

Bank Teller’s Simple Banking Application Development

3/06/24

…

INFO213

67307220

Jordan

Withell

25108392

Daniel

Morris



# Features

* Adding / editing customer details
* Searching customers by customer number or last name
* Adding / editing accounts for customers,
* searching accounts by number
* Searching transactions and making (cash) deposits into customers’ accounts.
* Exporting to XML format
* Uploading in XML format to 3rd party APIs
* Importing of data in XML format
* Error handling and logging

# Adding/Editing customer details

From the Main Menu, under the Customer tab, there is a Add button which opens the AddCustomer form, where you can enter details of the new customer, there is a status bar at the bottom which will display any errors but incorrectly filling fields or “Success” if it works.

You can edit customer details by pressing the Edit customer button on the List Customers form or the Search Customers form. The design of this form

# Searching customers

A form for searching customers by number or last name exists and is reachable by pressing the Seach button under the Customer tab. There are 2 buttons to show the accounts of the account or to edit the account.

# Adding / editing accounts for customers

When looking at the accounts of a customer, there are buttons to edit and add accounts, editing an account open up another form for the user to fill out while adding an account only uses a radio button for whether the new account is a Current Account or a Savings Account. You are also able to search Accounts by number from the main menu under the Accounts tab.

# Exporting to XML format

When an account is selected, the button to Export is enabled. Pressing this button opens the Save File dialog box to select a location to save the XML file with the transactions. When you select a location, a XML file will be generated with all the transactions in the account and the customer details. A success dialog will appear to tell the user that it was successful.

# Uploading in XML format to 3rd party APIs

Next to the export button, there is a Upload button to upload the account in xml format to a 3rd party API. Currently it is hardcoded to be to <http://c141kn.canterbury.ac.nz/sbmxmlv/>. If this fails, a Network Error Exception is raised. Otherwise, a success dialog pops up to inform the user.

# Importing Accounts in XML format

On the main menu, under the Accounts tab, there is an import button that lets you choose a XML on your computer to import into the application. A dialog appears when completed to tell the user whether it was successful or not. We achieved this by using the JadeXMLParser, and storing the attributes as they were read by the parser until the closing tag for the class was found and the object was created.

# Design Patterns

## Singleton Pattern

We have used the singleton pattern to only hold one bank instance. This is because this application is for 1 bank only so we should make sure there is only one instance of the class. This helps prevent referencing a different bank object that doesn’t hold all the customer data.

## Bridge Pattern

We have used the bridge pattern for transactions and bank accounts where the superclass is the abstraction, and the implementation is done in the subclasses. For example, getChange() returns the change of the bank account, and it changes depending on what subclass it is.

## Factory Pattern

For exporting accounts to XML, a factory like pattern was used to create the xml, transactions is the object and addTransactions is the factory method.

# Walk through instructions

Included in the submission is some example XML files that can be imported. Import any of these files with the import button described in Importing Accounts in XML format. Now a customer will be created with a bank account and the transactions. You can now use all the features outlined above.

To test the Error Logger, one way to do this is to disconnect from the wifi and try to upload an account to the 3rd party API, it will timeout and a Network Error Exception will be created and written to the file errorLog.log.