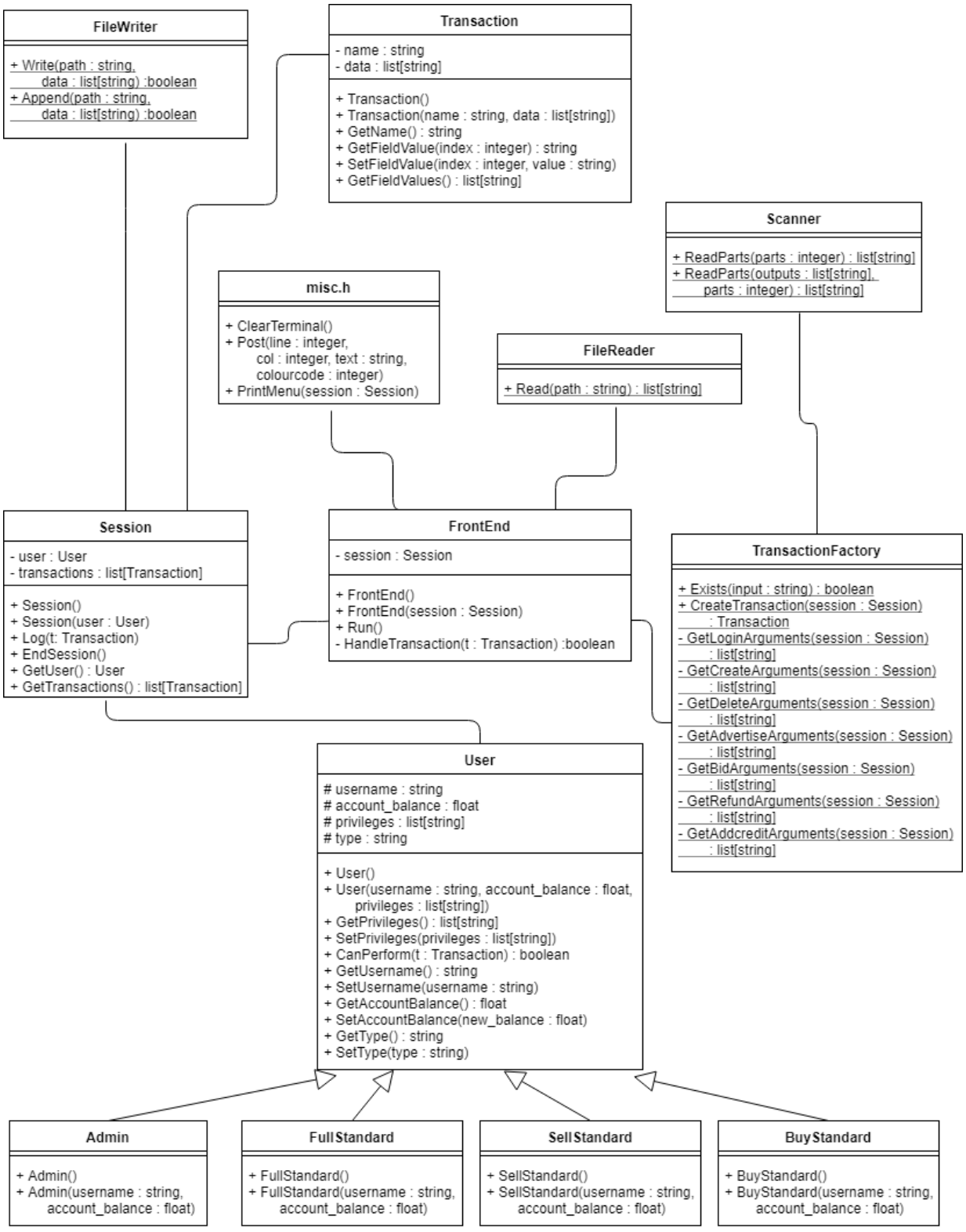
Front End Design Document - Genesis

Contents

UML Diagram


FileWriter
+ Write(path : string,
data : list[string] :boolean
+ Append(path : string,
data : list[string] :boolean
Transaction
- name : string
- data : list[string]
+ Transaction()
+ Transaction(name : string, data : list[string])
+ GetName() : string
+ GetFieldValue(index : integer) : string
+ SetFieldValue(index : integer, value : string)
+ GetFieldValues() : list[string]
Scanner
+ ReadParts(parts : integer) : list[string]
+ ReadParts(outputs : list[string],
parts : integer) : list[string]
misc.h
+ ClearTerminal()
+ Post(line : integer,
col : integer, text : string,
colourcode : integer)
+ PrintMenu(session : Session)
FileReader
+ Read(path : string) : list[string]
Session
- user : User
- transactions : list[Transaction]
+ Session()
+ Session(user : User)
+ Log(t: Transaction)
+ EndSession()
+ GetUser() : User
+ GetTransactions() : list[Transaction]
FrontEnd
- session : Session
+ FrontEnd()
+ FrontEnd(session : Session)
+ Run()
- HandleTransaction(t : Transaction) :boolean
TransactionFactory
+ Exists(input : string) : boolean
+ CreateTransaction(session : Session)
: Transaction
- GetLoginArguments(session : Session)
: list[string]
- GetCreateArguments(session : Session)
: list[string]
- GetDeleteArguments(session : Session)
: list[string]
- GetAdvertiseArguments(session : Session)
: list[string]
- GetBidArguments(session : Session)
: list[string]
- GetRefundArguments(session : Session)
: list[string]
- GetAddcreditArguments(session : Session)
: list[string]
User
username : string
account_balance : float
privileges : list[string]
type : string
+ User()
+ User(username : string, account_balance : float,
privileges : list[string])
+ GetPrivileges() : list[string]
+ SetPrivileges(privileges : list[string])
+ CanPerform(t : Transaction) : boolean
+ GetUsername() : string
+ SetUsername(username : string)
+ GetAccountBalance() : float
+ SetAccountBalance(new_balance : float)
+ GetType() : string
+ SetType(type : string)
Admin
+ Admin()
+ Admin(username : string,
account_balance : float)
FullStandard
+ FullStandard()
+ FullStandard(username : string,
account_balance : float)
SellStandard
+ SellStandard()
+ SellStandard(username : string,
account_balance : float)
BuyStandard
+ BuyStandard()
+ BuyStandard(username : string,
account_balance : float)

Functions in Classes

(main.cpp)

- Contains the *main* function
- Starts the *FrontEnd*

(misc.h)

Post(line : integer, col : integer, text : string, colour_code : integer)	Prints a line of text to the terminal in the specified location.
ClearTerminal()	Clears the terminal.
PrintMenu(session : Session)	Prints the relevant menu for the user.

Classes

FileReader	
Read(path : string) : list[string]	Reads data from a file and returns the list of lines in the file.
FileWriter	
Write(path : string, data : list[string]) : boolean	Writes data to a file. Returns true if the file write was successful.
Append(path : string, data : list[string]) : boolean	Appends data to a file. Returns true if the file write was successful.
FrontEnd	
Run()	Runs the front-end and manages the session.
HandleTransaction(t : Transaction) : boolean	Takes a transaction and performs it. This is the method where all transactions are processed. Returns true if and only if the transaction

	was performed.
Scanner	
ReadParts(parts : integer) : list[string]	Reads the specified number of lines from the terminal. Returns a list of lines entered by the user of length *parts*.
ReadParts(outputs : list[string], parts : integer) : list[string]	Reads the specified number of lines from the terminal and also for each line puts the corresponding output prompt text. Returns a list of lines entered by the user of length *parts*.
Session	
Log(t : Transaction)	Logs a valid transaction that was performed in the session.
EndSession()	Ends the session and writes to the *Daily Transaction* file.
GetUser() : User	Gets the user of this session.
GetTransactions() : list[Transaction]	Gets the list of valid transactions performed in the session, starting with login.
Transaction	
GetName() : string	Gets the name of the transaction (e.g. “login”)
GetFieldValue(index : integer) : string	Gets the specified field index (e.g. in the transaction login[CRLF]User001, index 0 would be “User001”)
SetFieldValue(index : integer, value : string)	Sets the field value at the specified index.
GetFieldValues() : list[string]	Gets the list of field values in the transaction.
TransactionFactory	
Exists(input : string) : boolean	Checks that the string starts with the name of a transaction (e.g. ‘login’). A transaction name is the set of characters from index 0 to the new-line character (or end of string). Returns true if and only if the transaction name exists.
CreateTransaction(session : Session) : Transaction	Attempts to create a transaction from user input using the *Scanner* class. Returns a Transaction object representing what the user entered.
Get<transaction_nam	Gets a valid list of arguments for the transaction based on the name.

e>Arguments(session : Session) : list[string]	
User	
CanPerform(t : Transaction) : boolean	Checks if the user can perform a given transaction.
GetUsername() : string	Gets the username of the user.
SetUsername(username : string)	Sets the username of the user. Ensures the username is at most 15 characters.
GetAccountBalance() : float	Gets the current account balance.
SetAccountBalance(new_balance : float)	Sets the current account balance. Ensures the balance is between 0 and 999,999.99 (inclusive).
GetPrivileges() : list[string]	Gets the user's transaction privileges.
SetPrivileges(privileges : list[string])	Sets the transaction privileges for the user.
Admin (inherits from User)	
FullStandard (inherits from User)	
BuyStandard (inherits from User)	
SellStandard (inherits from User)	

Class Descriptions

FileReader

The FileReader class reads data from a file.

FileWriter

The FileWriter class writes and appends data to a file.

FrontEnd

The FrontEnd class is responsible for managing the front-end of the program. It continuously gets transactions from the user and handles them.

Scanner

The Scanner class gets input from the user of a specified length in lines.

Session

The Session class keeps track of the user's session, tracking valid transactions. At the end of the session, the data for the *Daily Transaction* file is generated and appended to the file.

Transaction

The Transaction class represents a transaction (valid or not) and contains different data for the transaction's fields.

TransactionFactory

The TransactionFactory class is responsible for generating transactions and ensuring they exist.

User

The User class represents a user of the system. It checks if a transaction can be performed based on the privileges given to that user.

Admin (inherits from User)

The Admin class inherits from the User class and simply creates a User with all privileges.

FullStandard (inherits from User)

The FullStandard class inherits from the User class and simply creates a User which can:

- login
- logout
- advertise
- bid
- addcredit

BuyStandard (inherits from User)

The BuyStandard class inherits from the User class and simply creates a User which can:

- login
- logout
- bid
- addcredit

SellStandard (inherits from User)

The SellStandard class inherits from the User class and simply creates a User which can:

- login
- logout
- advertise
- addcredit

Program Flow

1. main (main.cpp): Starts an instance of the front-end with the *FrontEnd* class and calls *FrontEnd.Run()*
2. FrontEnd.Run(): continuously loops
3. FrontEnd.Run(): calls *TransactionFactory.CreateTransaction(user)*
4. CreateTransaction(...): gets transaction name from *Scanner* class
5. CreateTransaction(...): checks if transaction exists with Exists(input)
6. CreateTransaction(...): uses the *Scanner* class to get the appropriate number of fields for the transaction
7. FrontEnd.Run(): receives the *Transaction* from *TransactionFactory*
8. FrontEnd.Run(): calls *Session.Log(t)* and *FrontEnd.HandleTransaction(t)*
9. FrontEnd.HandleTransaction(...): performs the appropriate action, for example:
    a. Login: starts a session if no user is logged in
    b. Logout: if a user is logged in, logout and call *Session.EndSession()*
10. Repeat 2 - 9 until the user quits