Login successful scenario | 16 |
| Table:7 Login unsuccessful scenario | 16 |
| Table:8 Login cancel scenario | 16 |
| Table:9 login initial use case | 17 |
| Table:10 Login refine use case | 19 |
| Change Username |  |
| Table:11 Change Username successful scenario | 21 |
| Table:12 Change Username unsuccessful scenario | 22 |
| Table:13 Change Username cancel scenario | 22 |
| Table:14 Change Username initial use case | 23 |
| Table:15 Change Username refine use case | 23 |
| Change Password |  |
| Table:16 Change Password successful scenario | 26 |
| Table:17 Change Password unsuccessful scenario | 26 |
| Table:18 Change Password cancel scenario | 27 |
| Table:19 Change Password initial use case | 27 |
| Table:20 Change Password successful scenario | 28 |
| Transfer Money |  |
| Table:21 Transfer Money successful scenario | 31 |
| Table:22 Transfer Money unsuccessful scenario | 31 |
| Table:23 Transfer Money cancel scenario | 32 |
| Table:24 Transfer Money initial use case | 33 |
| Table:25 Transfer Money refine use case | 34 |
| Add Payee |  |
| Table:26 Add Payee successful scenario | 37 |
| Table:27 Add Payee unsuccessful scenario | 38 |
| Table:28 Add Payee cancel scenario | 38 |
| Table:29 Add Payee initial use case | 39 |
| Table:30 Add Payee refine use case | 39 |
| Delete Payee |  |
| Table:31 Delete Payee successful scenario | 42 |
| Table:32 Delete Payee unsuccessful scenario | 42 |
| Table:33 Delete Payee cancel scenario | 43 |
| Table:34 Delete Payee initial use case | 43 |
| Table:35 Delete Payee refine use case | 44 |
| Schedule Payment |  |
| Table:36 Schedule Payment successful scenario | 47 |
| Table:37 Schedule Payment unsuccessful scenario | 47 |
| Table:38 Schedule Payment cancel scenario | 48 |
| Table:39 Schedule Payment initial use case | 49 |
| Table:40 Schedule Payment refine use case | 50 |
| Modify Schedule |  |
| Table:41 Modify Schedule successful scenario | 52 |
| Table:42 Modify Schedule unsuccessful scenario | 53 |
| Table:43 Modify Schedule cancel scenario | 53 |
| Table:44 Modify Schedule initial use case | 54 |
| Table:45 Modify Schedule refine use case | 55 |
| Cancel Payment |  |
| Table:46 Cancel Payment successful scenario | 57 |
| Table:47 Cancel Payment unsuccessful scenario | 57 |
| Table:48 Cancel Payment cancel scenario | 58 |
| Table:49 Cancel Payment initial use case | 58 |
| Table:50 Cancel Payment refine use case | 59 |
| Logout |  |
| Table:51 Log Out successful scenario | 61 |
| Table:52 Log Out unsuccessful scenario | 62 |
| Table:53 Log Out initial use case | 62 |
| Table:54 Log Out refine use case | 63 |
| Boundary Use Case |  |
| Table:55 Boundary use case for configuration | 79 |

|  |  |
| --- | --- |
| LIST OF FIGURES |  |
| TITLE | **PAGE NO.** |
| Dynamic Model |  |
| Sign Up |  |
| Fig 1: Sequence diagram of Sign Up | 14 |
| Fig 2: State diagram of UserAccount | 14 |
| Login |  |
| Fig 3: Sequence diagram of Login | 18 |
| Fig 4: State diagram of UserAccount | 19 |
| Change Username |  |
| Fig 5: Sequence diagram of Change Username | 22 |
| Fig 6: State diagram of UserAccount | 23 |
| Change Password |  |
| Fig 7: Sequence diagram of Change Password | 26 |
| Fig 8: State diagram of UserAccount | 27 |
| Transfer Money |  |
| Fig 9: Sequence diagram of Transfer Money | 31 |
| Fig 10: State diagram of CheckingAccount | 32 |
| Fig 11: State diagram of SavingAccount | 33 |
| Fig 12: State diagram of Transaction | 33 |
| Add Payee |  |
| Fig 13: Sequence diagram of Add Payee | 37 |
| Fig 14: State diagram of Payee | 37 |
| Delete Payee |  |
| Fig 15: Sequence diagram of Delete Payee | 41 |
| Fig 16: State diagram of Payee | 41 |
| Schedule Payment |  |
| Fig 17: Sequence diagram of Schedule Payment | 45 |
| Fig 18: State diagram of Payment | 46 |
| Modify Schedule |  |
| Fig 19: Sequence diagram of Modify Schedule | 49 |
| Fig 20: State diagram of Payment | 50 |
| Cancel Payment |  |
| Fig 21: Sequence diagram of Cancel Payment | 53 |
| Fig 22: State diagram of Payment | 54 |
| Logout |  |
| Fig 23: Sequence diagram of Logout | 56 |
| Fig 24: State diagram of UserAccount | 57 |
|  |  |
| Analysis Object Model |  |
| Fig 25: Initial class diagram | 58 |
| Fig 26: Final class Diagram | 59 |
| Fig 27: Class, Attribute and operation | 60 |
| Fig 28: Object diagram of customer and checking account | 62 |
| Fig 29: Object diagram of customer and saving account | 62 |
| Fig 30: Object diagram of customer and transaction | 63 |
| Fig 31: Object diagram of customer and debitCard | 63 |
| Fig 32: Object diagram of customer and payment | 64 |
| Fig 33: Object diagram of customer and payee | 65 |
| Fig 34: Object diagram of customer and check | 65 |
| Fig 35: Object diagram of transaction, checking account and saving account | 66 |
| Fig 36: Object diagram of checking account, check and payment | 66 |
| Fig 37: Object diagram of checking account and debit card | 67 |
| Fig 38: Object diagram of saving account and debit card | 67 |
| Fig 39: Object diagram of payee and payment | 68 |
| Fig 40: Object diagram of customer and UserAccount | 68 |
|  |  |
| Object Design |  |
| Fig 41: Adapter Design pattern for UserAccount | 69 |
|  |  |
| System Design |  |
| Fig 42: UML deployment diagram for client server base | 76 |

|  |  |  |
| --- | --- | --- |
|  | LIST OF CONTENTS |  |
| SR NO. | **TITLE** | **PAGE NO.** |
|  | **Cover Page** | 1 |
|  | **List of Table** | 2 |
|  | **List of Figure** | 5 |
|  | **Online Banking System** | 10 |
| 1 | **Problem Statement** | 10 |
| 2 | **List of Activities [Functionality]** | 10 |
| 3 | **Analysis Model** | 11 |
|  | 3.1 Sign Up | 11 |
|  | 3.2 Login | 15 |
|  | 3.3 Change username | 19 |
|  | 3.4 Change Password | 23 |
|  | 3.5 Transfer Money | 27 |
|  | 3.6 Add Payee | 34 |
|  | 3.7 Delete Payee | 38 |
|  | 3.8 Schedule Payment | 42 |
|  | 3.9 Modify Payment | 46 |
|  | 3.10 Cancel Payment | 50 |
|  | 3.11 Logout | 54 |
| 4. | **Object Design** | 69 |
|  | 4.1 Design pattern: Adapter | 69 |
|  | 4.2 Interface Specification | 70 |
|  | 4.2.1 Interface specification of Customer | 70 |
|  | 4.2.2 Interface specification of UserAccount | 71 |
|  | 4.2.3 Interface specification of SavingAccount | 72 |
|  | 4.2.4 Interface specification of CheckingAccount | 72 |
|  | 4.2.5 Interface specification of Transaction | 73 |
|  | 4.2.6 Interface specification of Payee | 74 |
|  | 4.2.7 Interface specification of Payment | 74 |
|  | 4.2.8 Interface specification of Debit Card | 75 |
|  | 4.2.9 Interface specification of Check | 75 |
| 5. | **System Design** | 76 |
|  | 5.1 UML deployment Diagram | 76 |
|  | 5.2 Access Control | 77 |
|  | 5.2.1 Global access tables | 77 |
|  | 5.3 Strategies for global control | 79 |
|  | 5.4 Boundary use case | 79 |

|  |  |
| --- | --- |
| Online Banking System | |
| 1.Problem statement: |

Online banking system is a platform that lets the user manage his account with the following functionalities: Sign Up (a user registers in the banking system), Login (a user goes inside their personal page on the banking system website), Change Username (a user creates a new username for their private online banking system), Change Password (a user creates a new password), Add a Payee (a user adds a payee), Delete a Payee (a user deletes a payee), Schedule a Payment (a user schedules a payment that is paid either from the customer’s savings account or checking account), Modify a Payment (a user modifies a payment before its delivery date, such as change payment amount or delivery date), Cancel a Payment (a user cancels a payment before its delivery date), Logout (a user logs out), Account Overview (a user sees all the main links to other pages and sees the important notifications), Transfer (a user puts their money into other account(s)), Bill Pay (a user schedules his/her bill payments), View Statements (a user sees all the transactions).

|  |
| --- |
| 2. A list of activities (functionalities): |

* Customer/Customers **sign up** by providing their personal information such as first/last name, email, and so on
* Customer/Customers **login** by entering the username and password
* Customer/Customers **change a username** by calling their bank and making up a new username.
* Customer/Customers **change a password** by creating a new password that satisfies all the system’s requirements and is not the same as their last password.
* Customer/Customers **add a payee** by entering the payee’s information as well as the amount of money needed to be payed and the duration.
* Customer/Customers **delete a payee** by erasing the payee’s information from a customer’s account.
* Customer/Customers **schedule a payment** by entering the account a customer wants to schedule a payment from, the amount of money needed to be transferred, and the duration.
* Customer/Customers **modify a payment** by changing either the payment amount or its delivery date.
* Customer/Customers **cancel a payment** by erasing the information related to the given payment.
* Customer/Customers **logout** by making the system forget a customer’s banking information on a given static device.
* Customer/Customers see their **account overview** once they login successfully
* Customer/Customers **transfer** an amount of money from their Checking/Savings account to their Savings/Checking account after selecting source and destination and entering the amount.
* Customer/Customers set up **bill pay** by scheduling their payments and entering the amount of money they want to be sent to each of the bills.
* Customer/Customers **view statements** by going to the “View Statements” page from the account overview page.

|  |
| --- |
| 3. Analysis Model |

**3.1 A customer is able to sign up.**

**Functional Model:**

Functionality: Customers sign up by entering their personal information.

Functionality name: **SignUp**

**Scenarios**

**Example 1:** Susan attempts to sign up as Susan Smith with her email [smith@gmail.com](mailto:smith@gmail.com) and with her CustomerId Susan123 or SSN 123-34-5678 as well as with her password Susan@4422 on 5/01/2018 at 2:00PM. Scenario of an example above.

|  |  |
| --- | --- |
| *Use case name* | SignUpSuccessful:SignUp |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan activates the “SignUp” function of the system by clicking on SIGN UP.  2. Susan enters Susan, her first name, Smith, her last name, [smith@gmail.com](mailto:smith@gmail.com), her email, 123-34- 5678 her social security number, Susan@4422, her password, Susan@4422, her confirmed password 3. Susan signs up by clicking SUBMIT.  4. The system shows “Sign Up Successful”. |

**[Table: 1 Sign Up Successful Scenario]**

**Example 2:** Susan attempts to sign up as Susan Smith with her email [smith@gmail.com](mailto:smith@gmail.com) and with her CustomerId Susan123 or SSN 123-34-5678 as well as with her password Susan@4422 on 5/01/2018 at 2:00PM. Scenario of an example above.

|  |  |
| --- | --- |
| *Use case name* | SignUpUnsuccessful:SignUp |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan activates the “SignUp” function of the system by clicking on SIGN UP. 2. Susan enters Susan, her first name, Smith, her last name, [smith@gmail.com](mailto:smith@gmail.com), her email, 123-34- 5678 her social security number, Susan@4422, her password, Susan@442, her confirmed password.  3.Susan signs up by clicking SUBMIT.  4. The system shows “Sign Up Unsuccessful” because of  the password is not correct. |

**[Table: 2 Sign Up Unsuccessful Scenario]**

**Example 3:** Susan attempts to sign up as Susan Smith with her email [smith@gmail.com](mailto:smith@gmail.com) and with her CustomerId Susan123 or SSN 123-34-5678 as well as with her password Susan@4422 on 5/01/2018 at 2:00PM. Scenario of an example above.

|  |  |
| --- | --- |
| *Use case name* | SignUpCancel:SignUp |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan activates the “SignUp” function of the system by clicking on SIGN Up.  2. Susan enters Susan, her first name, Smith, her last name, [smith@gmail.com](mailto:smith@gmail.com), her email, 123-34- 5678 her social security number, Susan@4422, her password, Susan@4422, her confirmed password.  3. Susan cancels by clicking CANCEL.  4. The system shows “Sign Up Cancelled”. |

**[Table:3 Sign Up Cancel Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | SignUp |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1. The customer is signing up in the online banking system on a specific date at a specific time.  2. “Sign Up Successful”, “Sign Up Unsuccessful” or “Sign Up Cancelled” is shown in the middle of the screen on the same website page. |
| *Entry conditions* | The customer is logged in his/ her account and is on the account overview page. |
| *Exit conditions* | A confirmation is shown. |
| *Quality Requirements* | The process must be done in less than 3 seconds. |

**[Table: 4 Sign Up Initial Use Case]**

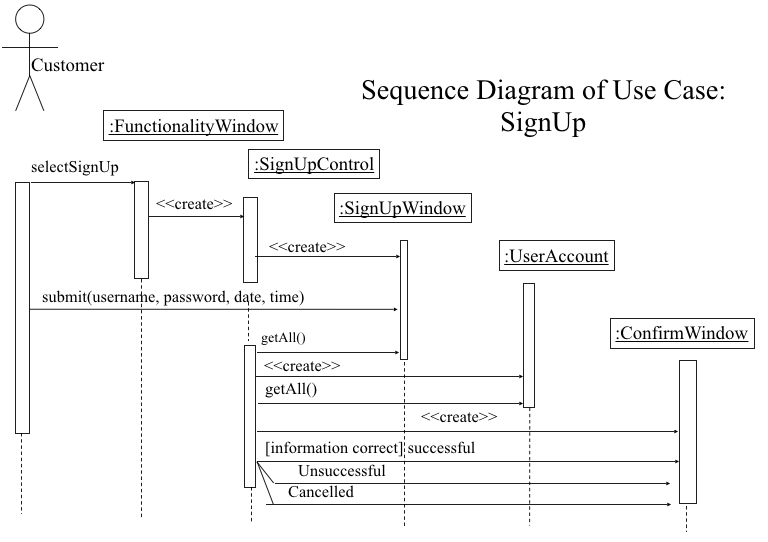
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | SignUp |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1. The customer is signing up in the online banking system on a specific date at a specific time by entering all required information in the SignUpWindow.  2.SignUpControl object created with entering the data by SignUpWindow.  3.SignUpControl object gets the information. If the information that entered is correct, SignUpControl object creates UserAccount object and invokes an operation of create User Account.  4.SignUpControl object creates the ConfirmWindow to show “Sign Up Successful”, “Sign Up Unsuccessful” or “Sign Up Cancel” in the middle of the screen on the same website page. |
| *Entry conditions* | The customer is logged in his/ her account and is on the account overview page. |
| *Exit conditions* | A confirmation is shown. |
| *Quality Requirements* | The process must be done in less than 3 seconds. |

**[Table: 5 Sign Up Refine Use Case]**

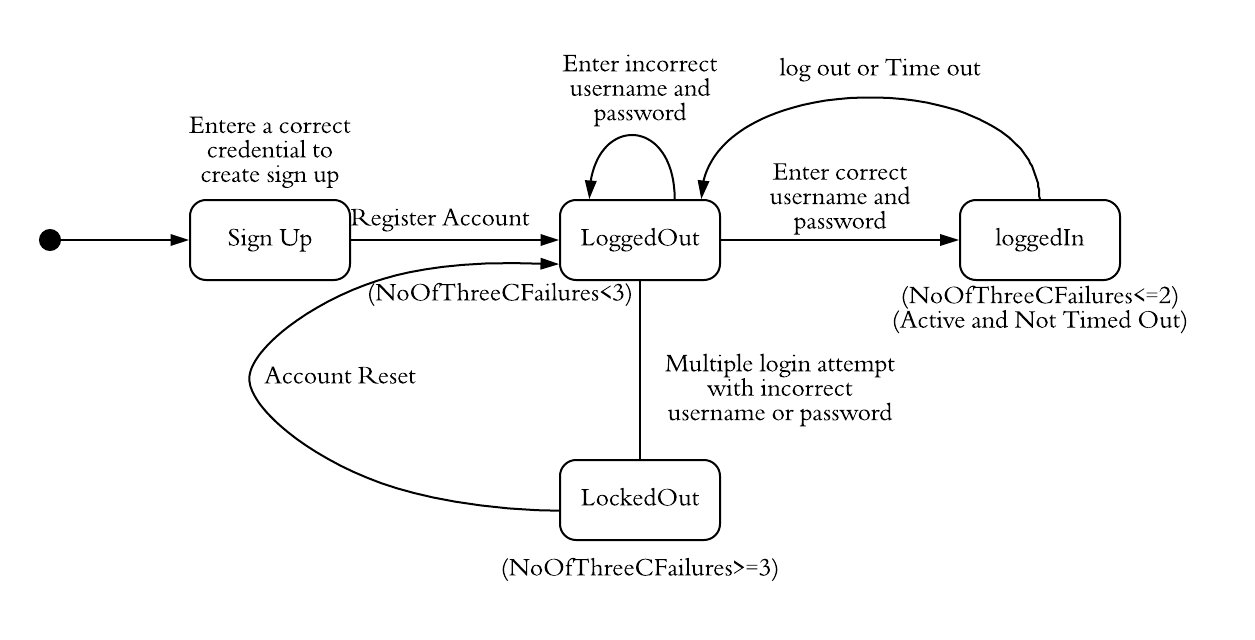
**Dynamic Model:**

**Sequences Diagram of Sign Up**



**[Fig 1: Sequence diagram of Sign Up]**

**State Diagram of UserAccount entity object**



**[Fig 2: State diagram of UserAccount]**

**3.2. A customer is able to login.**

Functionality: Customers log in their account by entering their username and password.

Functionality name: **Login**

**Functional Model:**

**Scenarios**

**Example 1:** Susan attempts to log in her account by entering her username susansmith and her password Susan@4422 on 5/01/2018 at 2:10PM. Scenario of an example above.

|  |  |
| --- | --- |
| *Use case name* | LoginSuccessful:Login |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan activates the “Login” function of the system by clicking on LOGIN. 2. Susan enters susansmith, her username, and Susan@4422, her password. 3. Susan logs in by clicking LOGIN. 4. The system shows the bank system homepage. |

**[Table: 6 Login Successful Scenario]**

**Example 2:** Susan attempts to log in her account by entering her username susansmith and her password Susan@4422 on 5/01/2018 at 2:10PM. Scenario of an example above.

|  |  |
| --- | --- |
| *Use case name* | LoginUnsuccessful:Login |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan activates the “Login” function of the system by clicking on LOGIN. 2. Susan enters susansmit, her username, and Susan@4422 , her password. 3. Susan logs in by clicking LOGIN. 4. The system shows “Login Unsuccessful”. |

**[Table: 7: Login Unsuccessful Scenario]**

**Example 3:** Susan attempts to log in her account by entering her username susansmith and her password Susan@4422 on 5/01/2018 at 2:10PM. Scenario of an example above.

|  |  |
| --- | --- |
| *Use case name* | LoginConsecutiveErrors:Login |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan activates the “Login” function of the system by clicking on LOGIN.  2. Susan enters susansmith, her username, and Susan@4422, her password.  3. Susan logs in by clicking LOGIN.  4. The system shows “3 Consecutive Errors. Please call the bank to access your account”. |

**[Table: 8 Login ConsecutiveError Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | Login |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1. The customer enters his/her Username and Password and clicks Login.  2. The system shows the customer a welcome message, the date and time of Last Login and his/her Account Summary if the Username and Password are correct or “We do not recognize your username and/or password. Please try again.” if the Username or Password are not correct or “You’re currently locked out. Please call our customer service.” If the customer got three consecutive login failures. |
| *Entry conditions* | The customer has loaded the bank’s login homepage. |
| *Exit conditions* | Login confirmed or error messages. |
| *Quality Requirements* | The login must be done in 0.01 seconds. The password has to be at least 8 characters in which at least one uppercase letter, one digit and one special character among @ $ %!#. Three consecutive entering wrong password leads to use account locked out. |

**[Table: 9 Login Initial Use Case]**

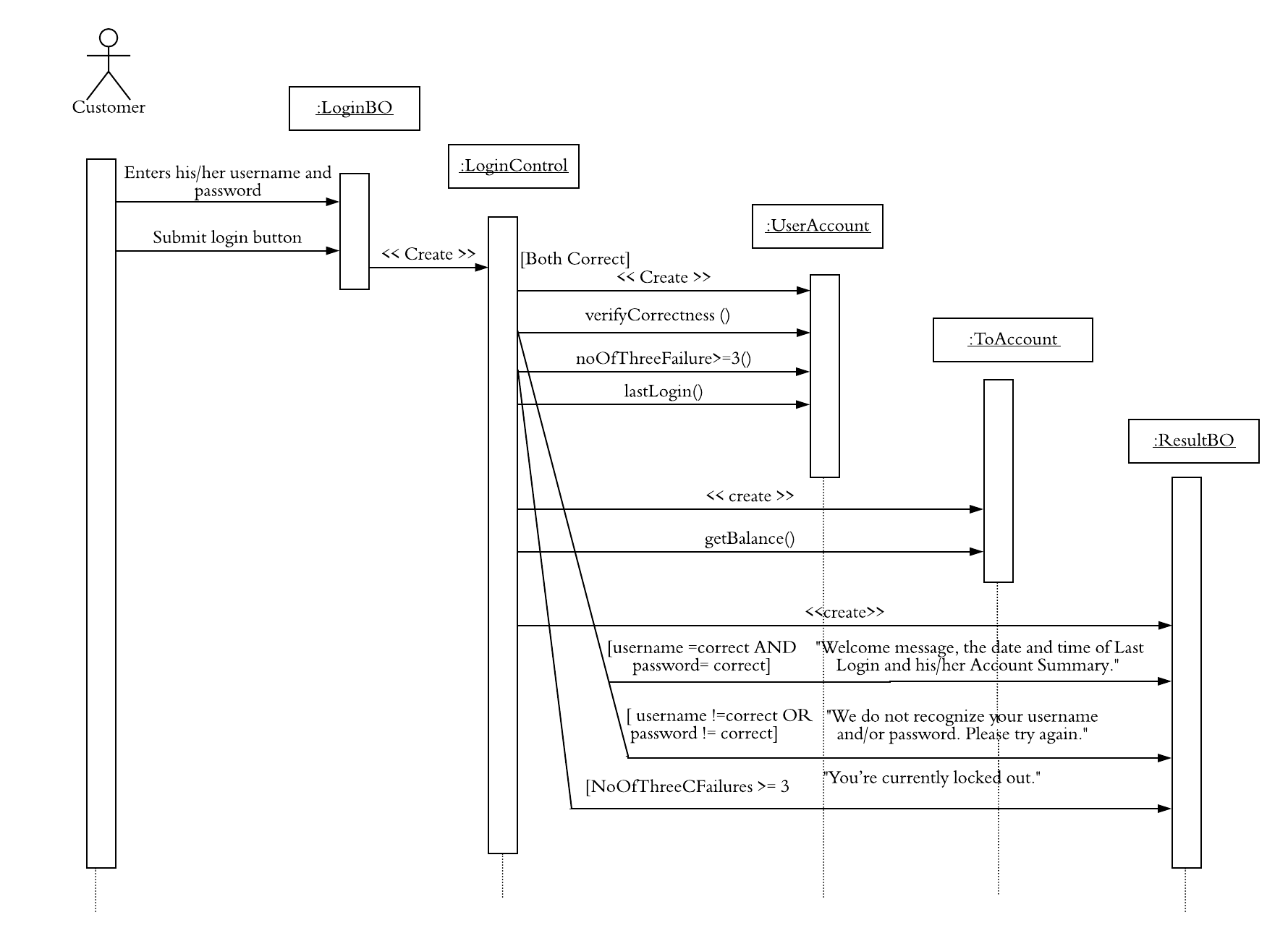
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | Login |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1. The customer enters his/her Username and Password and clicks Login in LoginBO. 2. LoginControl is created by LoginBO with passing values of Username and Password. 3. LoginControl creates UserAccount and invokes an operation to verify the correctness of Username and Password. If both are correct, LoginControl invokes an operation to check if NoOfThreeCFailures >= 3. If yes, go to Event 4 to show the locked out message; else LoginControl invokes an operation to reset NoOfThreeCFailures and an operation to update LastLogin, and creates CheckingAccount(and/or SavingsAccount) object and invokes an operation to get the balance of CheckingAccount(and/or SavingsAccount), and go to Event 4 to show login success. If only Password is not correct, LoginControl invokes an operation to increment NoOfThreeCFailures, and check if NoOfThreeCFailures >= 3. If yes, go to Event 4 to show the locked-out message; else go to Event 4 to show the message of not recognized. If only Username is not correct, go to Event 4 to show the message of not recognized 4. LoginControl creates ResultBO and shows the customer a welcome message, the date and time of Last Login and his/her Account Summary if the Username and Password are correct or “We do not recognize your username and/or password. Please try again.” if the Username or Password are not correct or “You’re currently locked out. Please call our customer service.” If the customer got three consecutive login failures. |
| *Entry conditions* | The customer has loaded the bank’s login homepage. |
| *Exit conditions* | Login confirmed or error messages. |
| *Quality Requirements* | The login must be done in 0.01 seconds. The password has to be at least 8 characters in which at least one uppercase letter, one digit and one special character among @ $ %!#. Three consecutive entering wrong password leads to use account locked out. |

**[Table: 10 Login Refine Use case]**

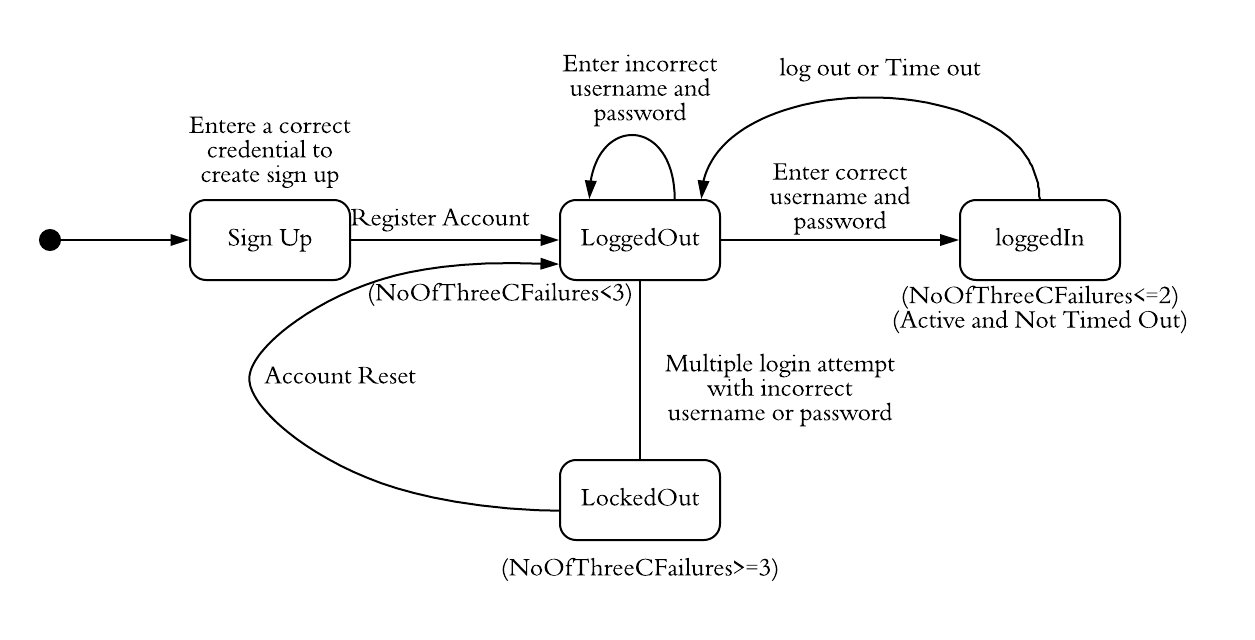
**Dynamic Model:**

**Sequences Diagram of Login**



**[Fig 3: Sequence diagram of Login]**

**State Diagram of UserAccount entity object**



**[Fig 4: State diagram of Login]**

**3.3. A customer is able to change username.**

Functionality: Customers change the username of the account.

Functionality name: **ChangeUsername**

**Functional Model:**

**Scenarios**

**Example 1:** Susan attempts to change her username of her account to susansmith on 5/01/2018 at 2:10PM. Scenario of an example above

|  |  |
| --- | --- |
| *Use case name* | SuccessfulChangeUsername:ChangeUsername |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan activates the “ChangeUsername” function of the system by clicking on CHANGE USERNAME 2. Susan enters susansmith,, a new username. 3. Susan changes the username by clicking COMPLETE. 4. The system shows “Username Changed Successfully”. |

**[Table: 11 ChangeUsername Successful Scenario]**

**Example 2:** Susan attempts to change her username of her account to susansmith on 5/01/2018 at 2:10PM. Scenario of an example above.

|  |  |
| --- | --- |
| *Use case name* | UnsuccessfulChangeUsername:ChangeUsername |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan activates the “ChangeUsername” function of the system by clicking on CHANGE USERNAME. 2. Susan enters susansmith,, a new username. 3. Susan changes the username by clicking COMPLETE. 4. The system shows “Username Changed Unsuccessfully”. |

**[Table: 12 ChangeUsername Unsuccessful Scenario]**

**Example 3:** Susan attempts to change her username of her account to ssusansmith on 5/01/2018 at 2:10PM. Scenario of an example above.

|  |  |
| --- | --- |
| *Use case name* | CancelChangeUsername:ChangeUsername |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan activates the “ChangeUsername” function of the system by clicking on CHANGE USERNAME. 2. Susan enters susansmith,, a new username. 3. Susan cancels by clicking CANCEL. 4. The system shows “Username Change Cancelled”. |

**[Table: 13 ChangeUsername Cancel Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | ChangeUsername |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1. The customer is changing his/ her username of the  account on a specific date at a specific time. 2. “Username Changed Successfully”, “Username Changed Unsuccessfully” or “Username Changed Cancelled” is shown in the middle of the screen on the same website page. |
| *Entry conditions* | The customer is logged in his/ her account and is on the  account overview page. |
| *Exit conditions* | A confirmation is shown. |
| *Quality Requirements* | The process must be done in less than 3 seconds. |

**[Table: 14 ChangeUsername Initial Use Case]**

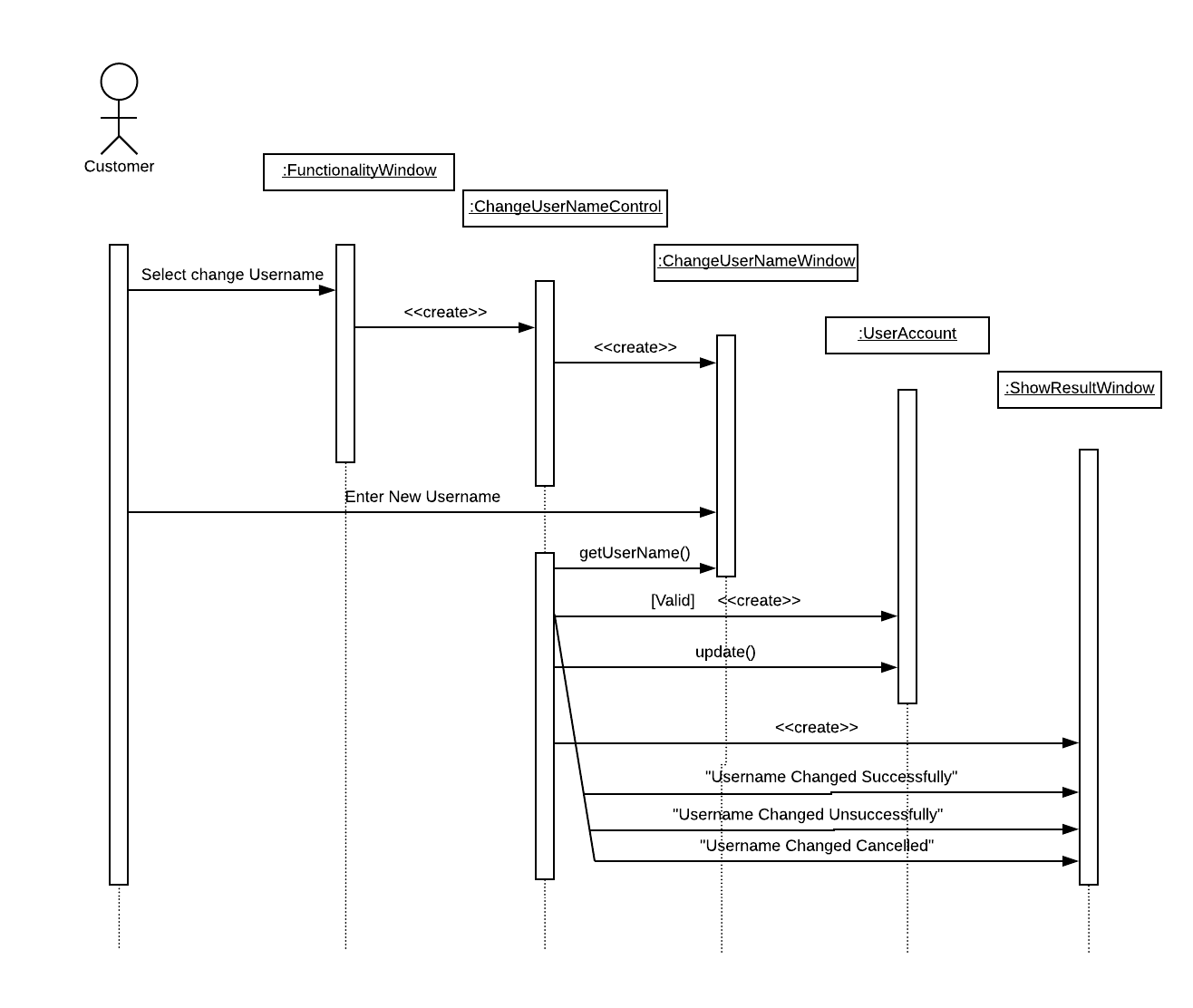
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | ChangeUsername |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1. Customer activates the “ChangeUsername” function of the system by clicking on Change Username .  2. An object of ChangeUsernameControl created by FunctionalityWindow.  3. The control object prompts the customer to enter the new  Username and clicks on submit in ChangeUsernameWindow  4.The ChangeUsernameControl object get the new Username.  If the Username is valid ChangePasswordControl object creates a UserAccount object and invokes an operation to update the Username.  5.ChangeUsernameControl object creates a ShowResultWindow to displays “Username Changed Successfully”or “Username Changed Unsuccessfully” or “Username Changed Cancelled”. |
| *Entry conditions* | The customer is logged in his/ her account and is on the  account overview page. |
| *Exit conditions* | A confirmation is shown. |
| *Quality Requirements* | The process must be done in less than 3 seconds. |

**[Table: 15 ChangeUsername Refine Use Case]**

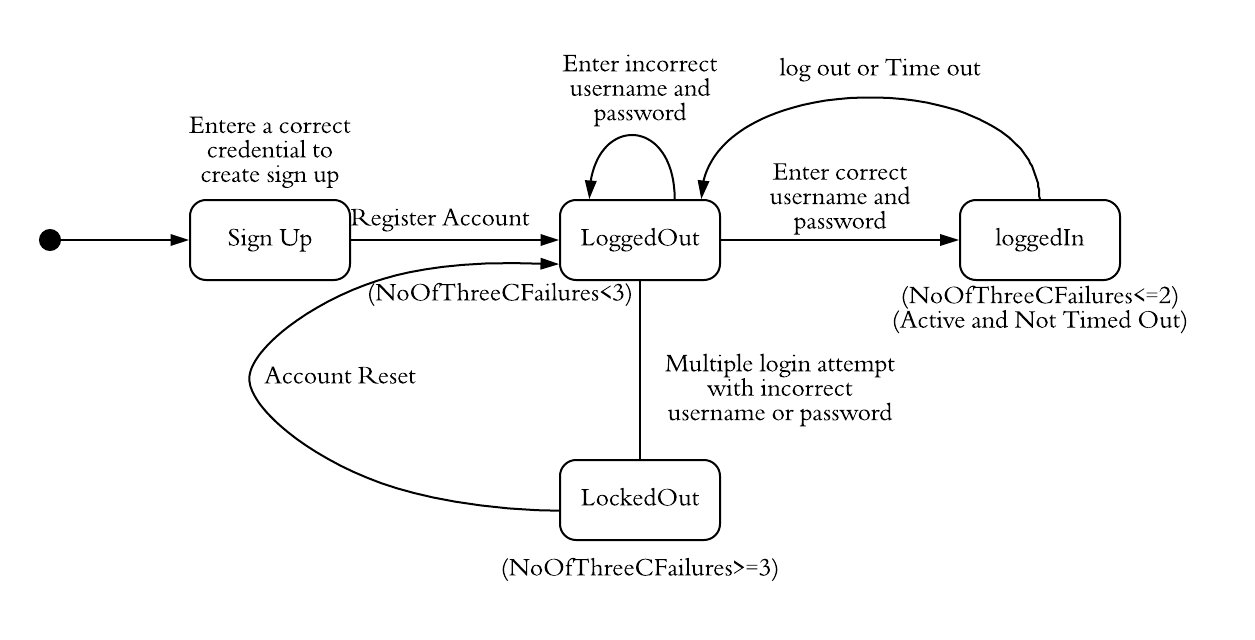
**Dynamic Model:**

**Sequences Diagram of Change Username**



**[Fig 5: Sequence diagram of Change Username]**

**State Diagram of UserAccount entity object**

**[Fig 6: State diagram UserAccount]**

**3.4. A customer is able to change password.**

Functionality: Customers change password of the account.

Functionality name: **ChangePassword**

**Functional Model:**

**Example 1:** Susan attempts to change his password (Susan@4422) to new password (Susan@5555).

**Scenarios**

|  |  |
| --- | --- |
| *Use case name* | ChangePasswordSuccess : ChangePassword |
| *Participating Actor* | Tom: Customer |
| *Flow of Events* | 1. Susan activates the “ChangePassword” function of the  system by clicking on button ChangePassword . 2. system shows “enter your old password. 3. Susan enters his old Password “Susan@4422”.  4. system shows “enter your new password. 5. Susan enters his new Password “Susan@5555”.  6. system shows “your Password changed successfully”. |

**[Table: 16 ChangePassword Successful Scenario]**

**Example 2:** Susan attempts to change his password (Susan@4422) to new password (Susan@5555).

|  |  |
| --- | --- |
| *Use case name* | UnsuccessfulChangePassword : ChangePassword |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. Susan activates the “ChangePassword” function of the system by clicking on button ChangePassword . 2. system shows “enter your old password. 3. Susan enters his old Password “Susan@4422”  4. system shows “enter your new password. 5. Susan enters his new Password “Susan@5555” 6. system shows “The Password invalid, try again”. |

**[Table: 17 ChangePassword Unsuccessful Scenario]**

**Example 3**: Susan attempts to change his password (Susan@4422) to new password (Susan@5555).

|  |  |
| --- | --- |
| *Use case name* | CancelChangePassword: ChangePassword |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. Susan activates the “ChangePassword” function of the system by clicking on button ChangePassword . 2. system shows “enter your old password. 3. Susan enters his old Password “Susan@4422” 4. system shows “enter your new password. 5. Susan cancels change password process by clicking CANCEL.  6. system shows “cancel password”. |

**[Table: 18 ChangePassword Cancel Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | ChangePassword |
| *Participating Actor* | Initiated by a customer |
| *Flow of Events* | 1. Customer activates the “ChangePassword” function of the system by clicking on button Change Password. 2. System prompts the customer to enter his old password.  3. Customer enters his old Password, and new password 4. system shows “your Password changed successfully”  or “The Password invalid, try again”, or “cancel password”. |
| *Entry conditions* | The actors are logged to the online banking System. |
| *Exit conditions* | The actors have received the confirmation of changing the  password or nothing. |
| *Quality Requirements* | Showing the confirmation of changing the password no later than 2 seconds. |

**[Table: 19 ChangPassword Initial Use Case]**

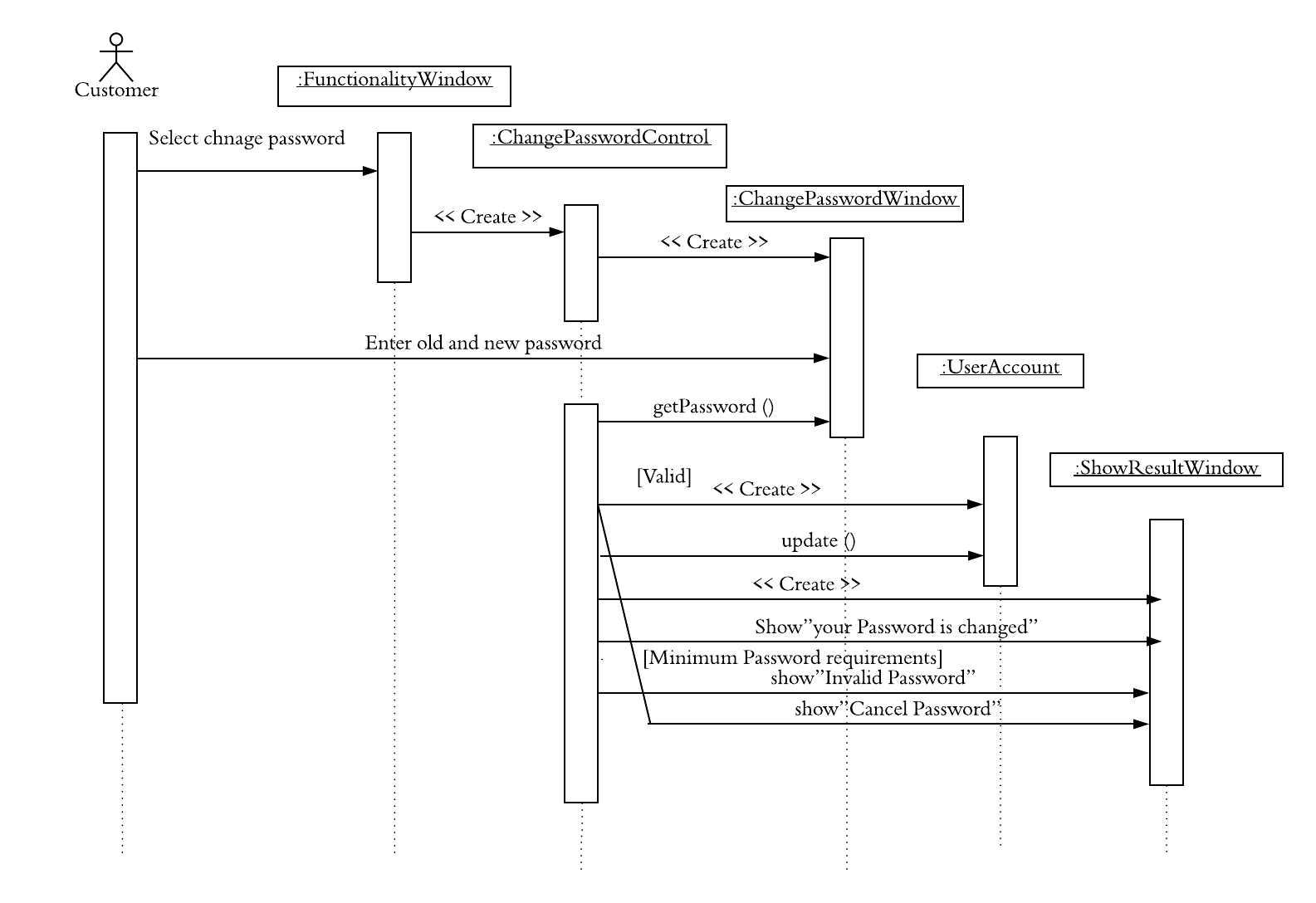
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | ChangePassword |
| *Participating Actor* | Initiated by a customer |
| *Flow of Events* | 1.Customer activates the “ChangePassword” function of the system by clicking on Change Password.  2. An object of ChangePassworControl created by FunctionalityWindow. 3. The control object prompts the customer to enter the old password and new password by ChangePasswordWindow . 4.The ChangePasswordControl get the old password and the new password. If the Password is valid, ChangePassworControl creates a ChangPassword object and invokes an operation to change the old password to new password  5. ChangePasswordControl creates a UserAccount object and invokes an operation to update the password.  6. ChangePasswordControl creates a ShowResultWindow to display “Your password is changed”," your password invalid, try again" because minimum password requirements or “cancel password”. |
| *Entry conditions* | The actors are logged to the online banking System. |
| *Exit conditions* | The actors have received the confirmation of changing the password or nothing. |
| *Quality Requirements* | Showing the confirmation of the transferring no later than 5. |

**[Table: 20 ChangePassword Refine Use Case]**

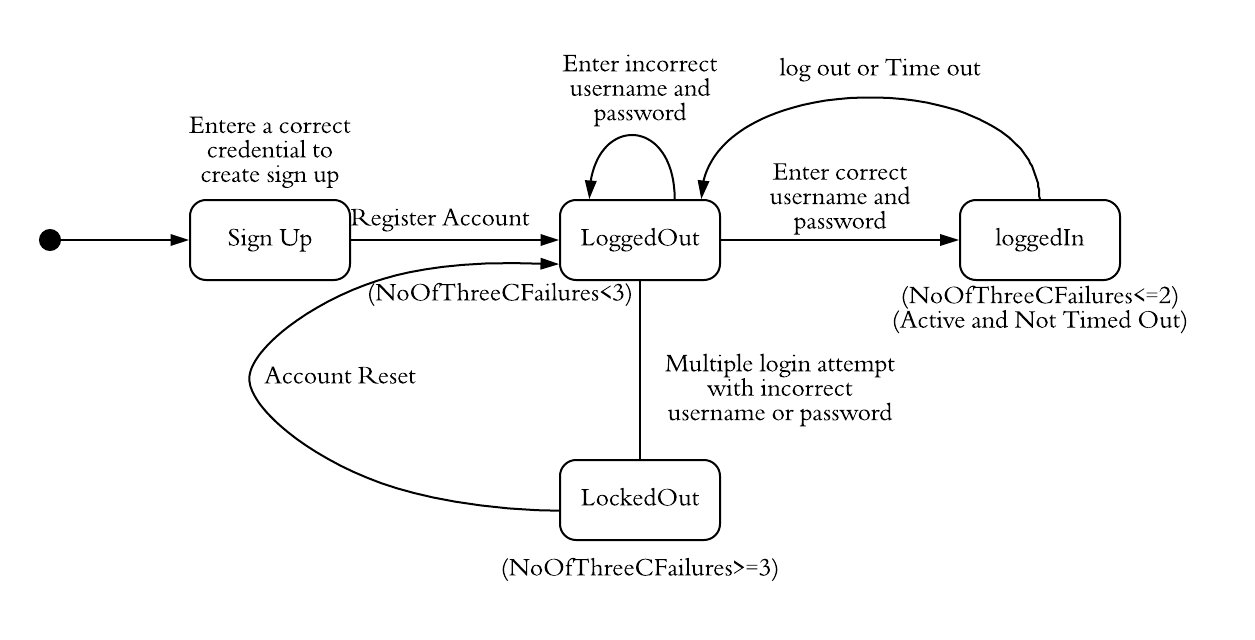
**Dynamic Model:**

**Sequences Diagram of Change Password**



**[Fig 7: Sequence diagram of Change Password]**

**State Diagram of UserAccount entity object**

**[Fig 8: State diagram of UserAccount]**

**3.5. A customer is able to transfer an amount of money from their one account to another account.**

Functionality: Customers transfer an amount of money from their one account to another account.

Functionality name**: Transfer Money**

**Functional Model:**

**Example 1:** Susan attempts to transfer 300$ from his saving account (6666) to his checking account (6600) on 02/12/2018 every month.

**Scenarios**

|  |  |
| --- | --- |
| *Use case name* | SuccessfulTransferMoney: TransferMoney |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. Susan select his saving account number 6666, his checking account number 6600, amount of money which he wants to transfer 300$, date of transfer 02/12/2018 and how many times he wants to transfer(monthly).  2.The system shows all information that entered,  “From: saving (6666)  To: checking (6600)  Amount: 300$  Date: 02/12/2018  Frequency: monthly”  3.Susan confirms the transfer information by clicking SUBMIT. 4.The system shows "your money transfer done successfully” and gives confirmation number. |

**[Table: 21 Transfer Money Successful Scenario]**

**Example 2:** Susan attempts to transfer 300$ from his saving account (6666) to his checking account (6600) on 02/12/2018 every month.

|  |  |
| --- | --- |
| *Use case name* | UnsuccessfulTransferMoney: TransferMoney |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. Susan enters his saving account number 6666, his checking account number 6600, amount of money which he wants to transfer 300$, date of transfer 02/12/2018 and how many times he wants to transfer(monthly).  2.The system shows all information that entered,  “From: saving (6666)  To: checking (6600)  Amount: 300&  Date: 02/12/2018  Frequency: monthly”  3.Susan confirms the transfer information by clicking SUBMIT.  4.The system shows "your money transfer incomplete". |

**[Table: 22 Transfer Money Unsuccessful Scenario]**

**Example 3:** Susan attempts to transfer 300$ from his saving account (6666) to his checking account (6600) on 02/12/2018 every month

|  |  |
| --- | --- |
| *Use case name* | CancelTransferMoney: TransferMoney |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. Susan enters his saving account number 6666, his checking account number 6600, amount of money which he wants to transfer 300$, date of transfer 02/12/2018 and how many times he wants to transfer(monthly).  2.The system shows all information that entered,  “From: saving (6666)  To: checking (6600)  Amount: 300&  Date: 02/12/2018  Frequency: monthly”  3.Susan confirms the transfer information by clicking CANCEL.  4.The system shows "your money transfer canceled". |

**[Table: 23 Transfer Money Cancel Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | TransferMoney. |
| *Participating Actor* | Initiated by customer. |
| *Flow of Events* | 1. Customers enter their saving account number, checking account number, amount of money to transfer, date of transfer and frequency transfer. 2.The system shows all information that entered, saving account number, checking account number, amount of money for transfer, date of transfer and frequency of transfer.  3.Customer confirms the transfer information by clicking SUBMIT or Cancels the transfer processing by clicking CANCEL.  4.The system shows” your money transfer done successfully” and gives confirmation number or shows "your money transfer incomplete” or shows "your money transfer canceled”. |
| *Entry conditions* | The actors are logged to the online banking System. |
| *Exit conditions* | The actors have received the confirmation of the transferring or nothing. |
| *Quality Requirements* | Showing the confirmation of the transferring details no later  than 10 seconds |

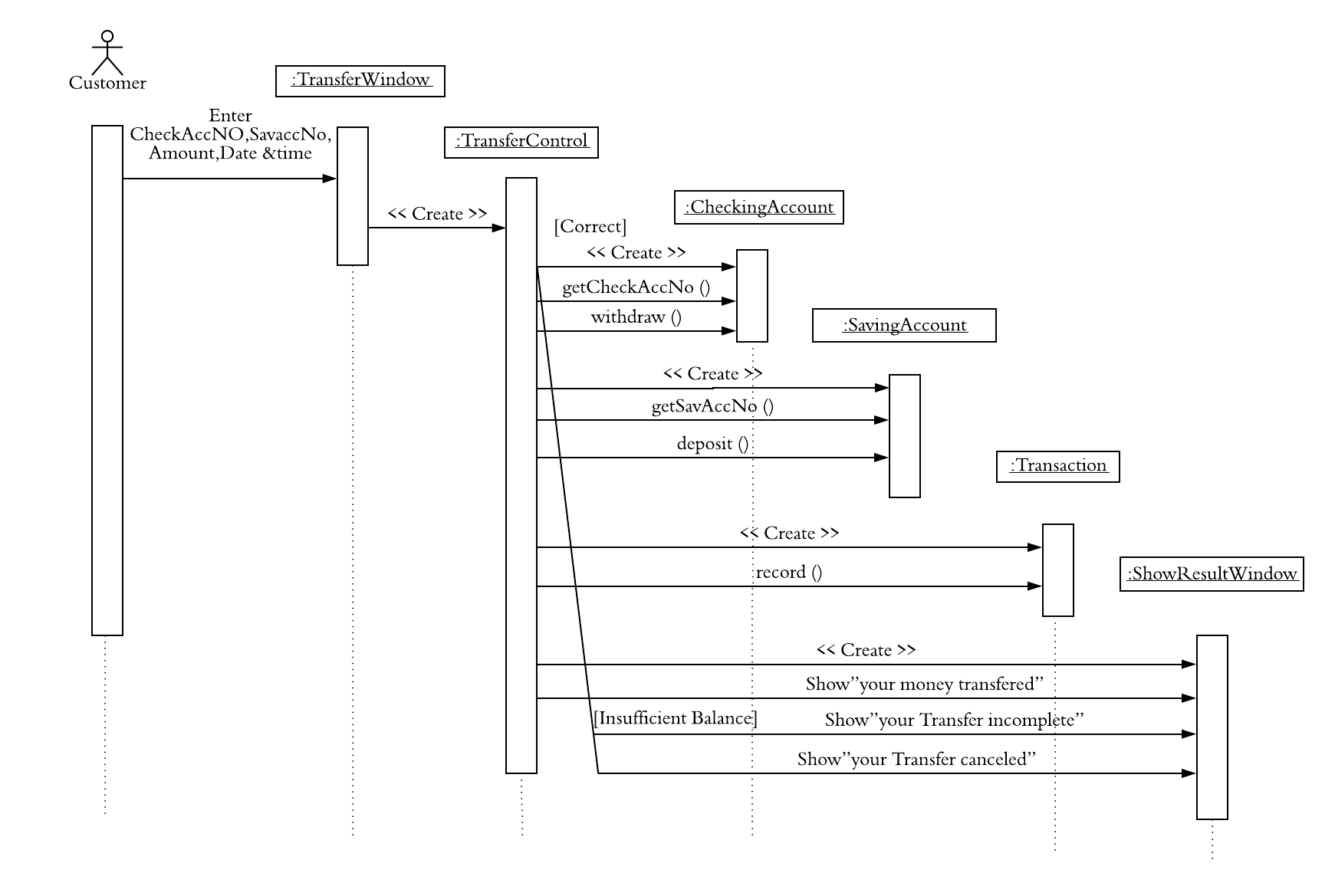
**[Table: 24 Transfer Money Initial Use Case]**

**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | TransferMoney |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1. The customers enter their saving account number checking account number, amount of money for transferring. date of transfer and how many times he wants to transfer by the TransferWindow.  2.TransferControl created by TransferWindow with entering the saving account number, checking account number, amount of money for transferring, date of transfer and how many times he wants to transfer.  3. If the data that entered is correct, TransferControl gets the customer’s checking account number and creates a CheckAccount object to invoke an operation of withdrawing the amount of money for transfer.  4. TransferControlet gets the customer’s saving account number and creates a SavAccount object and invokes an operation of put the deposit into the destination account.  5. TransferControl creates ShowResultWindow "your money transferred" and gives confirmation number,” your transfer incomplete” or” your transfer canceled”. |
| *Entry conditions* | The Customers is logged in to the Online Banking System. |
| *Exit conditions* | Online banking system shows "your money transfer done successfully" or "your money transfer incomplete" or "your money transfer canceled". |
| *Quality Requirements* | Money transfer would be process on the current business date. |

**[Table: 25 Transfer Money Refine Use Case]**

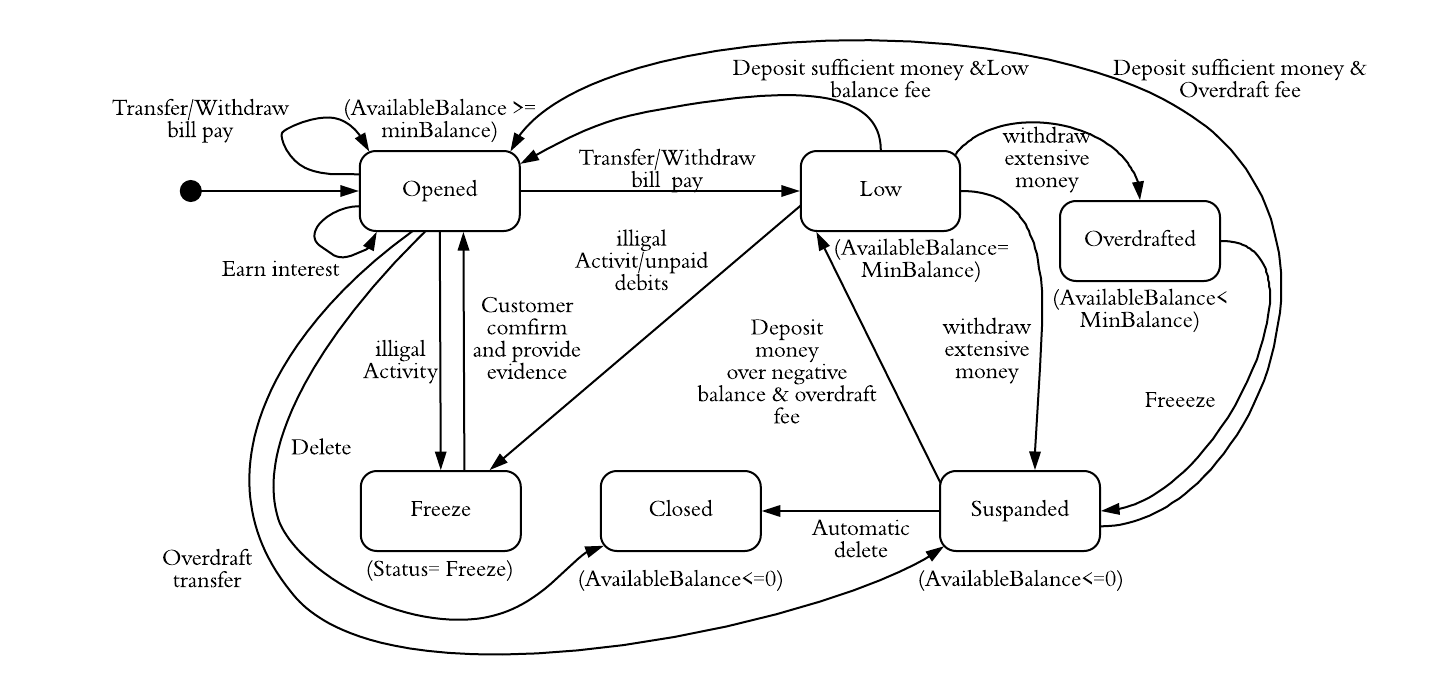
**Dynamic Model:**

**Sequences Diagram of Transfer Money**

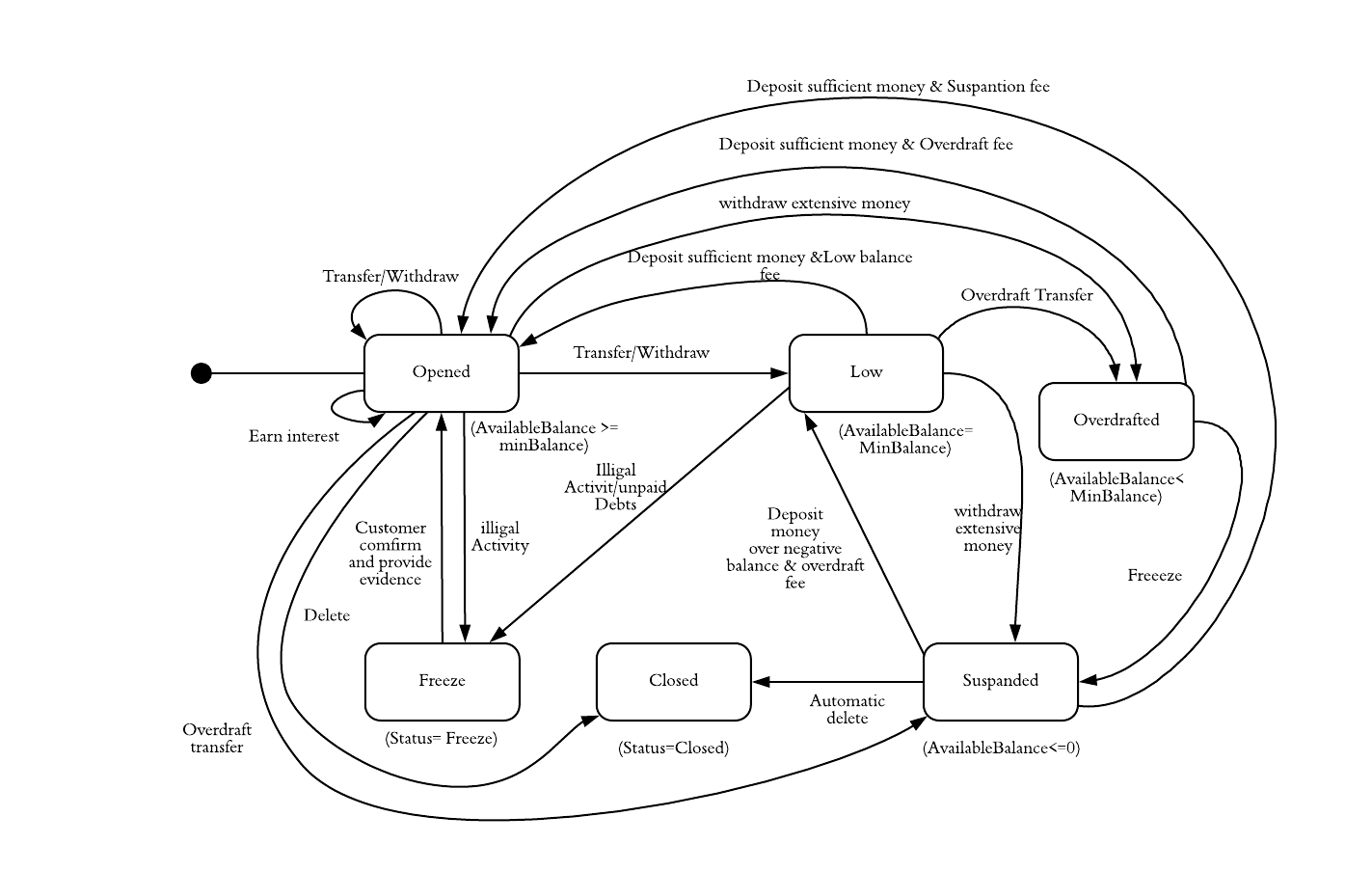
**[Fig 9: Sequence diagram of Transfer Money]**

**State Diagram**

**State diagram of CheckingAccount entity object**

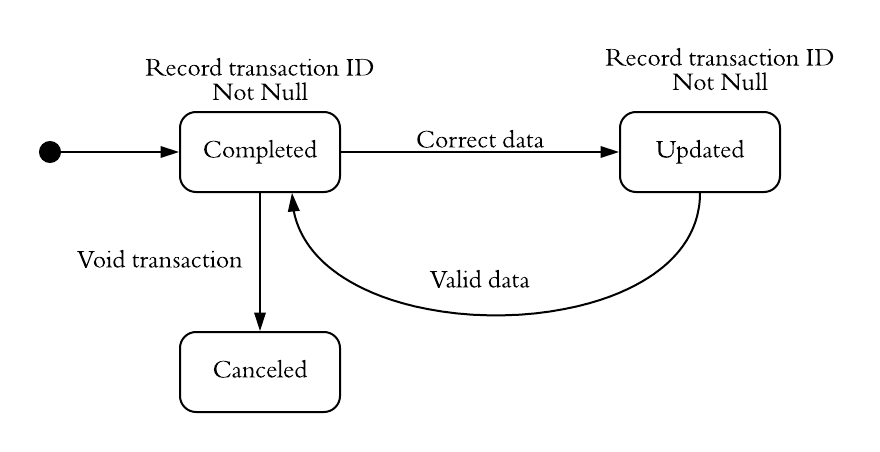
****

**[Fig 10: State diagram of CheckingAccount]**

**State diagram of SavingAccount entity object**

**[Fig 11: State diagram of SavingAccount]**

**State diagram of Transaction entity object**



**[Fig 12: State diagram of Transaction]**

**3.6. A customer is able to add a payee.**

Functionality: Customers add payee.

Functionality name**: Add Payee**

**Functional Model:**

**Scenarios**

**Example 1**: Susan attempt to add a Verizon as a payee which is in Paramus it’s zip code 07652 and her Account Number with Verizon 12345.

|  |  |
| --- | --- |
| *Use case name* | SuccessfulAddPayee:AddPayee |
| *Participating Actor* | Susan:Customer |
| *Flow of Events* | 1. Susan enters a payee name” Verizon”, payee address “Paramus, NJ", payee zip code “07655” and her Account Number with this payee “12345” and clicks "Add payee" button. 2. The system review all information that entered,  payee name:” Verizon”  Payee address: “Paramus, NJ”  payee zip code: “07655”  her Account Number with this payee: “12345”  3. Susan confirm the information by clicking SUBMIT.  4. The system shows "The payee is successful added". |

**[Table: 26 Add Payee Successful Scenario]**

**Example 2**: Susan attempt to add a Verizon as a payee which is in Paramus it’s zip code 07652 and her Account Number with Verizon 12345.

|  |  |
| --- | --- |
| *Use case name* | UnsuccessfulAddPayee: AddPayee |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. Susan enters a payee name” Verizon”, payee address “Paramus, NJ", payee zip code “07655” and her Account Number with this payee “12345” and clicks "Add payee" button. 2. The system reviews all information that entered,  payee name:” Verizon”  Payee address: “Paramus, NJ”  payee zip code: “07655”  her Account Number with this payee: “12345”  3. Susan confirm the information by clicking SUBMIT.  4. The system shows "The payee unsuccessful added, Try again". |

**[Table: 27 Add Payee Unsuccessful Scenario]**

**Example 3**: Susan attempt to add a Verizon as a payee which is in Paramus it’s zip code 07652 and her Account Number with Verizon 12345.

|  |  |
| --- | --- |
| *Use case name* | CancelAddPayee: AddPayee |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. Susan enters a payee name” Verizon”, payee address “Paramus, NJ", payee zip code “07655” and her Account Number with this payee “12345” clicks "Add payee" button. 2. The system reviews all information that entered,  payee name: " Verizon”  Payee address: “Paramus, NJ”  payee zip code: “07655”  her Account Number with this payee: “12345”  3. Susan cancels the operation by clicking CANCEL. 4. The system goes to the home page. |

**[Table: 28 Add Payee Cancel Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | AddPayee |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1.The Customer enters a payee name, payee address, payee zip code and his/her Account Number with this payee. 2. The system review all information that entered payee name, Payee address, payee zip code and customer’s Account Number with this payee  3. Customer confirm the information that entered by clicking SUBMIT or cancels the operation by clicking CANCEL. 4. The system shows "The payee successful added”,” The payee unsuccessful  added, try again” or goes to the home page. |

**[Table: 29 Add Payee Initial Use Case]**

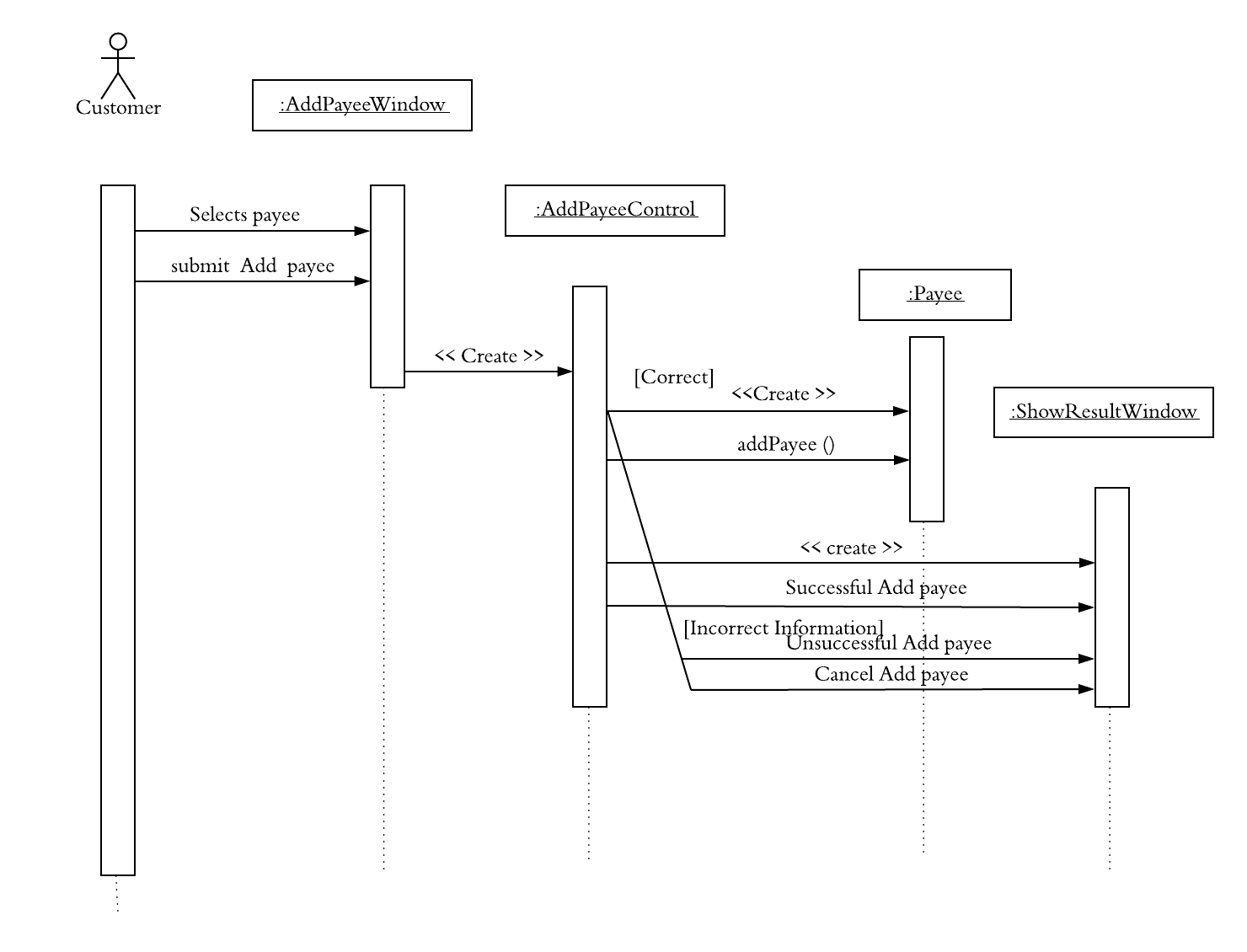
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | AddPayee |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1. Customer activate the "AddPayee" in AddPayeeWindow by entering payee name, payee address, payee zip code, account number and then customer click "Submit" button. 2. The AddPayeeControl object is created by receiving all the information from AddPayeeWindow. 3. If the information is correct, the control object creates entity object Payee. 4. The AddPayeeControl object invokes the operation AddPayee. 5. AddPayeeControl object display "successful Add payee", "unsuccessful Add payee because of incorrect information " or "cancel Add payee" in ShowResultWindow. |
| *Entry conditions* | Customer has added payee. |
| *Exit conditions* | Online banking system shows "Successful add payee" or "Unsuccessfully add payee" or “Canceled add payee". |
| *Quality Requirements* | Add payee must be done within 3 minutes. |

**[Table: 30 Add Payee Refine Use Case]**

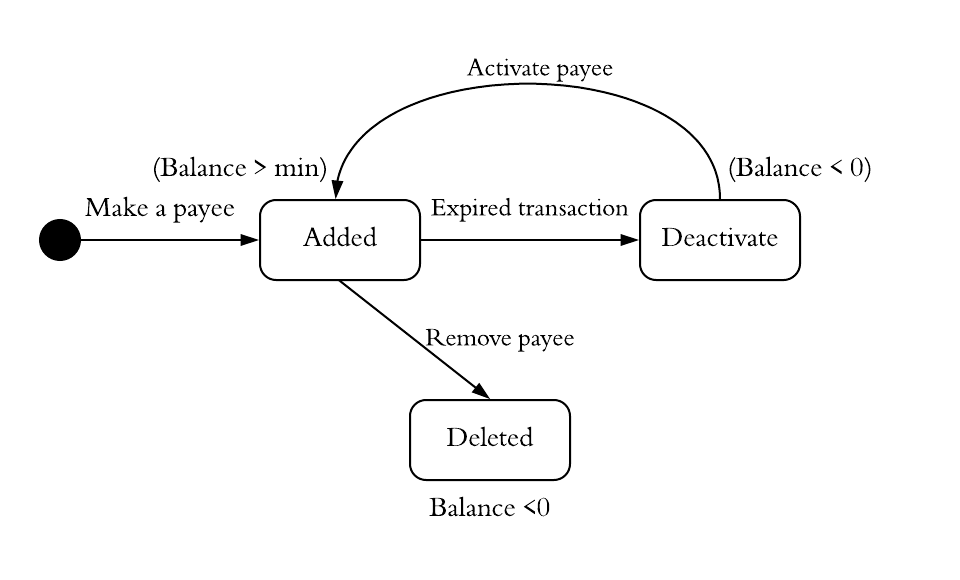
**Dynamic Model:**

**Sequences Diagram**



**[Fig 13: Sequence diagram of Add Payee]**

**State Diagram of Payee entity object**



**[Fig 14: State diagram of Payee]**

**3.7. A customer is able to delete a payee.**

Functionality: Customers delete a payee.

Functionality name**: Delete Payee**

**Functional Model:**

**Scenarios**

**Example 1:** Susan attempts to delete her payee by clicking "Delete Payee" on 04/12/2018 at 8:30PM.

|  |  |
| --- | --- |
| *Scenario name* | DeletePayeeSuccessful:DeletePayee |
| *Participating actor* | Susan:Customer |
| *Instances Flow of events* | 1.Susan activates the “Delete Payee” functionality of the system by clicking on “Delete Payee”.  2.Susan selects her Payee "Verizon", then Susan selects "View/Change payee details".  3.Susan clicks "Delete payee" button to delete payee.  4. Payee deleted successful. |

**[Table: 31 Delete Payee Successful Scenario]**

**Example 2:** Susan attempts to delete her payee by clicking "Delete Payee" on 04/12/2018 at 8:30PM.

|  |  |
| --- | --- |
| *Scenario name* | DeletePayeeUnsuccessful:DeletePayee |
| *Participating actor* | Susan:Customer |
| *Instances Flow of events* | 1.Susan activates the “Delete Payee” functionality of the system by clicking on “Delete Payee”.  2.Susan selects her Payee "Verizon", then Susan selects "View/Change payee details".  3.Susan clicks "Delete payee" button to delete payee.  4.Payee deleted unsuccessful. |

**[Table: 32 Delete Payee Unsuccessful Scenario]**

**Example 3:** Susan attempts to cancel delete payee by clicking "Cancel" on 04/12/2018 at 8:30PM.

|  |  |
| --- | --- |
| *Scenario name* | DeletePayeeCancel:DeletePayee |
| *Participating actor* | Susan:Customer |
| *Instances Flow of events* | 1.Susan activates the “Delete Payee” functionality of the system by clicking on “Delete Payee”.  2.Susan selects her Payee "Verizon", then Susan selects "View/Change payee details".  3.Susan clicks "Cancel Delete payee" button to don’t delete payee.  4.Delete payee canceled. |

**[Table: 33 Delete Payee Cancel Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | DeletePayee |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1.The customer goes to Bill Payment tab, selects his/her payee "Verizon" and selects "View/Change payee details".  2.The customer clicks on "Delete payee" button.  3.Delete Payee was "Successfully", "Unsuccessfully" or "Canceled". |
| *Entry conditions* | Customer has logged in the system. |
| *Exit conditions* | Online banking system shows "Successful delete payee" or "Unsuccessfully delete payee" or “Canceled delete payee". |
| *Quality Requirements* | Delete payee must be done within 2 minutes. |

**[Table: 34 Delete Payee Initial Use Case]**

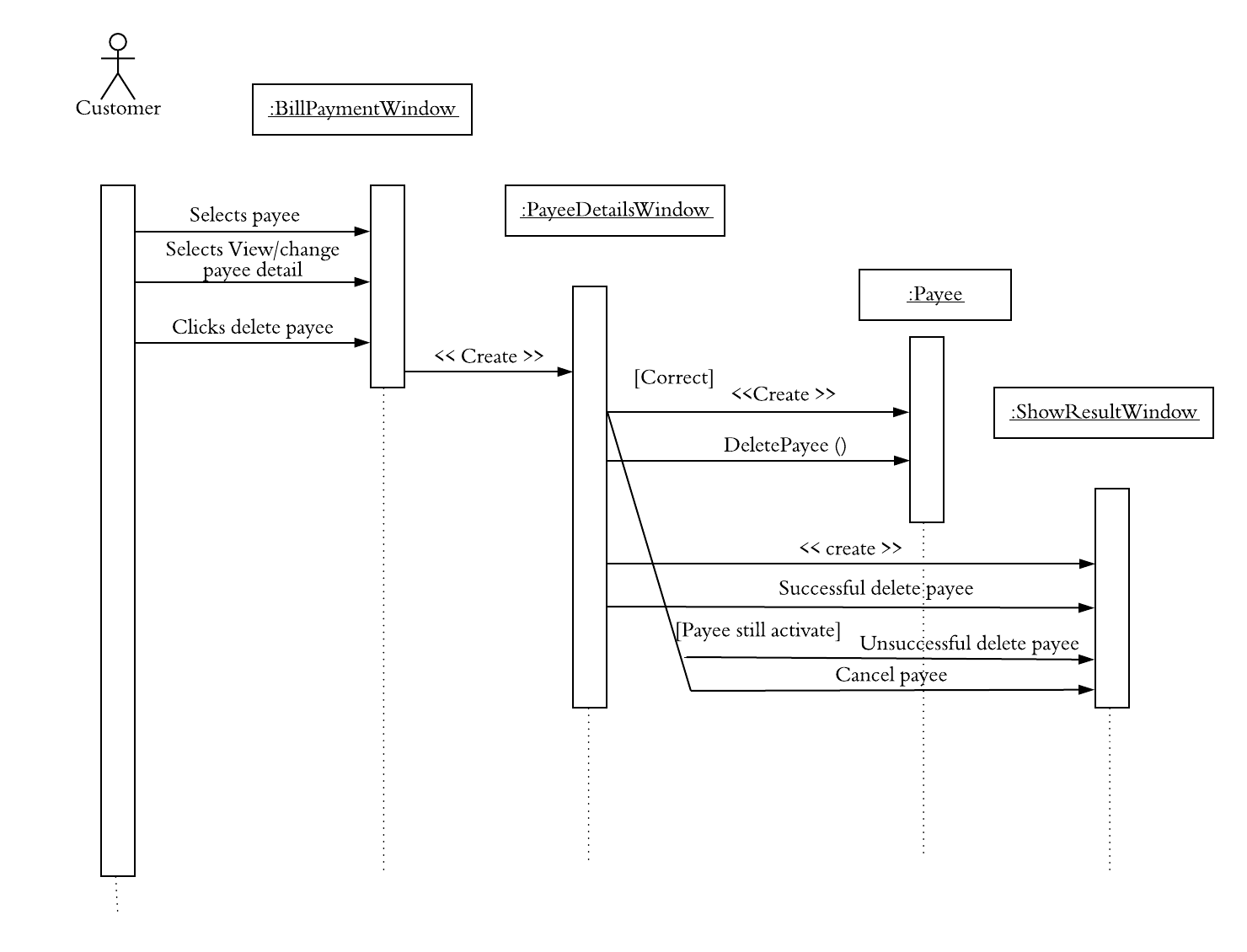
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | DeletePayee |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1.The customer is deleting a payee by selectsSlects payee, select View/change payee details and clicks “Delete Payee” button in BillPaymentWindow .  2.If the entered information is correct, The PayeeDetailsControl object is created.  3.The PayeeDetailsControl object receive all the details which customer entered. The PayeeDetailsControl object creates Payment entity object and invoke the operation deletePayee for delete payee.  4.After the invocation of the delete payee operation, the PayeeDetailsControl object displays “Successful delete payee” upon successful completion of the delete payee or “Unsuccessful delete payee” due to payee still active of the delete payee or “Cancel delete payee” in ShowResultWindow. |
| *Entry conditions* | Customer has logged in the system. |
| *Exit conditions* | Online banking system shows "Successful delete payee" or "Unsuccessfully delete payee" or “Canceled delete payee". |
| *Quality Requirements* | Delete payee must be done within 2 minutes. |

**[Table: 35 Delete Payee Refine Use Case]**

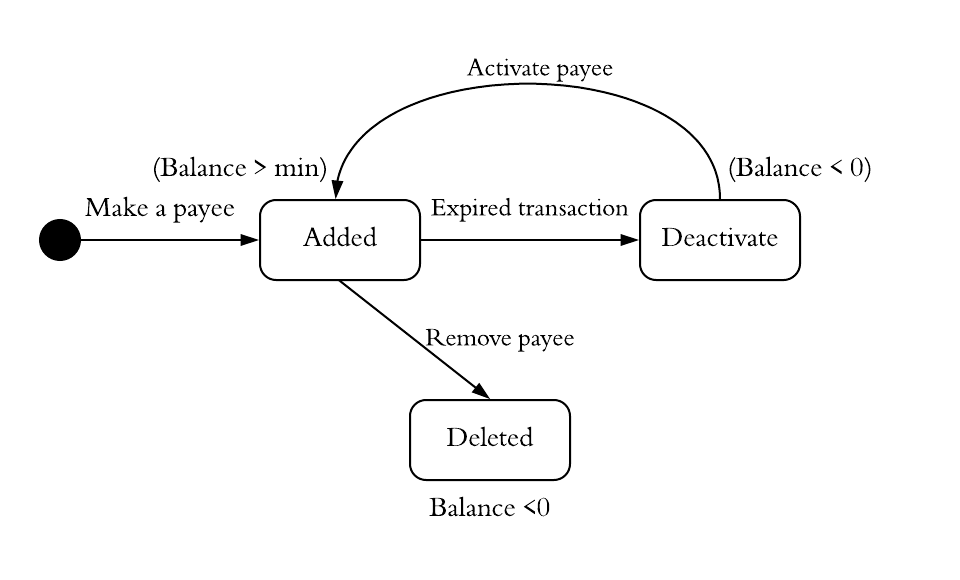
**Dynamic Model:**

**Sequences Diagram of Delete Payee**



**[Fig 15: Sequence diagram of Delete Payee]**

**State Diagram of Payee entity object**



**[Fig 16: State diagram of Payee]**

**3.8. A customer is able to schedule a payment that is paid either from the customer’s savings account or checking account.**

Functionality: Customers Schedule payment that is paid either from the customer's savings account or checking account.

Functionality name**: Schedule Payment**

**Functional Model:**

**Scenarios**

**Example 1:** Susan attempt to schedule a payment by entering amount "$200" and date "04/28/2018" at 9 PM.

|  |  |
| --- | --- |
| *Scenario name* | SchedulePaymentSuccessful: SchedulePayment |
| *Participating actor* | Susan: Customer |
| *Instances Flow of events* | 1.Susan activates the “Schedule Payment” functionality of the system by clicking on “Schedule Payment”.  2. Susan select her checking account or saving account, payee account “234567891”, enters a payee amount "$200". Susan select the date for the payment "04/28/2018".  3. The system review all information that entered, payee account “234567891”, enters a payee amount "$200". Susan select the date for the payment "04/28/2018”  4. Susan click on "Schedule Payment" button.  5.Displays the successful schedule payment. |

**[Table: 36 Schedule Payment Successful Scenario]**

**Example 2:** Susan attempt to schedule a payment by entering amount "$5000" and date "04/28/2018" at 9:30 PM.

|  |  |
| --- | --- |
| *Scenario name* | SchedulePaymentUnsuccessful: SchedulePayment |
| *Participating actor* | Susan: Customer |
| *Instances Flow of events* | 1.Susan activates the “Schedule Payment” functionality of the system by clicking on “Schedule Payment”.  2.Susan select her checking account or saving account, payee account “234567891”, enters a payee amount "$2000". Susan select the date for the payment "04/28/2018".  3.The system review all information that entered, payee account “234567891”, enters a payee amount "$2000". Susan select the date for the payment "04/28/2018”  4.Susan click on "Schedule Payment" button.  5.Displays the unsuccessful schedule payment because of her/his insufficient balance. |

**[Table: 37 Schedule Payment Unsuccessful Scenario]**

**Example 3:** Susan attempt to schedule a payment by entering amount "$200" and date "04/28/2018" at 9 PM.

|  |  |
| --- | --- |
| *Scenario name* | SchedulePaymentCancel: SchedulePayment |
| *Participating actor* | Susan: Customer |
| *Instances Flow of events* | 1.Susan activates the “Schedule Payment” functionality of the system by clicking on “Schedule Payment”.  2.Susan select her checking account or saving account, payee account “234567891”, enters a payee amount "$200". Susan select the date for the payment "04/28/2018".  3.The system review all information that entered, payee account “234567891”, enters a payee amount "$2000". Susan select the date for the payment "04/28/2018”.  4.Susan clicks on cancel payment button.  5.Displays canceled payment message. |

**[Table: 38 Schedule Payment Cancel Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | SchedulePayment |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1.A customer goes to Payment tab, selects her checking account or saving account, enter a payee account number, enter a payee amount and the date to schedule payment.  3.The system review all information that entered, payee account, enters a payee amount and select the date for the payment.  3. customer clicks "Schedule Payment" button.  4. Online banking system displays successful, unsuccessful or canceled message. |
| *Entry conditions* | Customer has scheduled payment. |
| *Exit conditions* | Online banking system shows "Successful schedule  payment" or "Unsuccessfully schedule payment" or “Canceled schedule payment". |
| *Quality Requirements* | Schedule payment must be done within 3 minutes. |

**[Table: 39 Schedule Payment Initial Use Case]**

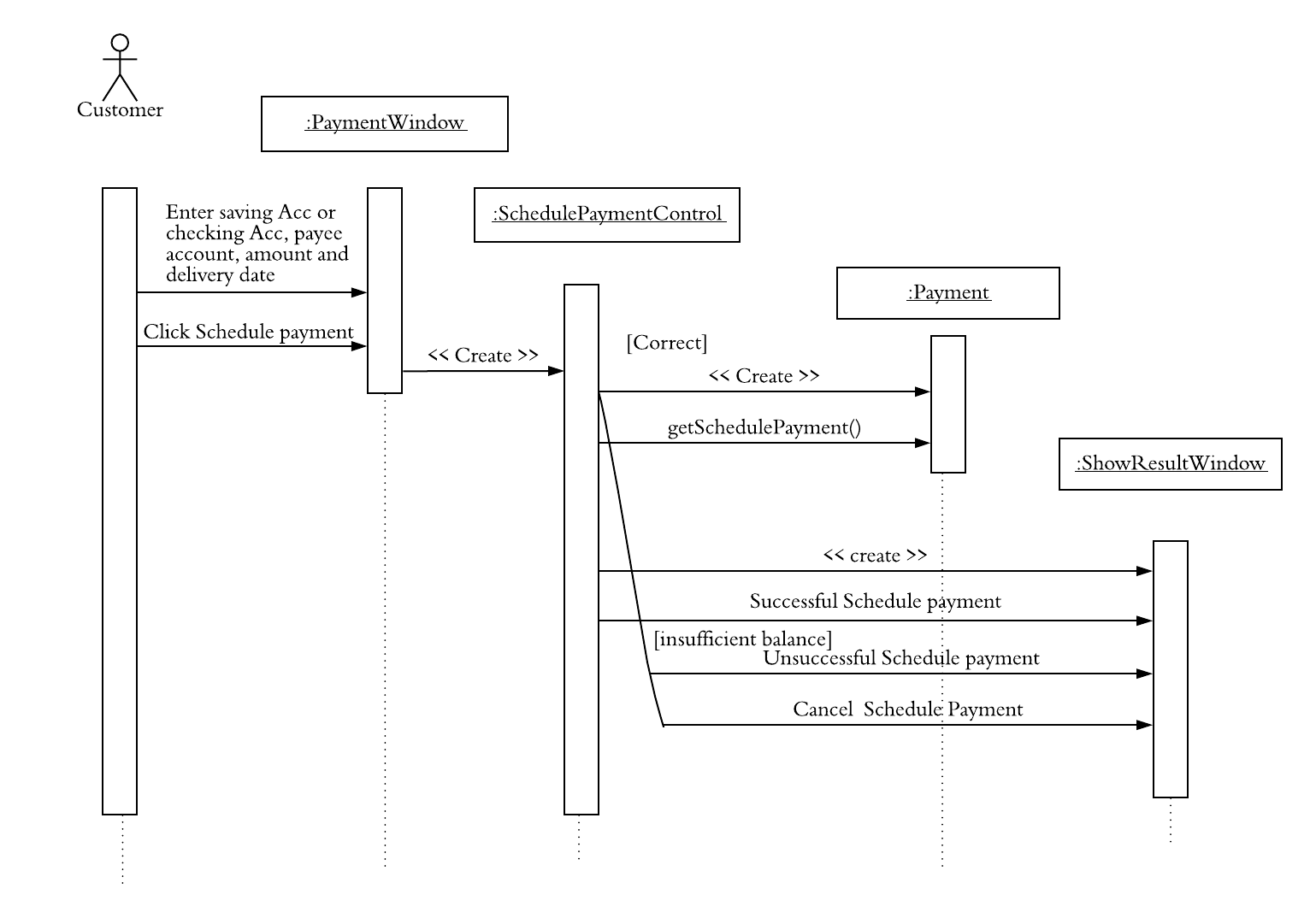
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | SchedulePayment |
| *Participating Actor* | Initiated by the customer |
| *Flow of Events* | 1. The customer is scheduling payment by select his/her savingAcc or checkingAcc , PayeeAccount, Amount and DeliveryDate and clicks on “schedule Payment” button in PaymentWindow.  2. If entered information is correct, The SchedulePaymentControl object is created.  3.The SchedulePaymentControl object receives all the details which customer entered. The SchedulePaymentControl object creates Payment entity objects..The SchedulePaymentControl object invokes the operation schedulePayment for schedule the payment.  4. After the invocation of the schedule payment operation, the SchedulePaymentControl object displays “Successful schedule payment” upon successful completion of the schedule payment operation or “Unsuccessful schedule payment” due to payer account has insufficient balance or “cancel scheduled payment” in ShowResultWindow. |
| *Entry conditions* | The customer is logged into his/her Online Banking System. |
| *Exit conditions* | Online banking system shows "Successful schedule payment" or "Unsuccessfully schedule payment" or “Canceled schedule payment". |
| *Quality Requirements* | The Process must be done in less than 3 seconds. |

**[Table: 40 Schedule Payment Refine Use Case]**

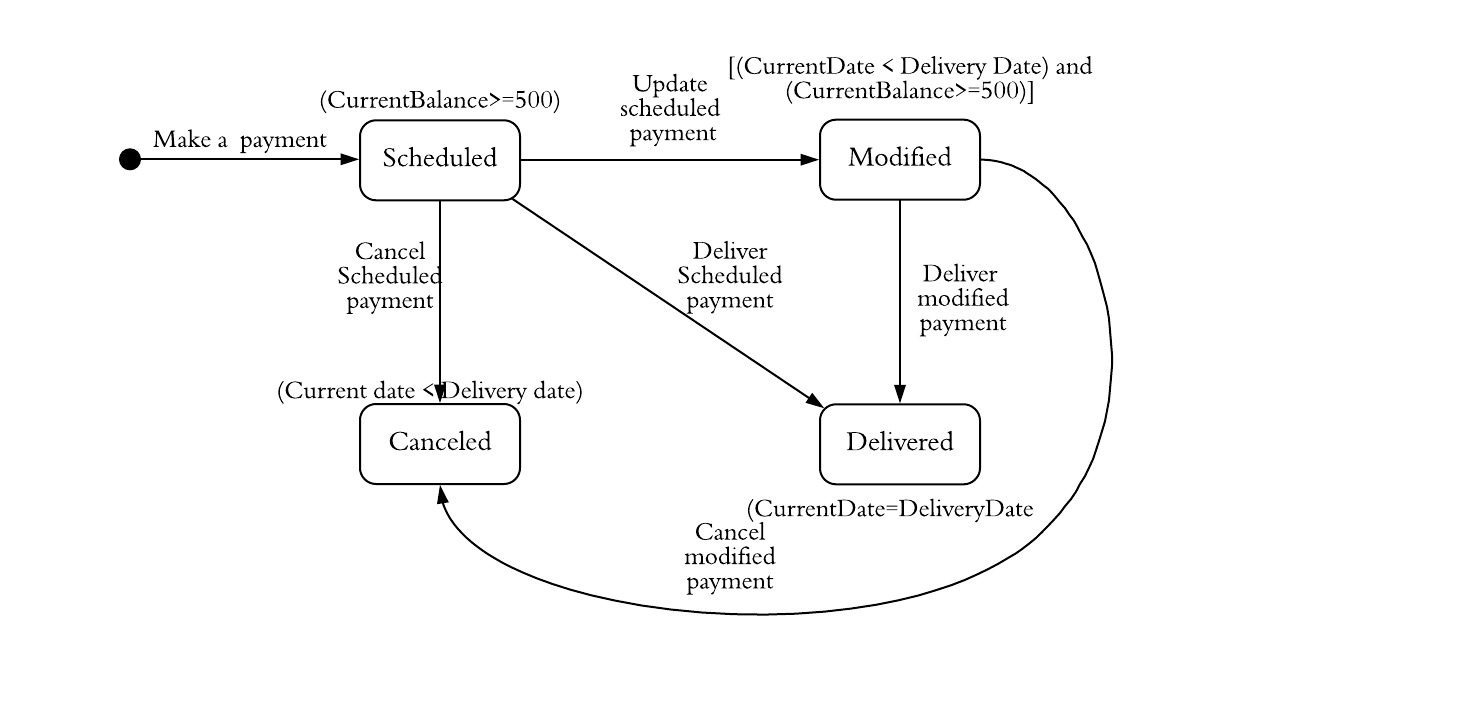
**Dynamic Model:**

**Sequences Diagram of Schedule Payment**



**[Fig 17: Sequence diagram of Schedule Payment]**

**State Diagram of Payment entity object**

**[Fig 18: State diagram of Payment]**

**3.9. A customer is able to modify a payment before its delivery date, such as change payment amount or delivery date.**

Functionality: Customers modify a payment before its delivery date, such as change payment amount or delivery date.

Functionality name**: Modify Payment**

**Functional Model:**

**Scenarios**

**Example 1:** Susan attempt to modify a payment by entering amount "$200" and date "04/29/2018" at 9 PM.

|  |  |
| --- | --- |
| *Scenario name* | ModifyPaymentSuccessful:ModifyPayment |
| *Participating actor* | Susan:Customer |
| *Instances Flow of events* | 1.Susan activates the “Modify Schedule Payment” functionality of the system by clicking on “Schedule Payment”.  2.Susan enters a payee amount "$200" and selects the date for the payment "04/29/2018" and clicks "Modify Payment" button.  3.Displays successful modify payment changes |

**[Table: 41 Modify Payment Successful Scenario]**

**Example 2:** Susan attempt to modify a payment by entering amount "$25000" and date "4/29/2018" at 9:15 PM.

|  |  |
| --- | --- |
| *Scenario name* | ModifyPaymentUnsuccessful: ModifyPayment |
| *Participating actor* | Susan:Customer |
| *Instances Flow of events* | 1.Susan activates the “Modify Schedule Payment". Functionality of the system by clicking on “Schedule Payment”.  2. Susan enters a payee amount "$25000" and selects the date for the payment "04/29/2018" and clicks "Modify Payment" button.  3. Displays unsuccessful modify payment changes due to exceeded amount per transaction. |

**[Table: 42 Modify Payment Unsuccessful Scenario]**

**Example 3:** Susan attempt to modify a payment by entering amount "$25000" and date "04/29/2018" at 9 PM.

|  |  |
| --- | --- |
| *Scenario name* | ModifyPaymentCancel:ModifyPayment |
| *Participating actor* | Susan:Customer |
| *Instances Flow of events* | 1.Susan activates the “Modify Schedule Payment” functionality of the system by clicking on “Schedule Payment”.  2.Susan enters a payee amount "$25000" and selects the date for the payment "04/29/2018" and clicks "Cancel Modify Payment" button.  3.Displays canceled modify payment message. |

**[Table: 43 Modify Payment Cancel Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | ModifyPayment |
| *Participating Actor* | Initiated by a Customer |
| *Flow of Events* | 1. The customer is modifying a payment before its delivery date, such as change payment amount or delivery date, on a specific date at a specific time. 2. “Payment Modified Successfully”, “Payment Modified Unsuccessfully” or “Payment Modified Cancelled” is shown in the middle of the screen on the same website page. |
| *Entry conditions* | The customer is logged in his/ her account and is on the account overview page. |
| *Exit conditions* | A confirmation is shown. |
| *Quality Requirements* | The process must be done in less than 3 seconds |

**[Table: 44 Modify Payment Initial Use Case]**

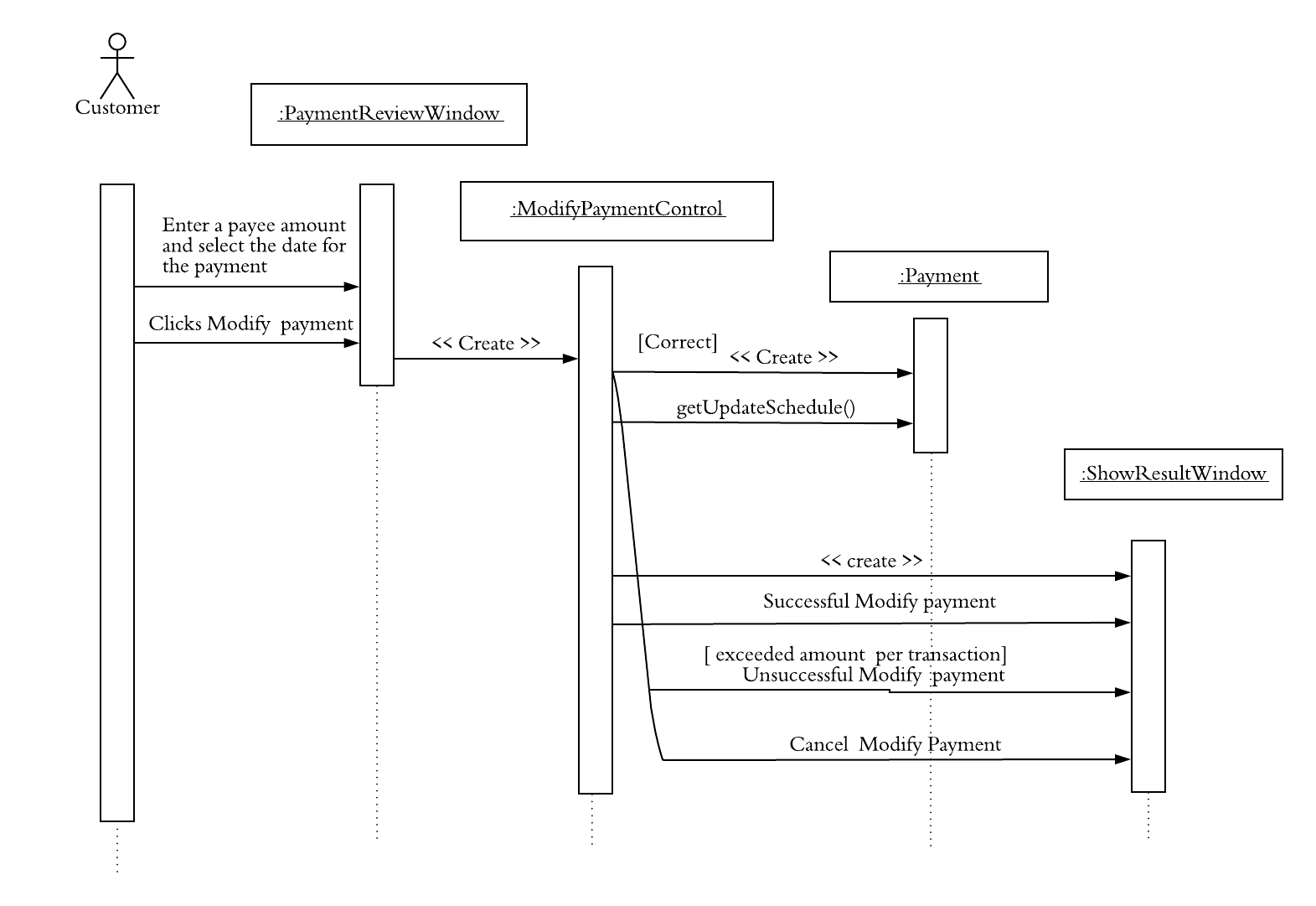
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | ModifySchedulePayment |
| *Participating Actor* | Initiated by the customer |
| *Flow of Events* | 1.The customer is updating his/her scheduled payment on a specific day and at specific time and clicks on “Modify payment” button in PaymentReviewWindow.  2. If entered information is correct, The ModifyPaymentControl object is created.  3.The ModifyPaymentControl object creates Payment entity object,the ModifyPaymentControl object invokes the updateSchedule operation to modify scheduled payment.  4. After the invocation of the updateSchedule operation, the ModifyPaymentControl object display “Successful modify payment” upon successful operation or “Unsuccessful modify payment” upon exceeded amount per transaction or “Canceled modify payment” in the ShowResultWindow.. |
| *Entry conditions* | The customer is logged into his/her Online Banking System. |
| *Exit conditions* | The online banking system shows “successfully modify the scheduled payment” or “Unsuccessful modify the scheduled payment”. |
| *Quality Requirements* | The Process must be done in less than 3 seconds. |

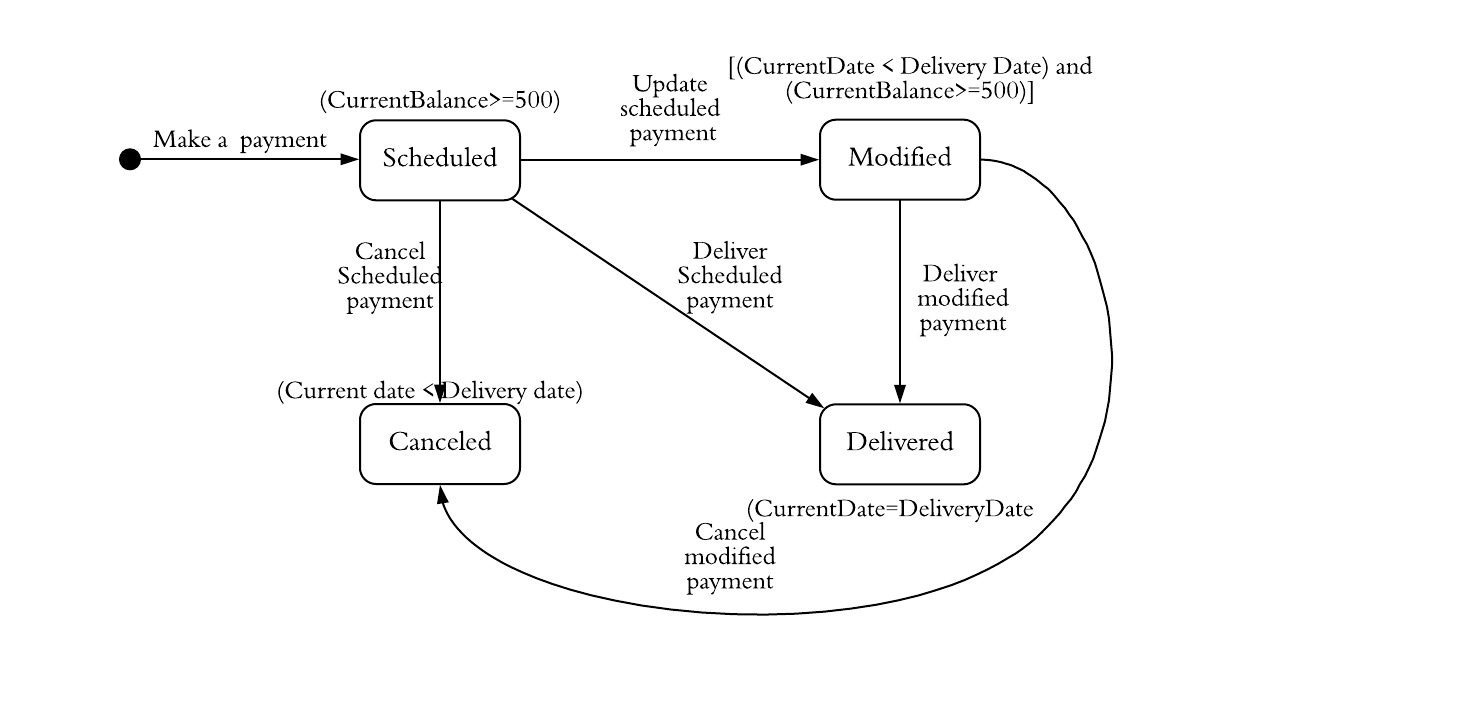
**[Table: 45 Modify Payment Refine Use Case]**

**Dynamic Model:**

**Sequences Diagram of Modify Payement**

**[Fig 19: Sequence diagram of Modify Payment]**

**State Diagram of Payment entity object**

**[Fig 20: State diagram of Payment]**

**3.10. A customer is able to cancel a payment before its delivery date.**

Functionality: Customers cancel a payment before its delivery date.

Functionality name**: Cancel Payment**

**Functional Model:**

**Scenarios**

**Example 1:** Susan attempt to cancel a payment by entering amount "$200" and date "04/30/2018" at 9 PM.

|  |  |
| --- | --- |
| *Use case name* | CancelSuccessful:CancelPayment |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. On a payment overview page, Susan activates the “cancel payment” functionality by clicking on the “Cancel Payment” button to cancel scheduled payment which has PaymentID p1111, CheckingAccNu 123456600, Amount $200, payeeID Tom12, PayeeAccountNu 123456789, Payment Delivery Date 04/30/2018.  2.Susan gets redirected to the confirmation page, where Susan clicks on “Confirm Cancel” button.  3. The system displays ‘successful cancel payment’ on the screen. |

**[Table: 46 Cancel Payment Successful Scenario]**

**Example 2:** Susan attempt to cancel a payment by entering amount "$200" and date "04/30/2018" at 9 PM.

|  |  |
| --- | --- |
| *Use case name* | CancelUnsuccessful:CancelPayment |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. On a payment overview page, Susan activates the “cancel payment” functionality by clicking on the “Cancel Payment” button to cancel the scheduled payment which has PaymentID p1111, CheckingAccNu 123456600, Amount $200, payeeID Tom12, PayeeAccountNu 123456789, PaymentDate 04/30/2018.  2.Susan is redirected to the confirmation page, where Susan clicks on “Confirm Cancel” button.  3. The system displays ‘unsuccessful message’ on the screen. |

**[Table: 47 Cancel Payment Unsuccessful Scenario]**

**Example 3:** Susan attempt to cancel a payment by entering amount "$200" and date "04/30/2018" at 9 PM.

|  |  |
| --- | --- |
| *Use case name* | Cancel:CancelPayment |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1. On the payment overview page, Susan activates the “cancel payment” functionality by clicking on the “Cancel Payment” button to cancel scheduled payment which has PaymentID p1111, CheckingAccNu 123456600, Amount $200, payeeID Tom12, PayeeAccountNu 123456789, PaymentDate 04/30/2018.  2.Susan is redirected to the confirmation page, where Susan clicks on “Don’t Cancel” button to exit the Cancel Payment operation. |

**[Table: 48 Cancel Payment Cancel Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | CancelPayment |
| *Participating Actor* | Initiated by a customer |
| *Flow of Events* | 1. Customer is canceling the scheduled payment by clicking on the “Cancel Payment” button on the payment review page at a specific time and on a specific day.  2.Customer clicks on the “Confirm Cancel” button to confirm the cancellation of scheduled payment or clicks on “Don’t Cancel” button.  3. The system shows “successful message "upon successful cancellation of the scheduled payment or “unsuccessful message” on the screen due to connection error or the customer is on the homepage. |
| *Entry conditions* | The customer is logged into his/her Online Banking System. |
| *Exit conditions* | The online banking system shows “Successfully cancel payment” or “Unsuccessful cancel payment” for scheduled payment. |
| *Quality Requirements* | The Process must be done in less than 3 seconds. |

**[Table: 49 Cancel Payment Initial Use Case]**

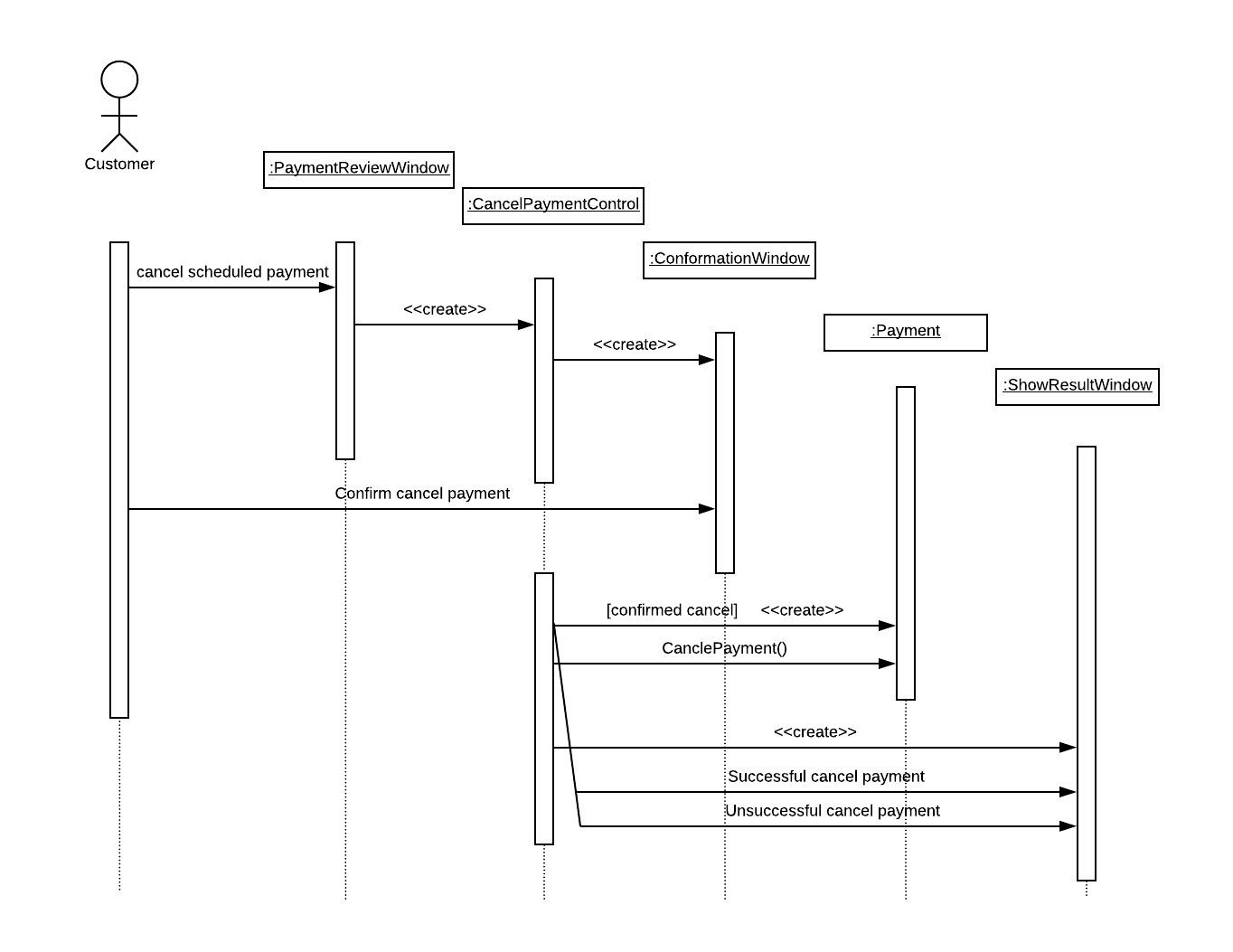
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | CancelPayment |
| *Participating Actor* | Initiated by a customer |
| *Flow of Events* | 1.The customer is canceling the scheduled payment on a specific day and at specific time in PaymentReviewWindow.  2. CancelPaymentControl object is created.  3. CancelPaymentControl prompt to the customer to confirm payment cancel in ConfirmationWindow .  3. If,the customer clicks on confirm payment cancel , CancelPaymentControl object creates the Payment entity object.  4.CancelPaymentControl object invokes the cancelSchedule operation and updates the details into the SchedulePayment object.  5. After the invocation of the cancleSchedule operation, the CancelPaymentControl object displays “Successful cancel payment” upon successful cancellation of the scheduled payment or “Unsuccessful cancel payment” upon connection error /Timeout error in ShowResultWindow. |
| *Entry conditions* | The customer is logged into his/her Online Banking System. |
| *Exit conditions* | The online banking system shows “Successfully cancel payment” or “Unsuccessful cancel payment” for scheduled payment. |
| *Quality Requirements* | The process must be done in less than 3 seconds. |

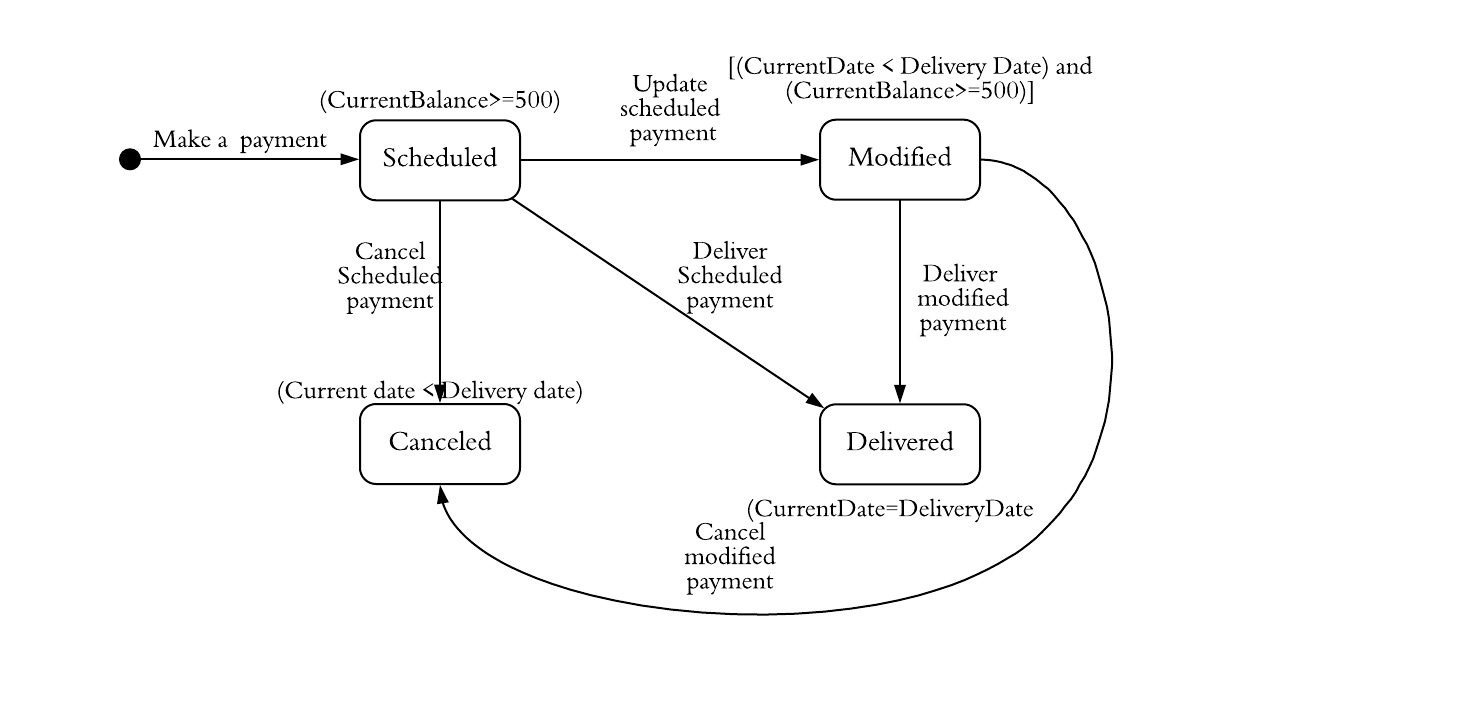
**[Table: 50 Cancel Payment Refine Use Case]**

**Dynamic Model:**

**Sequences Diagram of Cancel Payment**

**[Fig 21: Sequence diagram of Cancel Payment]**

**State Diagram of Payment entity object**

**[Fig 22: State diagram of Payment]**

**3.11. A customer is able to logout.**

Functionality: Customers logout from account.

Functionality name**: Logout**

**Functional Model:**

**Scenarios**

**Example 1:** Susan attempt to logout from her account on 5/01/2018 at 5:00PM.

|  |  |
| --- | --- |
| *Use case name* | LogoutSuccessful: Logout |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1.Susan activates the "Logout" function of the system by clicking on the “Logout tab” on her susansmith account’s main page.  2. Upon successful completion of the Logout, the system displays ‘successful message’ on the screen. |

**[Table: 51 Logout Successful Scenario]**

**Example 2:** Susan attempt to logout from her account on 5/01/2018 at 5:00PM.

|  |  |
| --- | --- |
| *Use case name* | LogoutUnsuccessful: Logout |
| *Participating Actor* | Susan: Customer |
| *Flow of Events* | 1.Susan activates the "Logout" function of the system by clicking on the “Logout” tab on her susansmith account’s main page.  2. The system shows ‘unsuccessful message’ appears on the screen due to unfinished tasks error. |

**[Table: 52 Logout Unsuccessful Scenario]**

**Initial Use case**

|  |  |
| --- | --- |
| *Use case name* | Logout |
| *Participating Actor* | Initiated by the customer |
| *Flow of Events* | 1.The customer is logging out from his/her account by clicks the “Logout” tab at specific time and specific date.  2.The system shows “successful message” or “unsuccessful message” on the screen. |
| *Entry conditions* | The customer is logged into his/her Online Banking System. |
| *Exit conditions* | The online banking system shows “Successful logout” or “Unsuccessful logout”. |
| *Quality Requirements* | The process must be done in less than 3 seconds. |

**[Table: 53 Logout Initial Use Case]**

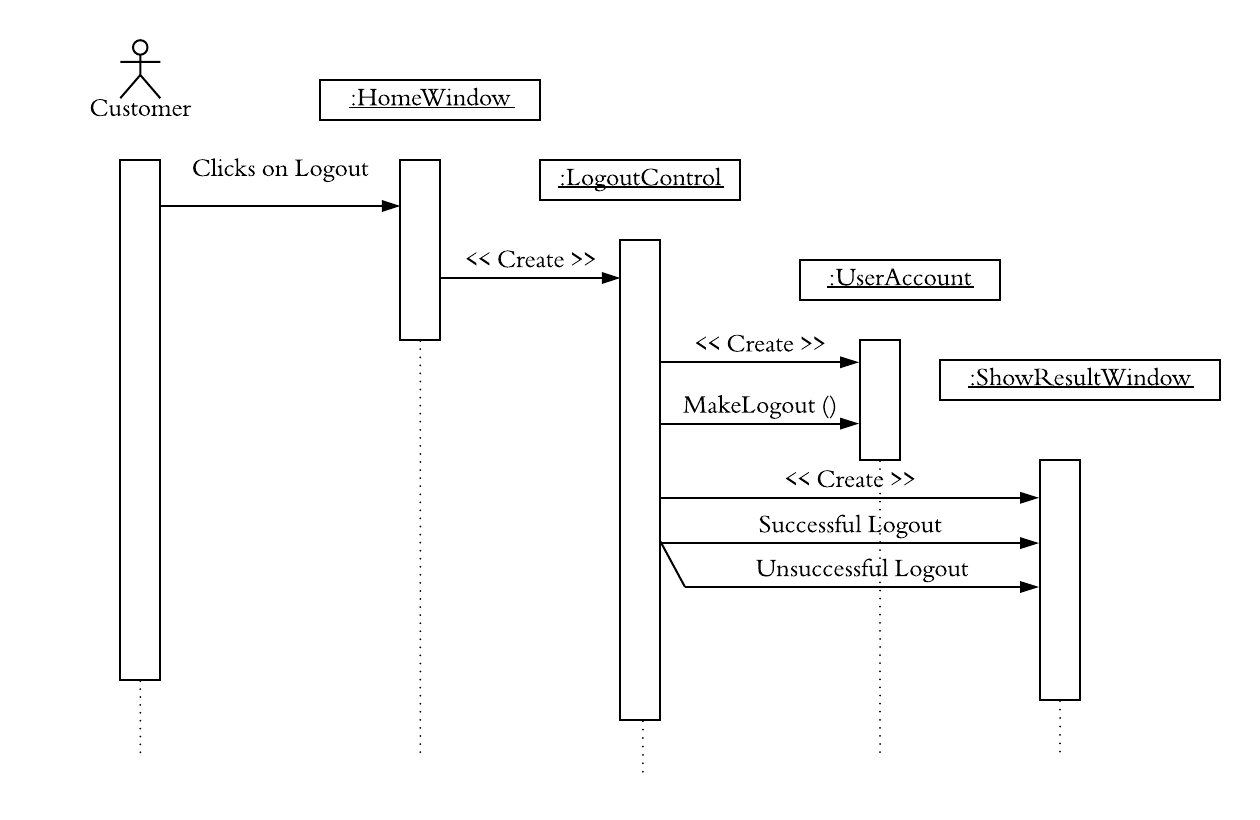
**Refine Use case**

|  |  |
| --- | --- |
| *Use case name* | Logout |
| *Participating Actor* | Initiated by a customer |
| *Flow of Events* | 1.The customer is logging out from his/her account by clicking the “Logout” tab in the HomeWindow.  2.The LogoutControl object is created.  3.The LogoutControl object creates the UserAccount entity object and invokes a logout operation.  4.After invocation of the logout operation, the LogoutControl object displays a “successful logout” or “unsuccessful logout” on a screen in ShowResultWindow. |
| *Entry conditions* | The customer is logged into his/her Online Banking System. |
| *Exit conditions* | The online banking system shows “Successful logout” or “Unsuccessful logout”. |
| *Quality Requirements* | The Process must be done in less than 3 seconds |

**[Table: 54 Logout Refine Use Case]**

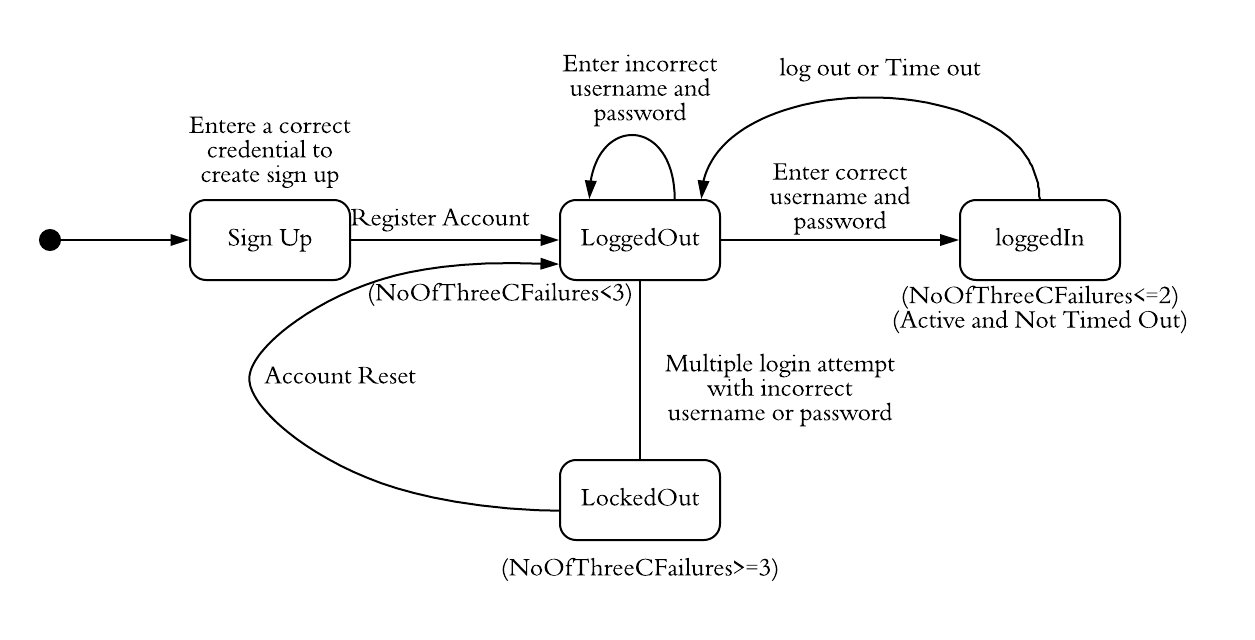
**Dynamic Model:**

**Sequences Diagram of Logout**



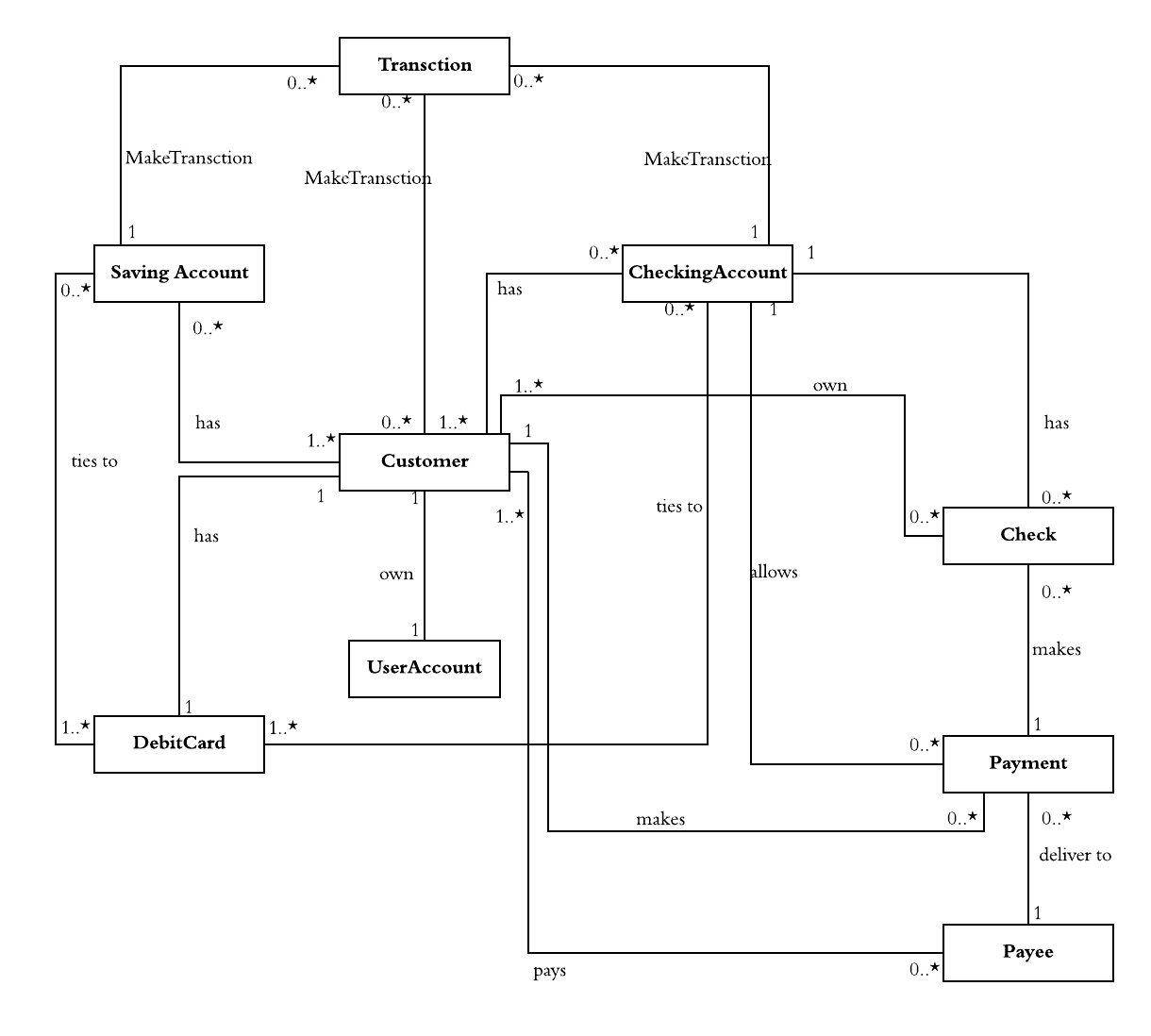
**[Fig 23: Sequence diagram of Logout]**

**State Diagram of UserAccount entity object**

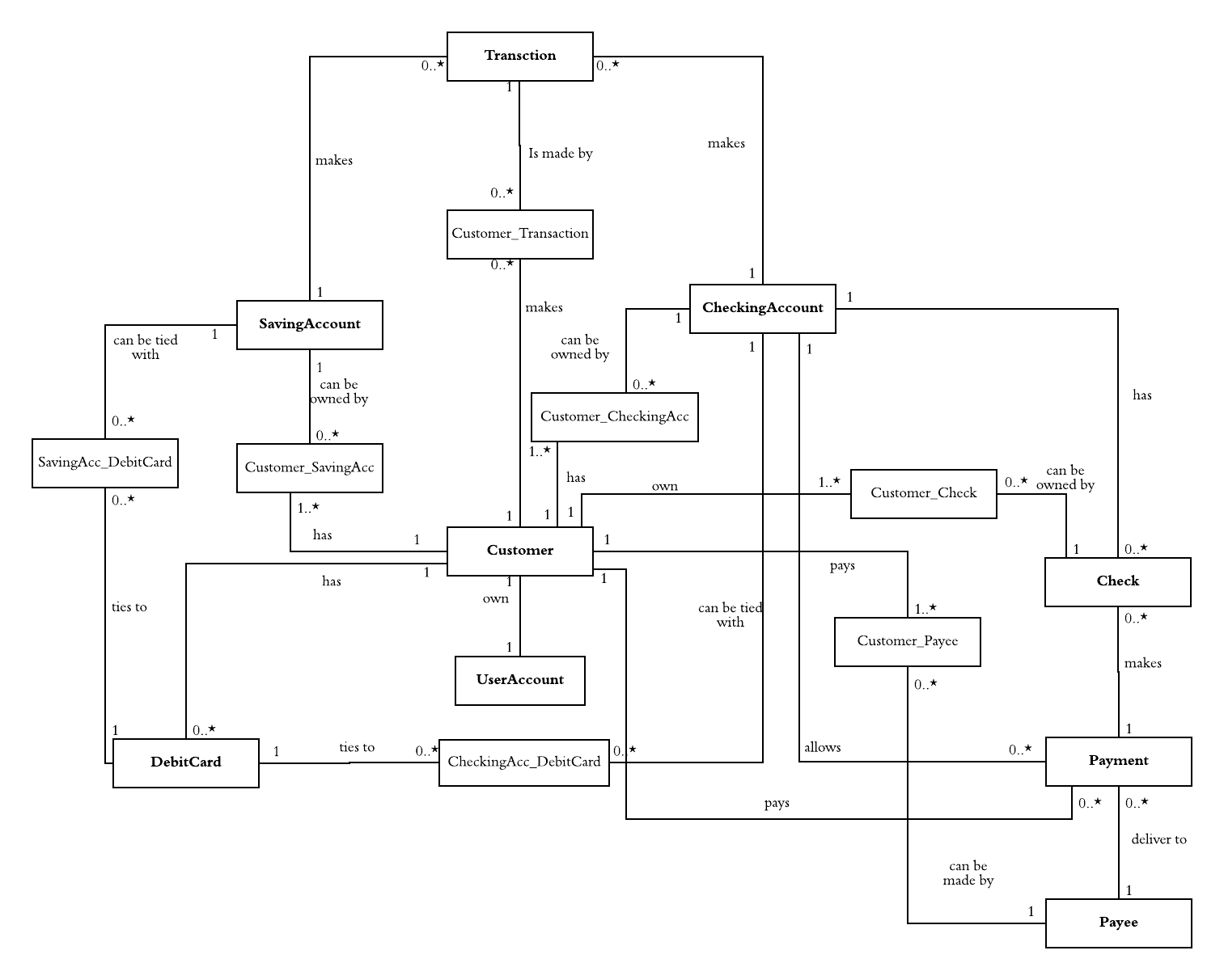


**[Fig 24: State diagram of UserAccount]**

**Analysis Object Model**

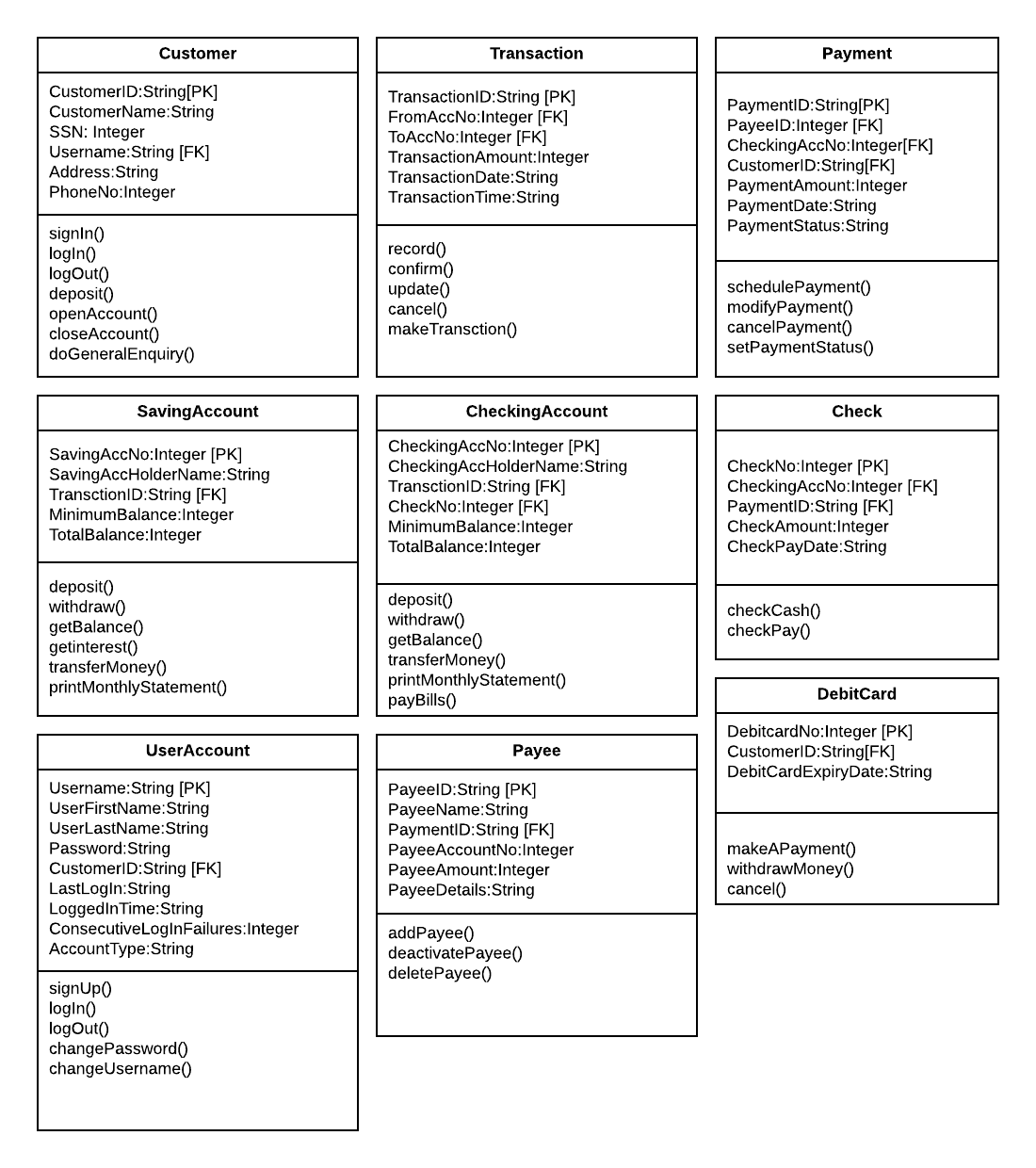
**Initial class diagram**

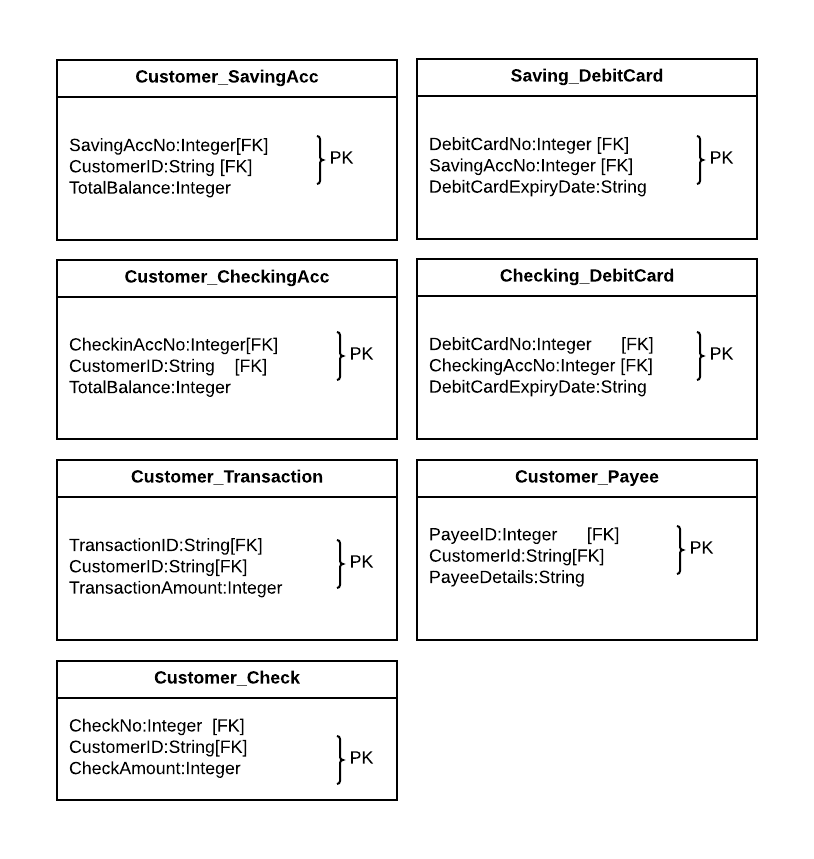
**[Fig 25: Initial Class Diagram]**

**Final class Diagram**

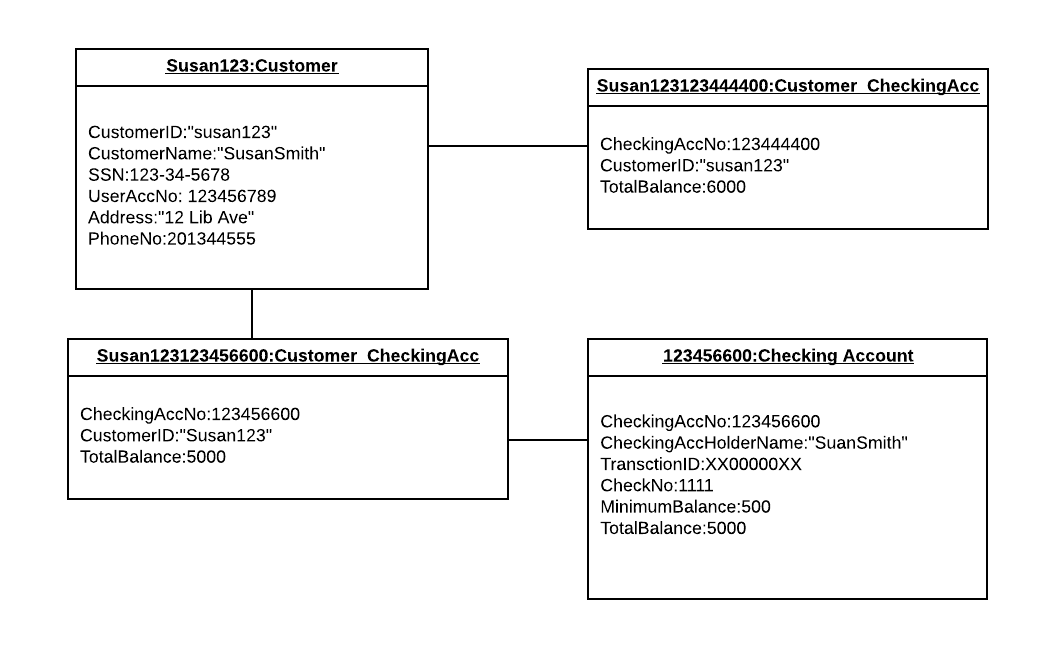
**[Fig 26: Final class Diagram]**

**Object Diagram**

**Class, Attributes and Operations**

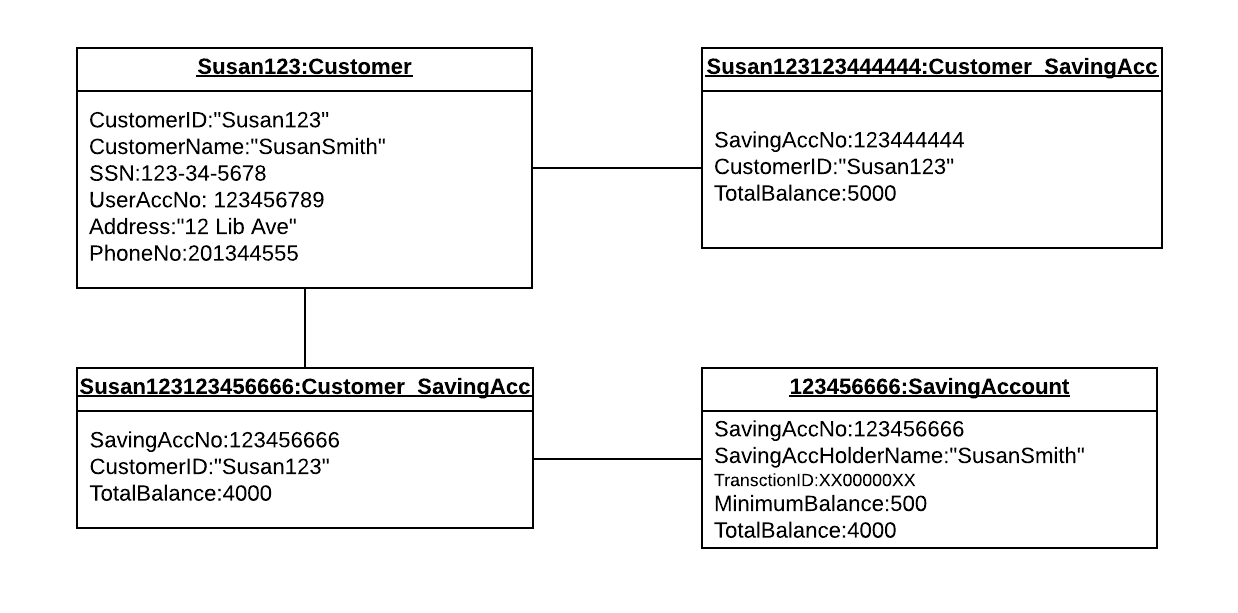


**[Fig 27: Class, Attribute and operation]**

**(1) Object Diagram of Customer and CheckingAccount** //When giving a value to an attrinute, use =, instead of :.

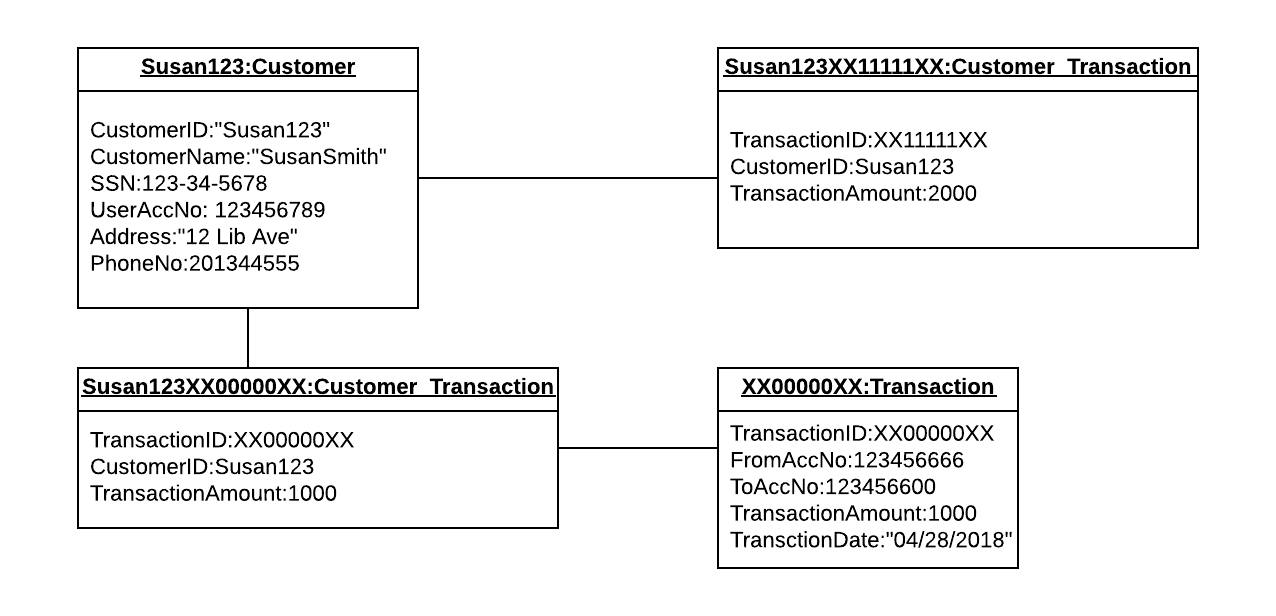
**[Fig 28: Object diagram of customer and checking account]**

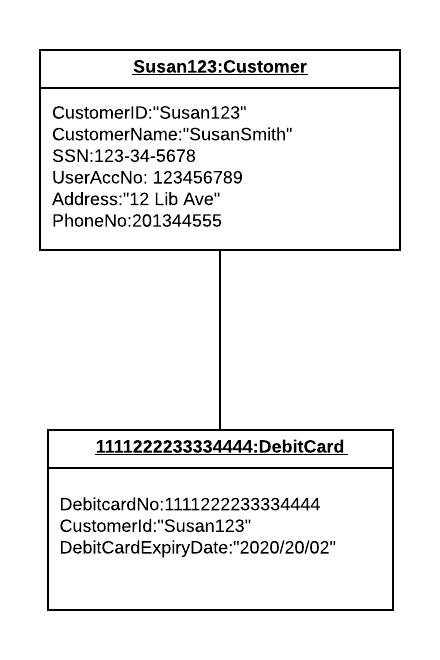
**(2) Object Diagram of Customer and SavingsAccount**



**[Fig 29: Object diagram of customer and saving account]**

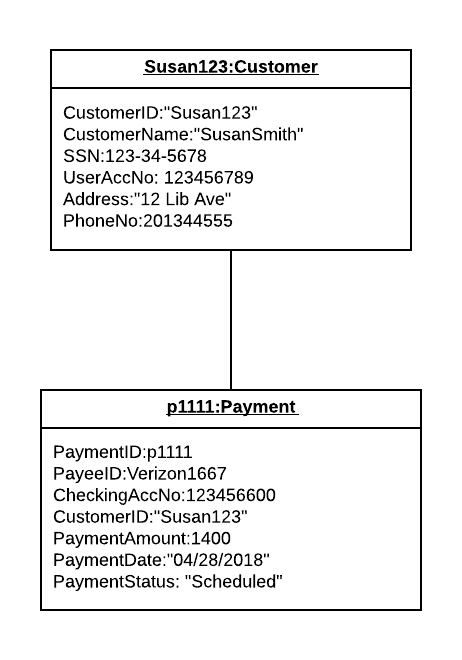
**(3) Object Diagram of Customer and Transaction**

**[Fig 30: Object diagram of customer and transaction]**

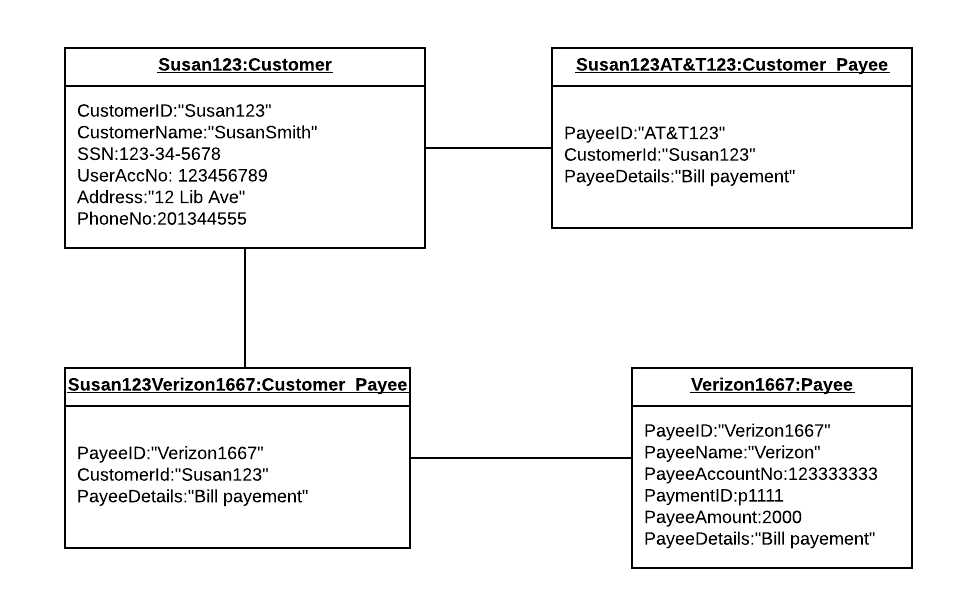
**(4) Object Diagram of Customer and DebitCard**

**[Fig 31: Object diagram of customer and debitCard]**

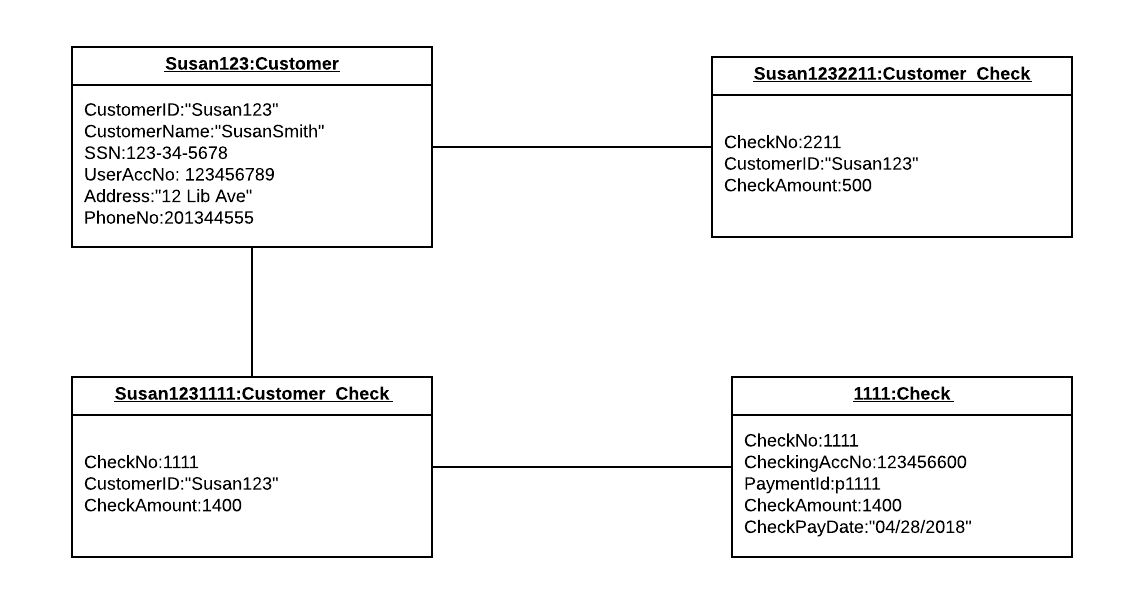
**(5) Object Diagram of Customer and Payment**



**[Fig 32: Object diagram of customer and payment]**

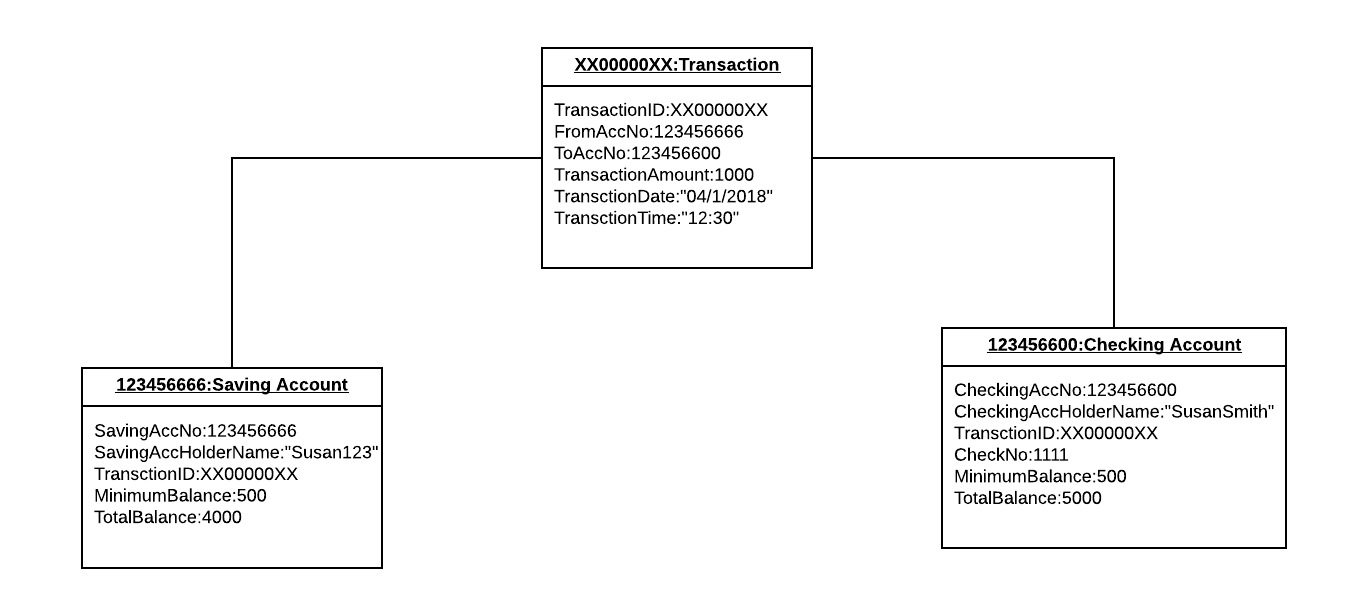
**(6) Object Diagram of Customer and Payee**

**[Fig 33: Object diagram of customer and payee]**

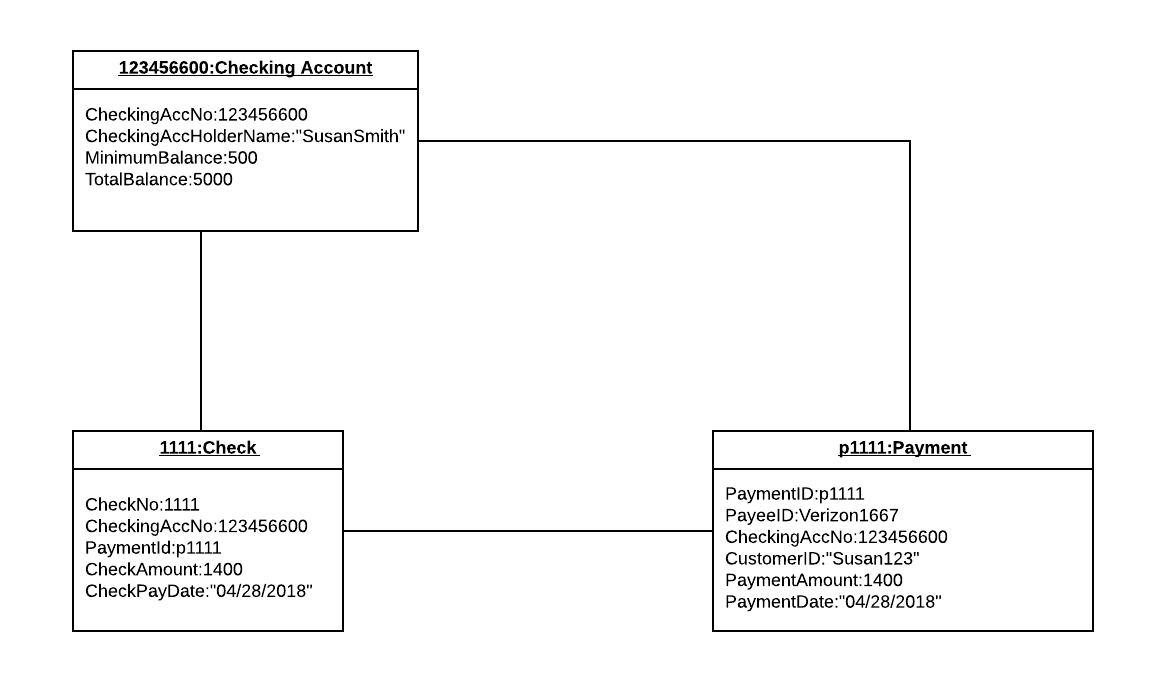
**(7) Object Diagram of Customer and Check**

**[Fig 34: Object diagram of customer and check]**

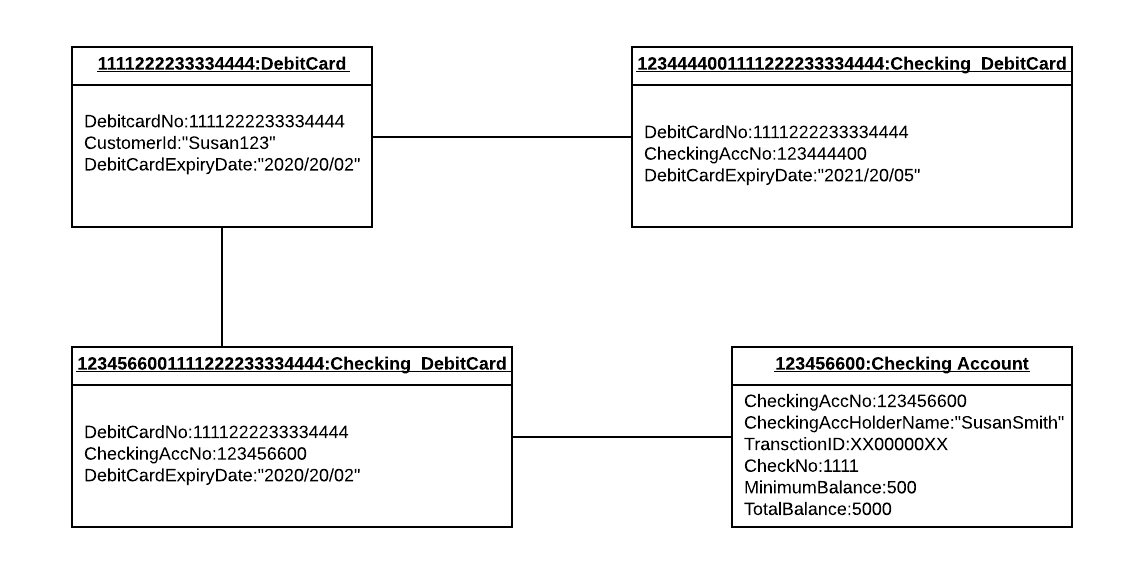
**(8) Object Diagram of Transaction and CheckingAccount** **and SavingsAccount**

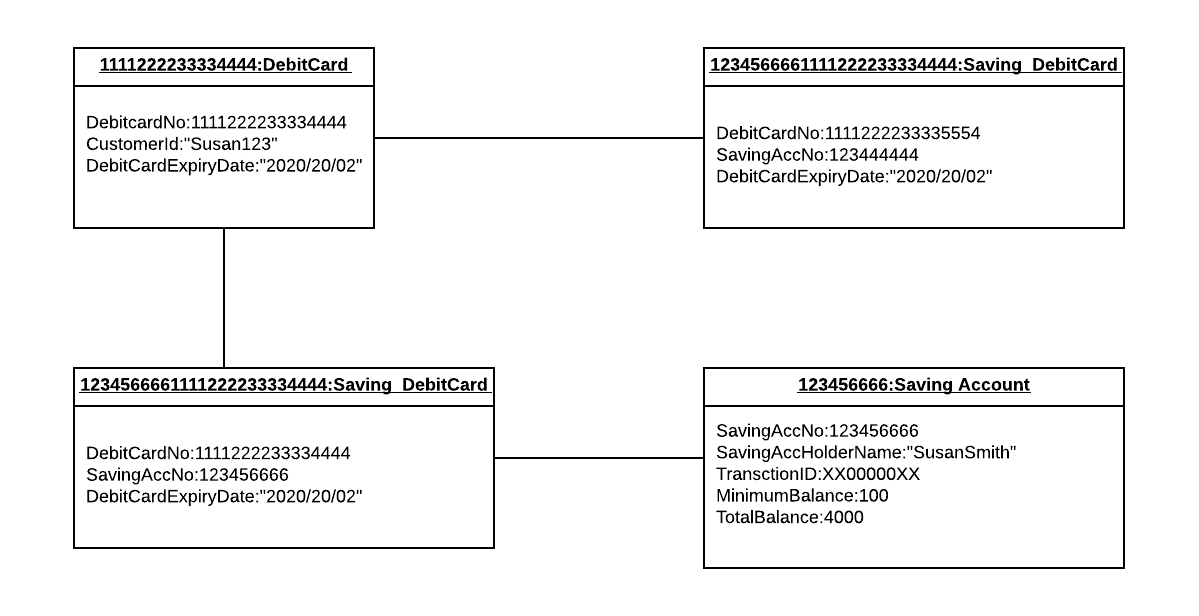
**[Fig 35: Object diagram of transaction, checking account and saving account]**

**(9) Object Diagram of CheckingAccount and Check** **and Payment**

 **[Fig 36: Object diagram of checking account, check and payment]**

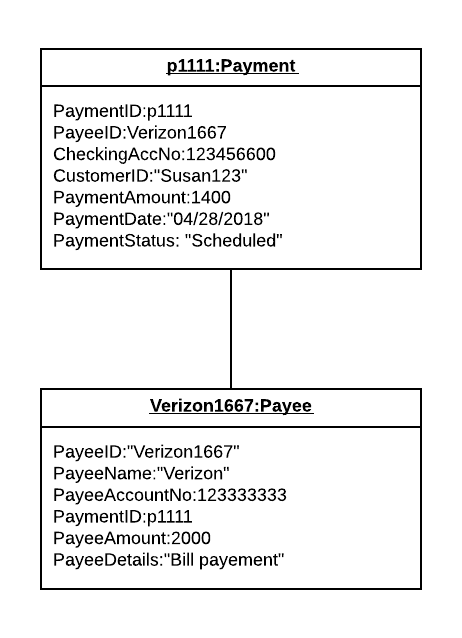
**(10) Object Diagram of CheckingAccount and DebitCard**

**[Fig 37: Object diagram of checking account and debit card]**

**(11) Object Diagram of Savings Account and DebitCard**

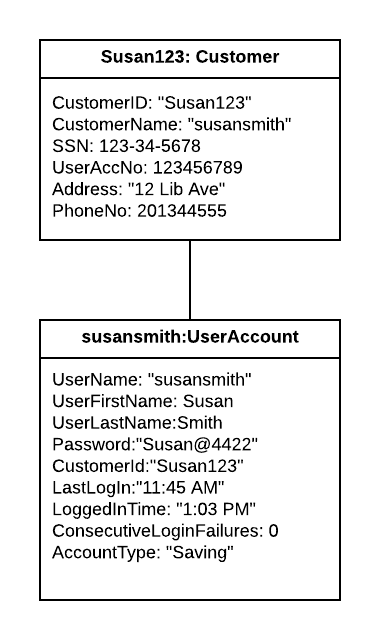
**[Fig 38: Object diagram of saving account and debit card]**

**(12) Object Diagram of Payee and Payment**



**[Fig 39: Object diagram of payee and payment]**

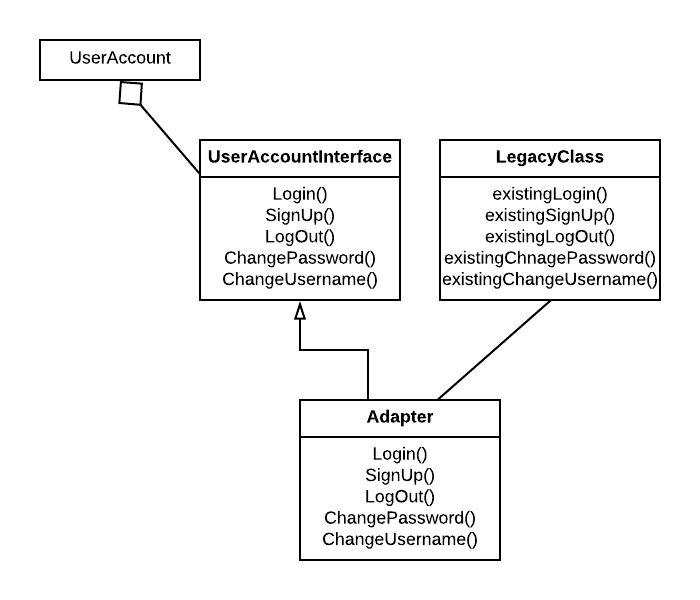
**(13) Object Diagram of Customer and UserAccount**



**[Fig 40: Object diagram of customer and UserAccount]**

|  |
| --- |
| 4. Object Design |

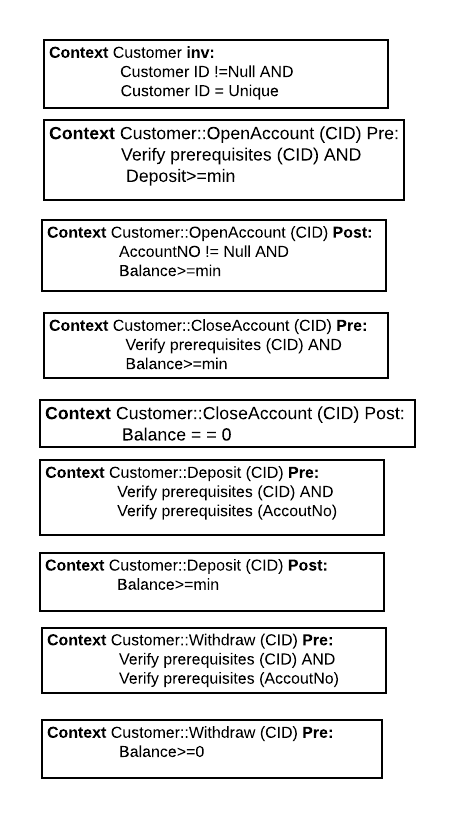
**4.1 Design pattern: Adapter**

**Adapter for UserAccount entity object**

**[Fig 41: Adapter Design pattern for UserAccount]**

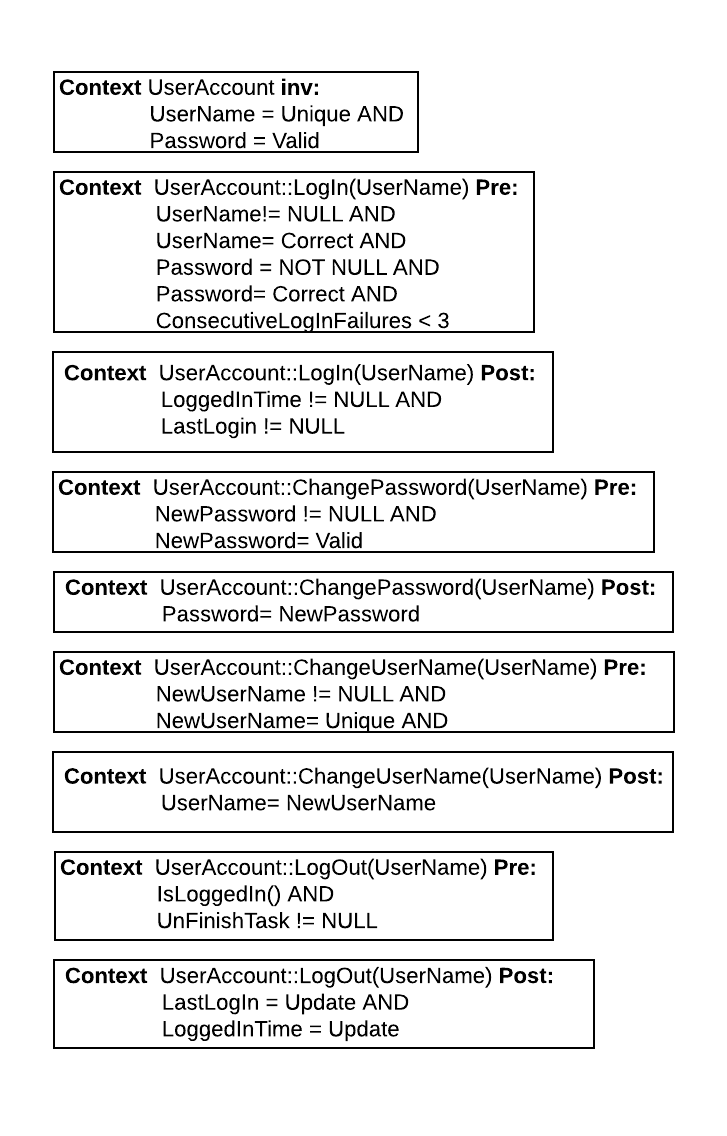
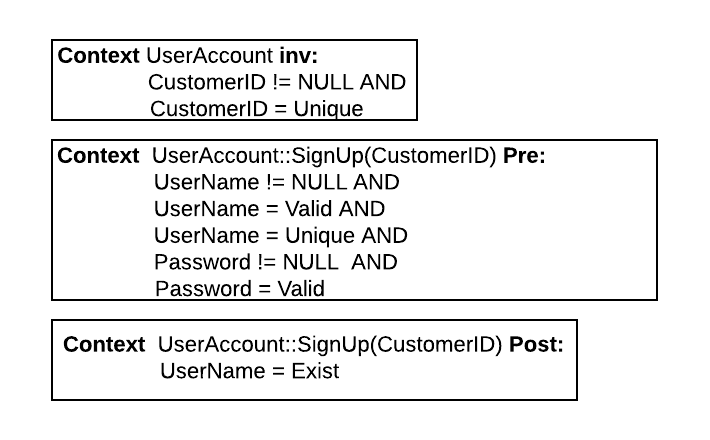
**4.2 Interface Specification**

**4.2.1 Interface specification of Customer**



**//deposit(), withdraw() are not operations of Customer.**

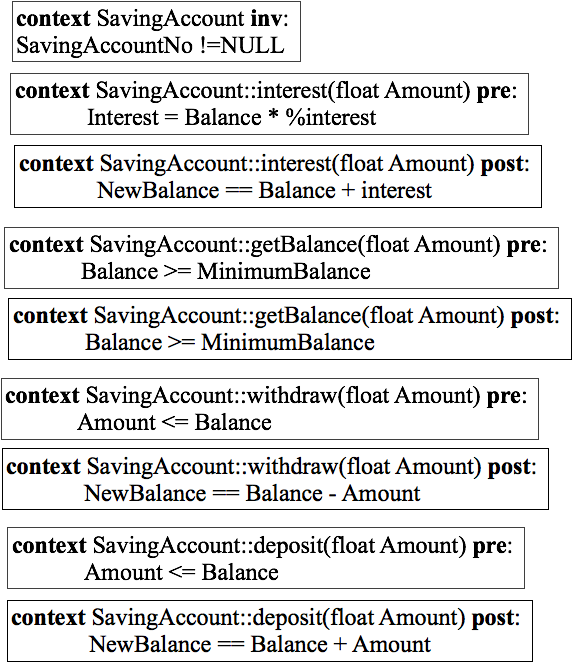
**4.2.2 Interface specification of UserAccount**



**//ConsecutiveLoginFailures < 3 is not necessary as a part of login() pre:**

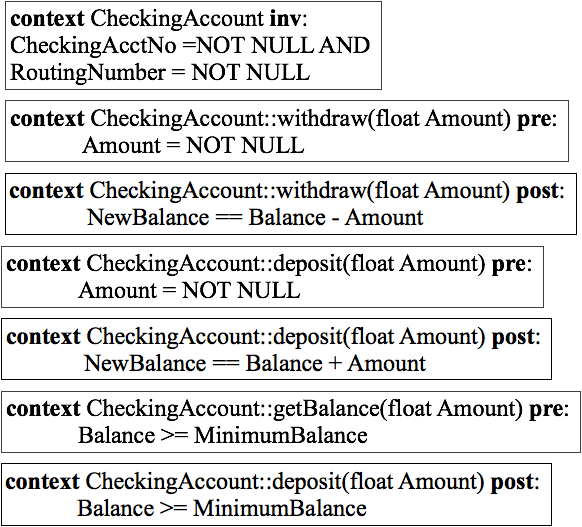
**//ChangePassword() also needs a parameter NewPassword. changeUsername() needs a parameter NewUsername**

**4.2.3 Interface specification of SavingAccount**

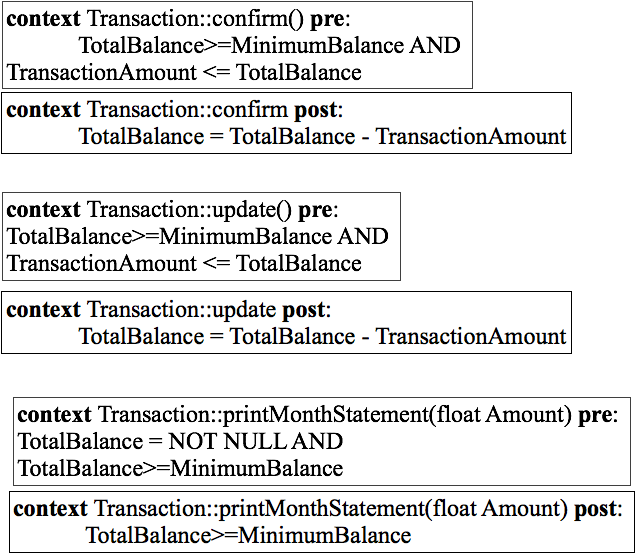
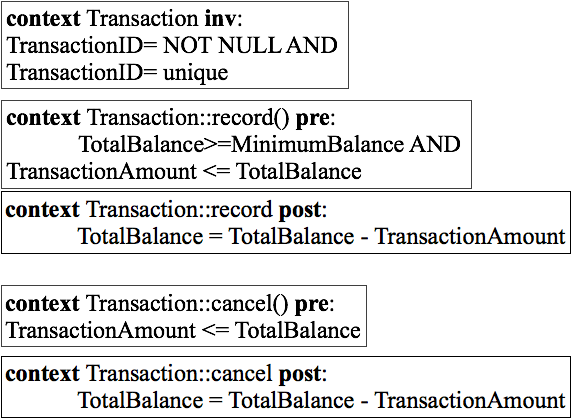


**//Amount>0 is necessary for both withdraw() and deposit(). The same for CheckingAccount.**

**4.2.4 Interface specification of CheckingAccount**



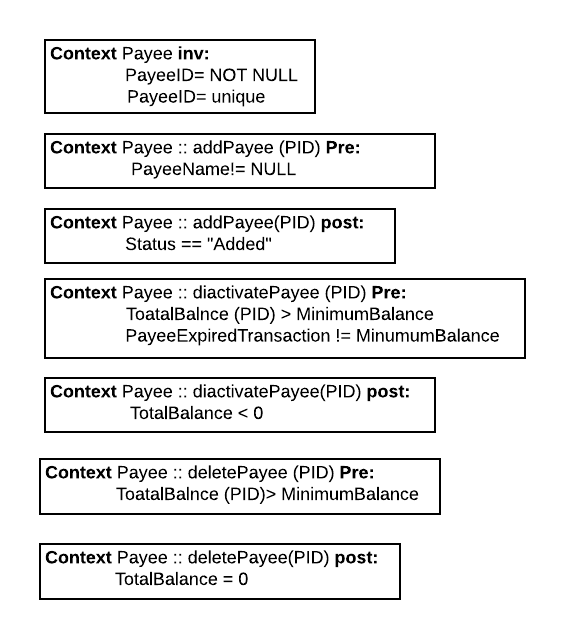
**4.2.5 Interface specification of Transaction**



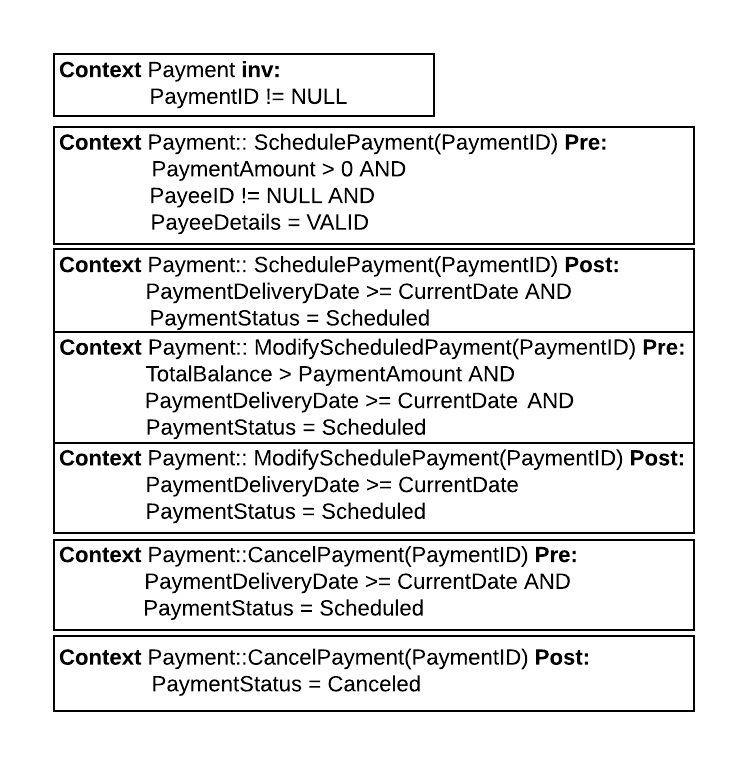
**//TotalBalance >= MinimumBalance does not make sense to the above operations.**

**4.2.6 interface specification of Payee**

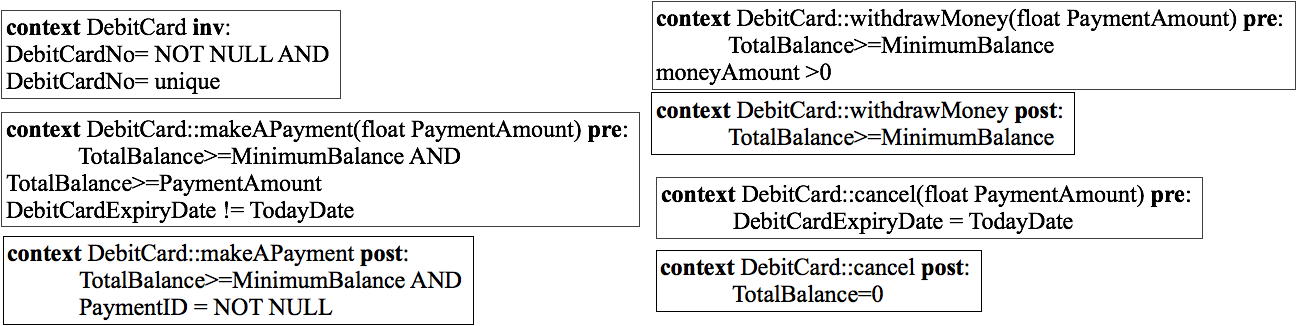
//Why TotalBalance<0 is the post-condition for diactivatePayee()? It does not make sense.

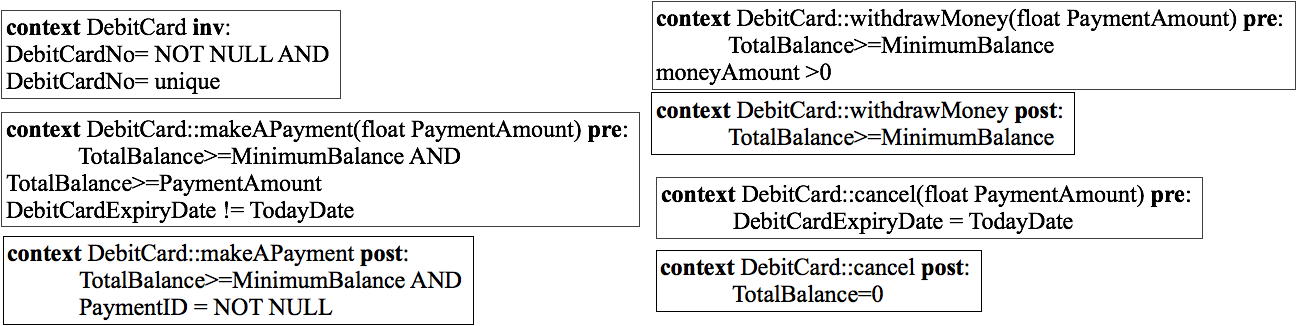


**4.2.7 Interface specification of Payment**

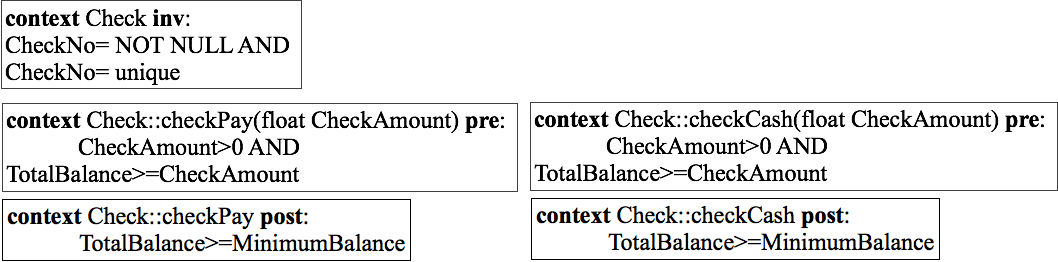
//parameters to schedulePayment() are PayeeID and PaymentAmount.

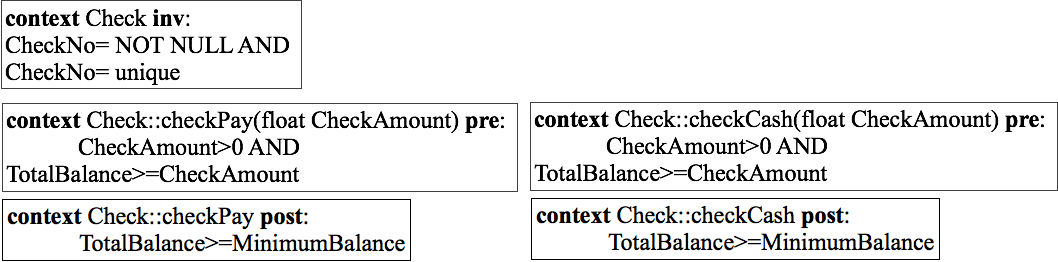
**4.2.8 Interface specification of Debit Card**





**4.2.9 Interface specification of Check**

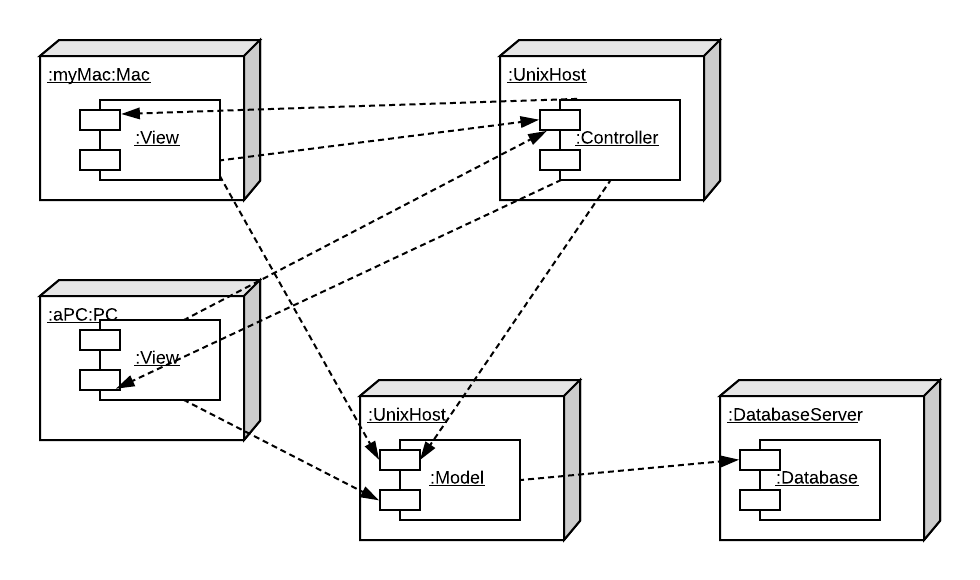




//All errors in Interface Specification (-5)

|  |
| --- |
| 5.System Design |

**5.1 UML Deployment Diagram**

****//Missed indicating the software architectural style is MVC + Client/Server (-3) though your deployment diagram is right.

**[Fig 42: UML deployment diagram for client server base]**

**5.2 Access Control**

**5.2.1 A global access tables**

**UserAccount**

(Customer, UserAccount, signUp())

(Customer, UserAccount, loginIn())

(Customer, UserAccount, logOut())

(Customer, UserAccount, changeUsername())

(Customer, UserAccount, changePassword())

**SavingAccount**

(Customer, SavingAccount, withdrawMoney())

(Customer, SavingAccount, depositMoney())

(Customer, SavingAccount, transferMoney())

(Customer, SavingAccount, getBalance())

(Customer, SavingAccount, getInterest())

(Customer, SavingAccount, printMonthlyStatement())

**CheckingAccount**

(Customer, CheckingAccount, withdrawMoney())

(Customer, CheckingAccount, depositMoney())

(Customer, CheckingAccount, transferMoney())

(Customer, CheckingAccount, getBalanceMoney())

(Customer, CheckingAccount, paybills())

(Customer, CheckingAccount, printMonthlyStatement())

**Transaction**

(Customer, Transaction, recordTransaction())

(Customer, Transaction, confirmTransaction())

(Customer, Transaction, updateTransaction())

(Customer, Transaction, cancelTransaction())

(Customer, Transaction, makeTransaction())

**Payment**

(Customer, Payment, schedulePaymentt())

(Customer, Payment, modifyPaymentt())

(Customer, Payment, cancelPaymentt())

(Customer, Payment, paymentStatus())

**Payee**

(Customer, Payee, addPayee())

(Customer, Payee, deletePayee())

(Customer, Payee, deactivatePayee())

**Check**

(Customer, Check, checkPay())

(Customer, Check, checkCash())

**DebitCard**

(Customer, DebitCard, makeAPayment())

(Customer, DebitCard, withdrawMoney())

(Customer, DebitCard, cancel ())

**5.3 Strategies for global control**

The system uses the combination of the event-driven control and threads.

**5.4 Boundary use case**

**Boundary use case of Configuration in online banking system**

|  |  |
| --- | --- |
| *Use case name* | ConfigureSystemForUsingDBs |
| *Participating Actor* | Initiated by Systems Admin |
| *Flow of Events* | 1.A systems admin activates the "configure " Functionality.  2.Choose configure DB option for the application.  3.The system prompts the user to enter DB names.  4.The users enters DB names.  5.The system confirms the DB names. |
| *Entry conditions* | The admin has logged to the system. |
| *Exit conditions* | The DBs are available for the system. |
| *Quality Requirements* |  |

**[Table: 55 Boundary Use Case for configuration]**