

Requirements

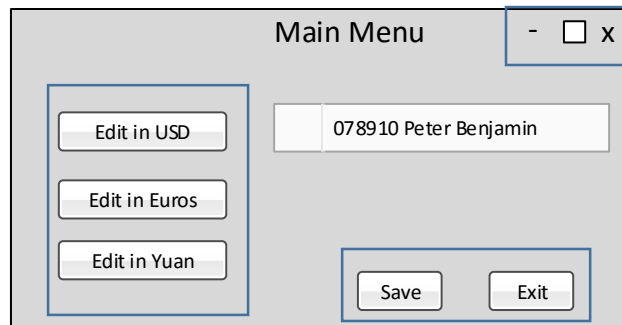
Example Input

| name | id | amount |
|----------------|--------|----------|
| Malapa Bushra | 123456 | \$100 |
| Peter Benjamin | 078910 | \$200.50 |
| Goday Priya | 125671 | \$0.50 |
| John Smith | 444451 | \$3000 |
| Daniel Nguyen | 000xxx | \$2000 |

| Simple Account Management | |
|---------------------------|---|
| ID | Requirement |
| 1 | Software system shall use Model View Controller Architecture |
| 2 | The view shall receive updates from a model by events using a push model |
| 3 | The exchange rates can be programmed as constants (public final static double) in the model as: 1 USD = 0.88 Euro 1 USD = 6.47 Yuan |
| File IO | |
| 4 | On startup, the system should load a list of accounts from a file specified in a command line |
| 5 | An entry in a file with a list of accounts shall contain a string with <ul style="list-style-type: none">a first namelast name separated by a space (only letters allowed)a string with an ID (only digits allowed);a string with an amount in \$ (digits and decimal point allowed; thus amount >=0.0).IDs are unique;names can repeat |
| Main Menu | |
| 6 | On startup, the system shall present the user with a frame that contains a drop down list populated with account IDs with names appended (in increasing order of IDs) |
| 7 | The drop down list of account IDs shall have the first account ID should be pre-selected |
| 8 | The drop down list of account IDs shall have the items in the drop down list be immutable |
| 9 | The frame shall contain buttons "Edit in USD; "Edit in Euros"; "Edit in Yuan" |
| 10 | The frame shall contain button Save |
| 11 | The frame shall contain button Exit |
| 12 | On pressing "Save" the system shall write the current state of accounts to the file whose name appeared in the command line. |
| 13 | On pressing "Exit" the system shall write the current state of accounts if they have been modified to the file whose name appeared in the command line since last save. |
| 14 | After executing 14 the system shall exit |
| 15 | On pressing one of the buttons "Edit in .." the system shall open another window that contains: window's title NameOfAccountHolder ID; Operations in {USD or Euros or Yuan}> |
| 16 | On pressing one of the buttons "Edit in .." the system shall open an Account window that contains: immutable textfield that shows the current amount in the account titled "Available funds" (i.e. Available funds: <textfield>) |
| 17 | On pressing one of the buttons "Edit in .." the system shall open an Account window that contains: editable textfield initialized to 0.0 that only allows digits and decimal point to be entered; titled "Enter amount in {USD or Euros or Yuan}" |
| Account Window(s) | |
| 18 | On pressing one of the buttons "Edit in .." the system shall open an Account window that contains: |

| | |
|---------------------------|---|
| | buttons "Deposit", "Withdraw", "Dismiss" |
| 19 | In the Account window, on pressing "Dismiss" the window shall close |
| 20 | In the Account window, on pressing "Deposit" if the editable textfield contains a positive number greater than 1 then that amount is added to the account amount; |
| 21 | The result of the operation in one Account window must be seen in all other Account windows open for this account with appropriate exchange rate modifications so that all open windows show consistent account state. |
| 22 | In an Account window, on pressing "Withdraw" if the editable textfield contains a positive number greater than 1 and there are sufficient funds then that amount is subtracted from the account amount |
| 23 | In an Account window, the result of the withdraw operation must be seen in all other windows open for this account with appropriate exchange rate modifications. |
| 24 | If funds requested for withdrawal are insufficient, then an exception must be raised by some method from a class in the model package that will eventually be caught by the controller |
| 25 | If funds requested for withdrawal are insufficient, the controller must request the view to open a pop-up window that contains message "Insufficient funds: amount to withdraw is greater than available funds: y" |
| 26 | The pop-up window in 25 contains button "Dismiss" on pressing which the pop-up window shall close. |
| 27 | In the Account window(s), After pressing the "Deposit" button the content of the "Enter amount" textfield should be reset to 0.0 |
| 28 | In the Account window(s), After pressing the "Withdrawal" button the content of the "Enter amount" textfield should be reset to 0.0 |
| 29 | It shall be possible to keep multiple windows open for the same and/or different accounts in same and/or different currencies. |
| Exception Handling | |
| 30 | The system shall be designed with appropriate exception handling mechanisms |
| 31 | The system shall have other exception as needed (e.g. in response to corrupted or inconsistent content of the file with accounts; e.g. inadmissible characters in the fields of an account entry; broken format of the input file). |
| 32 | Exceptions shall result in useful messages (e.g. "Name must have only letters" or "Amount must not be negative") via dismissible pop-up windows (i.e. JDialog). |
| 33 | If an error is unrecoverable the system shall exit on dismissing the pop up window. |
| Documentation | |
| 34 | Comments should be provided for each class, important public methods and important code portions inside the methods. |
| 35 | Comments for classes and methods should be in Javadoc format so that a simple API documentation can be produced automatically. |

Mockup of Main Menu and Account Window



A mockup of a 'Main Menu' window. The title bar at the top reads 'Main Menu' and includes standard window controls (minimize, maximize, close). The window has a light gray background. On the left side, there is a vertical stack of three buttons: 'Edit in USD', 'Edit in Euros', and 'Edit in Yuan'. To the right of these buttons is a text input field containing the text '078910 Peter Benjamin'. At the bottom right of the window, there are two buttons: 'Save' and 'Exit'.

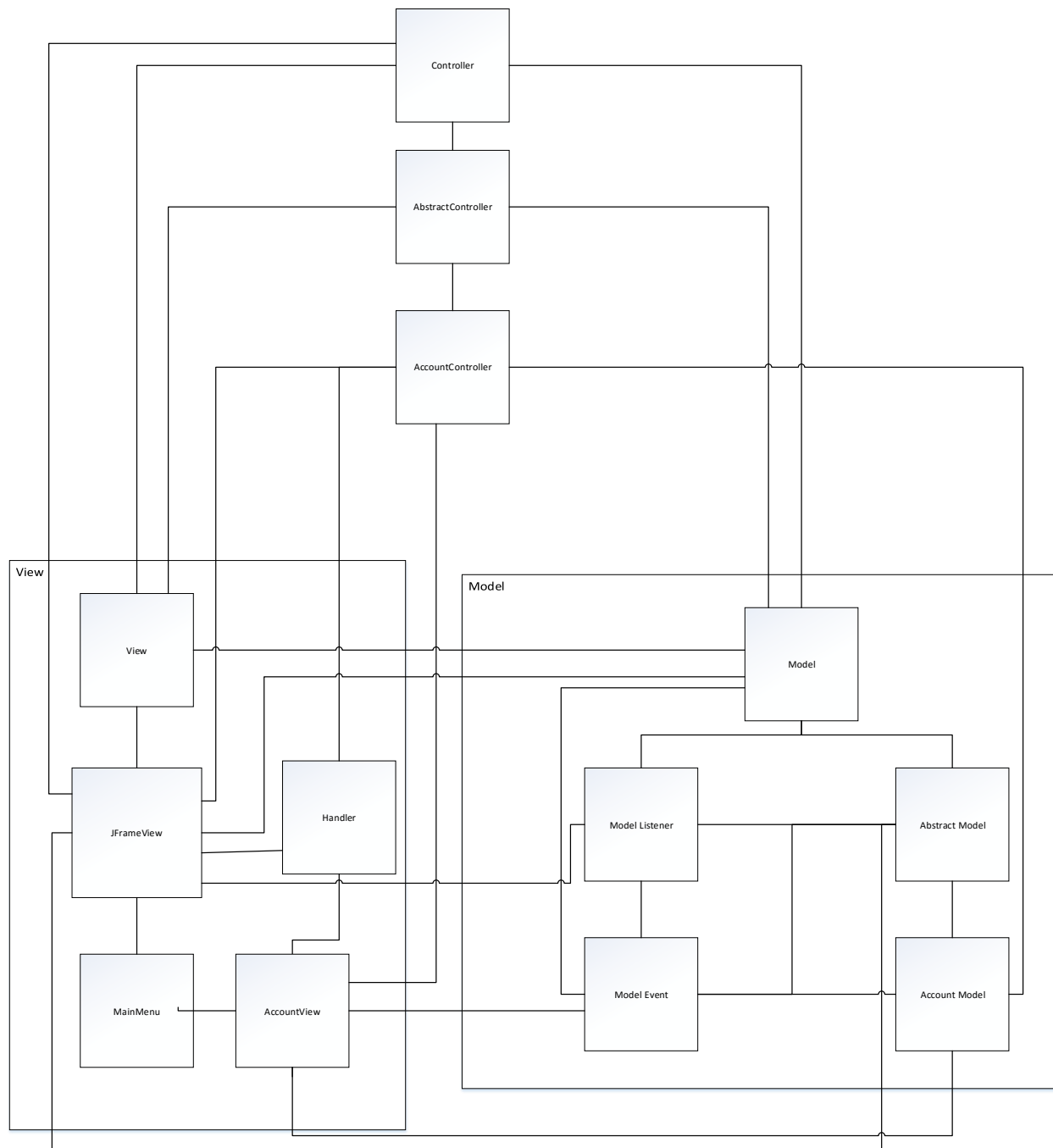


A mockup of an account window titled 'Peter Benjamin (USD)'. The title bar includes standard window controls. The main content area displays the text 'Your available funds: \$200.50' followed by the prompt 'Enter amount in USD'. Below this prompt is a text input field with the value '0.0'. At the bottom of the window, there are three buttons: 'Deposit', 'Withdraw', and 'Dismiss'.

Step 1: Identify Objects

| Noun List for potential classes | | | |
|-------------------------------------|--------------------------------------|-----------------------------|--------------------------------|
| Exchange rates | List of accounts | A file | Entry in a file |
| Frame | Drop down list | Account IDs with names | Edit in USD button |
| Edit in Euro button | Edit in Yuan button | Save button | Exit button |
| Current state of accounts | Main Menu window | Account window | Deposit button |
| Withdrawal button | Dismiss button | Editable textfield | Account amount |
| Result of operation | All account windows for this account | Exchange rate modifications | Consistent state |
| Positive number >1 | Sufficient funds | Amount | Account amount |
| Result of withdrawal | All other windows | Insufficient funds | Exception |
| Some method of a class | Model | Controller | View |
| Popup window | Message | Dismiss button | Deposit button |
| Content of 'enter amount' textfield | 0.0 | Multiple windows | Same and/or different accounts |
| Same and/or different currencies | Exception handling mechanisms | Useful messages | Dismissible popup windows |
| Unrecoverable error | | | |

Step 2: Create a Conceptual Model



Step 3: Identify Responsibilities

| Verb list to identify class responsibilities | | | |
|---|---|---|---|
| Receive updates | Use a push model | Programmed as constants | Load a list of accounts |
| Present the user (with a Main Menu window) | Pressing "Save" | Write the current state of accounts | Pressing "Exit" |
| Write the current state of accounts if they have been modified | Exit | Press "Edit in ..." button | Open account window |
| Press "Dismiss" | Window shall close | press "Deposit" | Add amount to the account amount if number >1 |
| The result must be seen in all other Account windows open for this. | all open windows show consistent account state | press "Withdraw" | Subtract amount from account amount if number >1 and there is sufficient funds |
| If insufficient funds, then raise exception | If insufficient funds, the controller must request the view to open a pop-up window | Pressing "Dismiss" in the popup window the pop-up window shall close. | Press "Deposit" button the content of the "Enter amount" textfield resets to 0.0 |
| Press "Withdrawal" button the content of the "Enter amount" textfield resets to 0.0 | Keep multiple windows open for the same and/or different account in same and/or different currencies. | Appropriate exceptions are raised with exception handling mechanisms | exceptions are raised as needed (e.g. corrupted/inconsistent content of the file with accounts; e.g. inadmissible characters in the fields of an account entry; broken format of the input file). |
| Exceptions emit useful messages | Exit on dismissing the pop up window if error is unrecoverable. | | |

Step 4: Assign Responsibilities in Conceptual Model

