April 29, 2024

Assignment – 6

[300 Points]

CSC-413-02
Spring 2024

San Francisco State University
Computer Science Department

Assignment Goals:

This will be the culmination of the Banking Application that you have been building parts of during the course of this semester in this class. At a high-level, in this assignment you will be creating the remaining Graphical User Interface (GUI) components for Business Objects. Having these components will allow you to:

(a) Enter data required to search and retrieve single/multiple business objects of a type(customer, account, transaction)
(b) Display detail for these business objects
(c) Persist(save to db) newly created objects
(d) Update and save existing business objects

Note:

1. Please use the Project provided with the assignment to add all requisite classes, as also the methods (functionality) for them, as asked for in the assignment.
2. Before starting with this final assignment, perform the following steps to load project within your IDE
   (a) Unless your IDE allows you to import the zipped version of a Java project, unzip the project zip
   (b) Import the project into your IDE
   (c) Validate the project structure to ensure there are following 3 packages. Each of these packages have classes to get you started with this final assignment
   - Assignment6Controller
   - Assignment6Model
   - Assignment6View

Objective-1: Use the GUI provided to display multiple BankCustomer Objects

Task1 – Searching and Displaying BankCustomers [50 pts]

In this section you will first write code in the GUI frame to search customers based on a search criteria, and then display. The skeleton **CustomerSearch** and **CustomerList** GUI frames have been provided in the Assignment6View package of the project. The CustomerSearch GUI will have the field(s) that will provide the

search criteria data to retrieve BankCustomers, while the CustomerList will have the widget to display the retrieved customer data. You will need to write code that will use the data from the search field(s) to run db query, and then display BankCustomer object(s) retrieved. [25 pts]

1. While many search criteria, such as customers living in a city, or customer with an ID, or customers with a specific last name are legit, for the purposes of this assignment and to keep it simple, we will only search for customers that live in a given city. To do that:

   (a) You will need to define a click action for the 'Search' button in the **CustomerSearch** GUI frame such that when the user clicks it, a method will run that interacts with the CustomerDTO and DAO objects to run a database query similar to the following:

   *Select * from Customer where city = ?*

   (b) In the above statement, the '?' will be replaced by the city name you would have entered in the city field of the **Customer Search** GUI frame. This method should be named '*findCustomer()*' that will get data in the City field and initiate the query, providing the city as the argument. Refer to

   Note:
   A process similar to the one used to create a customer as part of Assignment-3, that used the CustomerDTO and CustomerDAO objects, will be also be used here. Only difference will be the query to fetch BankCustomer object(s) will be based on the search criteria

   (c) The above query could return one or more customer object(s). All the objects returned by the query will then have to be displayed in the List widget in the provided **Customer List** GUI frame. To do that, you will be required to 'display/open' the **Customer List** GUI frame and update the list widget with the customer objects returned from the DB

Task2 – Updating a BankCustomer selected from the list [50 pts]

1. For this task the **CustomerDetail** GUI frame has been provided. You will implement click action that will open up CustomerDetail frame with details of the customer that was selected in the list. To do that:

   (a) You will need to define a click action in the CustomerList frame for the 'Show Details' button such that when the user clicks it a method will be run that opens the **CustomerDetail** GUI frame. This method should be named '*showCustomerDetail()*'
   (b) Upon opening, the frame will display the customer attributes, as pulled out from the BankCustomer selected in the list, and populate the appropriate fields of the detail frame
   (c) Keep in mind that CustomerAddress is part of the core attributes of the BankCustomer. Therefore, the CustomerDetail frame will include the Address field, which will show the string representation of the CustomerAddress related to the BankCustomer
   (d) To look at the address detail for this customer, you will click the 'Update Address' button, which should open the **CustomerAddress** GUI frame. You should be able to update any field of the CustomerAddress in this frame and click Save to update the address for the customer. So, you will need to a click action for the Save button on the CustomerAddress GUI frame. The method to save address should be called '*saveCustomerAddress()'*
   (e) You will then update any of the other non-address fields and click 'Save' button on the CustomerDetail. This should save the customer with the updated data. This method should be named '*saveCustomer()*'

Objective-2: Use the GUI provided to display BankAccount Objects

Task1 – Displaying List of BankAccounts [50 pts]

The **AccountList** GUI frame that will display BankAccounts has been provided. You will need to write code in this to run a search that returns BankAccount object(s) [25 pts]

1. While many search criteria can be defined to search for BankAccount(s), for the purposes of this assignment we will search for account(s) that are associated with the customer selected from the Customer list in Objective-1. To do that:

   (a) You will need to define a click action for the 'Show Accounts' button on the **CustomerDetail** GUI frame such that when the user clicks it, a method will be run that runs a query similar to the following:

   *Select * from Account where customerId = ?*

   (b) In the above statement, your code will replaced '?' by the ID of the customer you selected from the Customer List. This method should be named '*findCustomerAccounts()*'

   (c) Since a customer may have 1 or more accounts, the above query could return one or more account object(s). The account object(s) returned by the query will be displayed in the list widget within provided **AccountList** GUI frame. To do that, you will be required to 'display/open' the **AccountList** frame and display the retrieved accounts in the list widget within the frame.

   Note:
   You will need to create the BankAccountDTO and DAO classes, along the lines of BankCustomerDTO and DAO, that will have the methods to create, retrieve and update account objects

Task2 – Updating a BankAccount selected from the list [50 pts]

1. For this task you will create controller classes in support of BankAccount class, and also build functionality to show details for the account selected in **AccountList** frame that has been provided. To do that:

   (a) Create AccountDTO class, in the Controller package of the project, modeled similar to CustomerDTO
   (b) Create AccountDAO class, in the Controller package of the project, modeled similar to CustomerDAO
   (c) Define a click action for the 'Show Detail' button on the **AccountList** frame such that when the user clicks on it after selecting an account from the account list widget, a method will be run that opens the **AccountDetail** GUI frame
   (d) Upon opening, the AccountDetail frame will display the Account attributes, as pulled out from the BankAccount object that was double-clicked on in the list, and populate the appropriate fields of the **AccountDetail** frame.
   (e) You will then update any field and click Save button. This should save the account with the updated data.
   (f) You will need to define a click action for the 'Save' button on the **AccountDetail** frame such that when the user clicks on it after updating the appropriate field(s) of an account, a method will be run that saves the account object. This method should be named '*saveAccount()*'

Objective-3: Use the GUI provided to display AccountTransaction Objects

Task1 – Displaying List of AccountTransactions [50 pts]

For this task you will create controller classes in support of AccountTransaction, and also build functionality to show details for the account selected in **AccountList** frame The **AccountTransactions** GUI frame that will display transactions related to an account is provided. You will need to write code in the widget to initiate a search that returns Transaction objects [25 pts]

1. While many searches for transaction(s) related to an account may exist, such as transaction(s) relating to a bank account or transaction(s) relating to accounts of a BankCustomer, for the purposes of this assignment we will search for an account whose ID we will be provided in the **AccountSearch** GUI frame. To do that:

   (a) You will need to define a click action for the 'Search' button such that when the user clicks it, a method will be run that interacts with the BankAccountTransactionDTO and BankAccountTransactionDAO objects to run a database query similar to the following:

   *Select * from transaction where accountId = ?*

   (b) In the above statement, the '?' will be replaced by the account ID you would have entered in the AccountID field of the SearchTransaction GUI frame. This method should be named '*findTransactions()*'

   (c) The above query could return one or more transaction object. All transaction objects returned by the query will then have to be displayed in the provided GUI that will display a list of account objects as returned by the query. To do that, you will be required to 'display/open' the 'TransactionList' GUI and update the list with the returned transaction objects

   Note:
   Unlike in the case of BankCustomer, the BankAccountTransactionDTO and BankAccountTransactionDAO classes will need to be created in the Controller package of the project. They should follow the architectural framework as DTO and DAO classes in support of the BankCustomer

Assignment Submission:

As mentioned at the start of this assignment, you will be adding all new classes to the project that you created in Assignment-3 and augmented in Assignment-5. Therefore, the submission for this assignment should be a zipped version of the project.

Process for Application Testing

Your test process should be initiated in the Driver/Adaptor class you had created for testing Assignments 3 and 5. Since testing related to this assignment will be the final test of your BankingApplcation, at a high level, your testing process should do the following:

1. Launch the Landing Page of the Application
2. Start the Banking Application and show the Main Menu page
3. Once on the Main Menu page, you must test your Banking Application in the following individual steps, which will then also be used by the user/grader to test your application code

Testing Step1 [10 pts]

- Click on “Show All Customers” button. This should open the CustomerList frame with all customers currently in the DB

Testing Step2 [10 pts]

- Select a customer from the list and click “Show Detail” button. This should open the CustomerDetail frame with all attributes of the selected customer

Testing Step3 [10 pts]

- Update a customer attribute in a field and click ‘Update’. This should update the customer detail in the database and show MessageBox with the message “Customer Detail Updated Successfully..”
- Click ‘Close Button’

Testing Step4 [10 pts]

- Click “Show Account” button on the CustomerList frame
- This should open the BankAccountList frame showing all accounts related to the selected customer

Testing Step5 [10 pts]

- Click “Account Details” button on the BankAccountList frame
- This should open the AccountDetail frame showing detail for the selected accounts
- Click Close button to close AccountDetail frame

Grading Criteria:

Your submission will be evaluated based on the following criteria:

1. Correctness and completeness of the implementation in accordance with individual requirements stated within each objective
2. Adherence to best coding practices and documentation standards as specified in the coding guidelines document
3. Accuracy of the test cases and demonstration of the flow of test steps outlined in the ‘Process for Testing Application’ section above