Finance Tracker

Software Design Document

Pablo Bandera Lopez

April 21, 2025

CS 225, Spring 2025

Embry-Riddle Aeronautical University

Daytona Beach campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

INTRODUCTION:

The program developed will be a personal monthly finance tracker to help the user keep track of their expenses throughout a given month. The user should be able to interact with the application through the use of a multi-windowed GUI. The user will be able to specify a set of data per entry including: a user entered amount, user selected type of transaction from a predetermined list, category from a predetermined list, and relevant sub-categories from a predetermined list. The goal of this data is to allow the user to refer to any entry and immediately identify what the transaction was. Banking software and similar generic finance trackers don't keep specific enough data for transactions. Many times, payment processors have generic names and payments may take a few days to be posted to an account which is sufficient time for the user to no longer be able to determine what the charges or payments were for. This document is a guide describing the expected functionality of the program. It will discuss data collection & storage, data retrieval, data calculation, and data display. This way a user will be able to keep track of all monthly transactions and have totals as well as individual entries displayed to them.

PROBLEM DESCRIPTION:

Upon startup, the application will display a Main Menu (MM) for the user to select from. The menu will contain the different actions the program can perform. The options shall include at minimum the following: New Entry (NE), Edit Entry (EE), Delete Entry (DE), View Current Month (VCM), View Past Month (VPM), New Month (NM), and Clear Month (CM).

On user selection of NE, the user will be prompted to create a new entry. The data to be collected for every entry is as follows: USD amount (double), type of entry, category, any relevant subcategories, account to be charged, and a free text comment for the user to add a description.

Table 1: Predetermined List of Types

Types of Entries:	Expense, Income

Table 2: Predetermined List of Accounts

Accounts	Personal, Flight, Mother

Table 3: Predetermined List of Categories

Types of Entry:	**Available Categories:**
Income	Pay Stub, Mother, Other
Expense	Living, Purchase, Flight, School

Table 4: Predetermined List of Subcategories Level 1 Depending on the Category Selected.

Selected Category	Available Sub-category level 1
Living	Apartment, Food, Car
Purchase	Personal, Travel, Paperwork
Flight	Medical, Currency, Fun
School	Tuition, Flight Training, Books

Table 5: Predetermined List of Subcategories Level 2 Depending on the Subcategory Level1 Selected.

Selected Sub-Category Level 1	Available Sub-category level 2
Apartment	Rent, Expense
Food	Groceries, Food Out, Snacks
Car	Insurance, gas, tolls, service, cleaning
Flight Training	Training, Exam, Paperwork

Table 6: Predetermined List of Subcategories Level 3 Depending on the Subcategory Level 2 Selected.

Selected Sub-Category Level 2	Available Sub-category level 3
Rent	Base Rent, Utilities
Expense	Cleaning Supplies, Furniture, Kitchen
Groceries	Publix, Bakery, Convenience
Food Out	Eat Out, Take Out, Uber Eats
Snacks	Riddle, Coffee, Ice-Cream, Other

Table 6: Predetermined List of Subcategories Level 4 Depending on the Subcategory Level 3 Selected.

Selected Sub-Category Level 3	Available Sub-category level 4
Base Rent	none
Utilities	Power, Wifi, Cell, Subscriptions, Gas, Water
Eat Out	Riddle, Restaurant

In order to create an entry, the user must enter a dollar value greater than 0, select a type, select a category, if there are any available sub-categories; select a sub-category for every level available, select an account to be charged or deposited to, and a non-empty comment. Every entry shall be sequentially number from the previous entry. The user shall have the option to cancel the entry or submit the entry. If the entry is properly created and the user submits the entry, it shall be stored in the corresponding months' text file. If the user cancels, the entry shall be forgotten. Once the corresponding action is complete, the user will be returned to the MM.

On user selection of EE, the program shall prompt the user for an entry number. If an existent entry number is entered, the program will read the entry and display it to the user. The user shall be able to edit each of the parameters as they did during the entry creation. The user shall have the option to cancel the editing or submit the correction. If the entry is complete per the criteria

required to create a new entry and the user selects to submit, the entry shall be stored in the corresponding months' text file, overriding the previous entry of that number. If the user opts to cancel the editing, the program shall forget the corrected entry. Once the corresponding action is complete, the user will be returned to the MM.

On user selection of DE, the program shall prompt the user for an entry number. If there is an existent entry with that number in the months' corresponding text file, it shall be replaced with a null entry, keeping the line in the file. The user shall be notified of the deletion status and returned to the MM.

On user selection of VCM, the program shall retrieve all entries from the current months' file. Once file has been read, the entries shall be displayed in a neat table in plain text. Any NULL entries shall be omitted from the table. The program shall also display a running total for each account taking the remaining balance of the past month updating it with every single entry of the current month. The user shall have the option to close the table and go back to the main menu at any moment.

On user selection of VPM, the user will be prompted to select a month file from those found in the directory. The user shall have the option to select or cancel. If the user cancels, they shall be returned to the main menu. If they select a month and continue, the program shall retrieve all entries form that months' file. Once the file is read, the entries shall be displayed in a neat table in plain text. Anu NULL entries shall be omitted from the table. The program shall display the total income and expenses of each account for that month. The user shall have the option to close the table and go back to the main menu at any moment.

On user selection of NM, the program shall fetch the current system date and create a file named after the date month, located inside a directory named after the date year, in the /Files/ directory of the program. If there is already an existent month file for the current month, the user shall be notified and given the option to override it. Upon completion the user shall be returned to the MM.

On user selection of CM, the program shall fetch the current system date. It will look for a file named after the date month, located in a directory named after the date year, in the /Files/ directory of the program. If the file does not exist, the user shall be notified and returned to the MM. If the file is found, the program will delete the file and notify the user of the file deletion. Upon completion the user will be returned to the MM.

The only way to exit the program will be for the user to click the x button of the program window. It shall run continuously until the user decides to exit.

PROBLEM SOLUTION:

Textual Description of Classes:

FinanceTracker – Contains Main function and acts as the entry point for the application by instantiating the main menu (MM).

AppFrame – Abstract class that contains the blueprint for an application frame. It contains attributes such as the dimensions of the application and various settings for fonts, color schemes, etc. This class extends JFrame, a built-in class from Java swing.

MainFrame – Class extends AppFrame. Creates the main menu buttons. Implements ActionListener in order to implement the event handlers for the buttons. Identifies the source of a button press and executes the according action.

EntryFrame – Class extends AppFrame. Creates a frame with a series of text-inputs and combo boxes for the user to create an entry from. Offers the user the ability to submit or cancel an entry. Is instantiated by the event handler in MainFrame. If all fields are properly filled out and the user submits the Entry will be added to the Entries collection class.

DisplayFrame – Class extends AppFrame. Creates a frame with a table and 2 labels. The table will be loaded from the Entries class and the totals for income and expenses will be displayed on the two labels. The user has the option to close the frame and return to the MM.

LayoutMismatchException – Type of exception to be thrown if an instance of a child of AppFrame attempts to add a JComponent to a JPanel which has a different Layout.

InvalidEntryException – Type of exception to be thrown when invalid data is being entered into an entry.

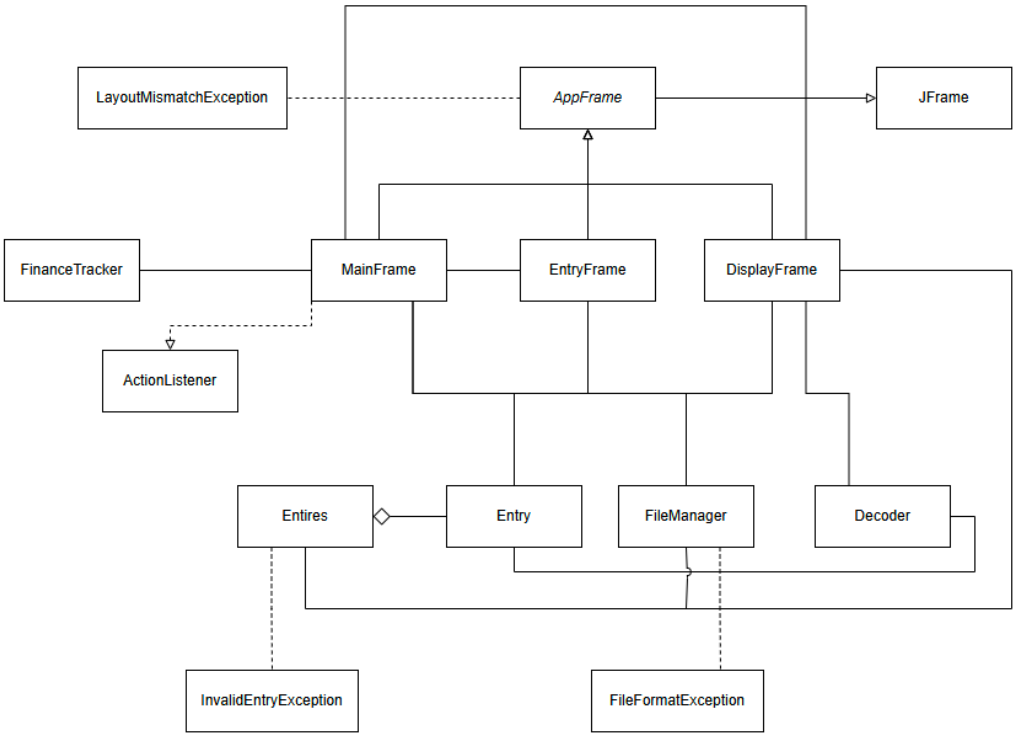
Entry – Class contains a series of attributes corresponding to the data to be stored for every transaction (i.e. double Amount, int type, int category, etc.). Setters and getters are available for all attributes. Every new entry is assigned a sequential ID based on the previous entry, if no previous entries exist, it is assigned an ID of 1.

Entries – Contains an ArrayList<Entry> containing all the entries. Acts as a dynamic storage during application runtime. Contains methods to add and remove entries which in term update the corresponding data file using an instance of the FileManager class.

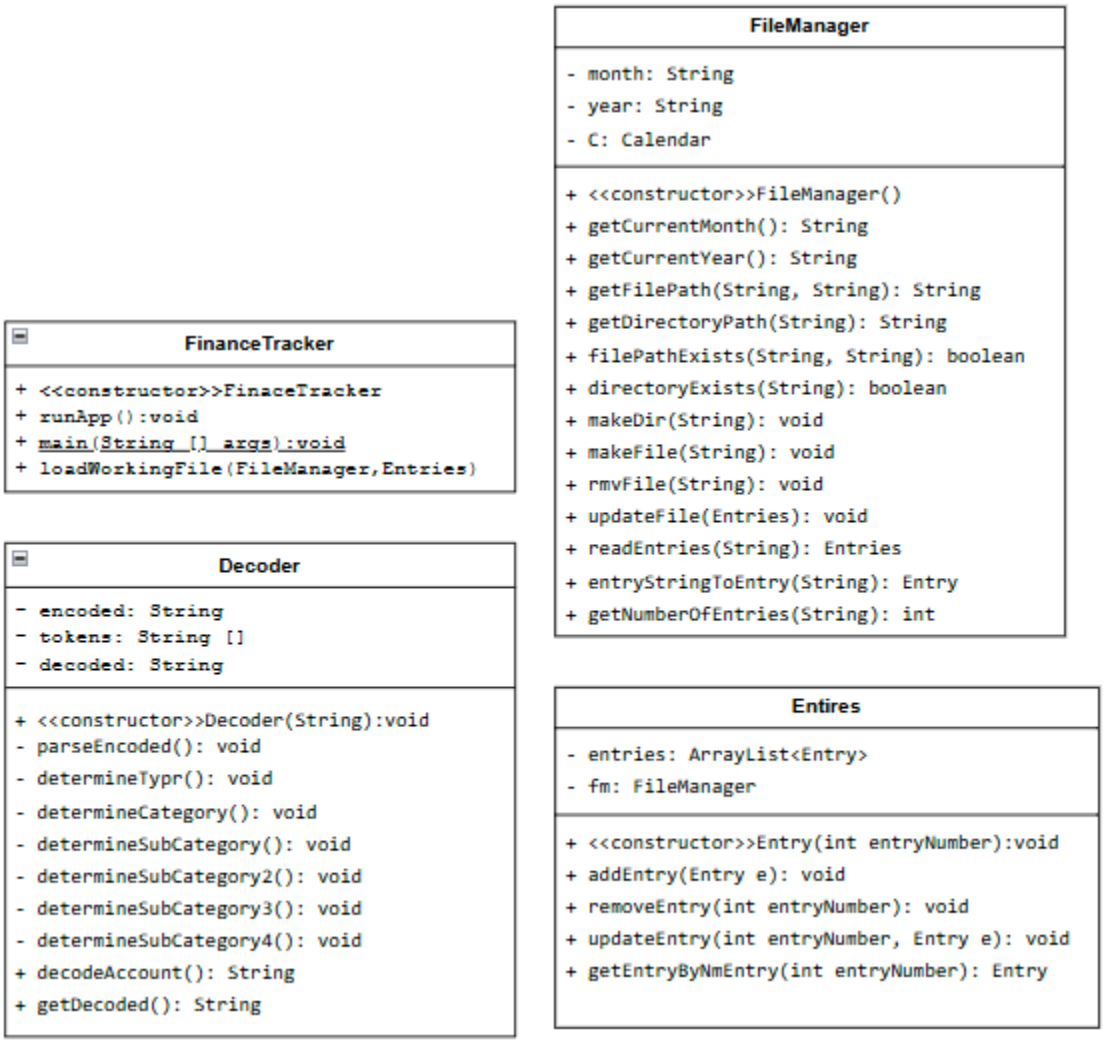
FileManager – Class contains methods for file I/O. It acts as the bridge between the Entries class and the data files. Manages creation of appropriate directories and files.

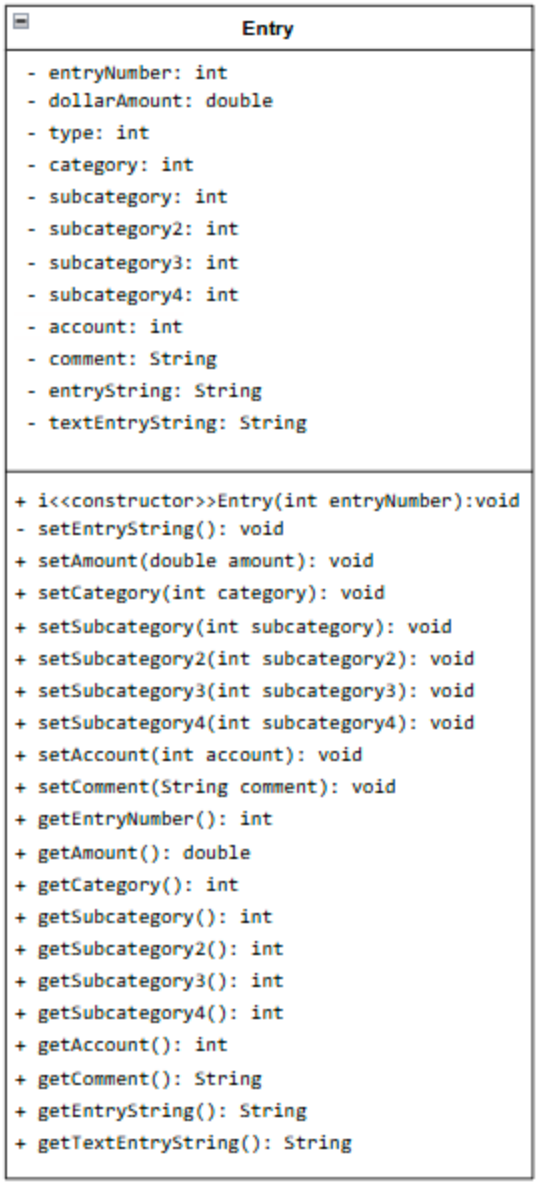
SelectionMap – Class contains a HashMap to be able to decode the index data stored in the Entry and translate it to type String. This data will be displayed in text form for the View Month and View Past Month operations.

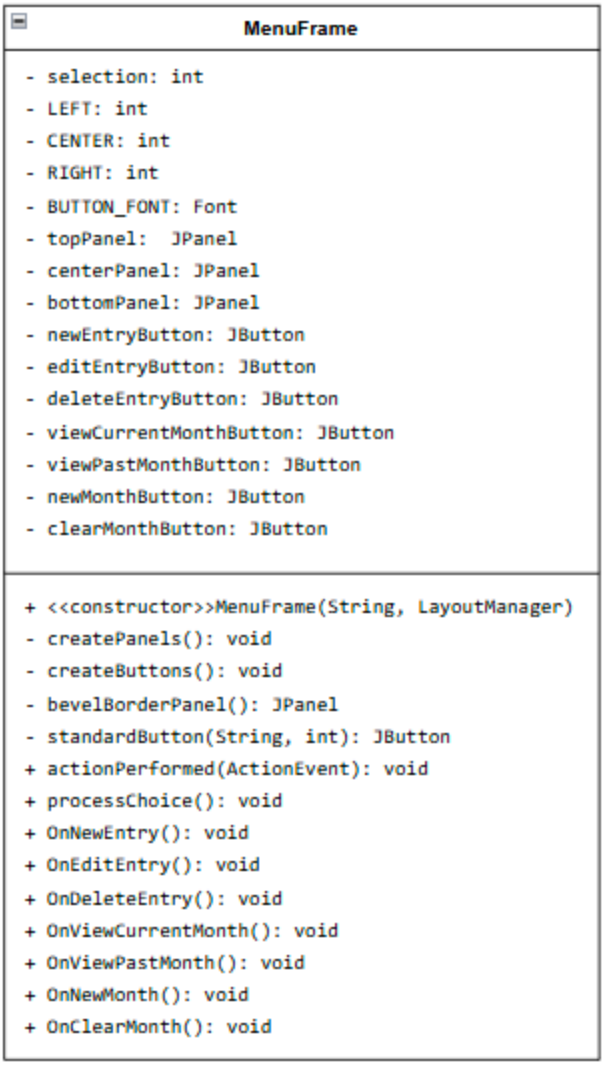
TOP LEVEL UML



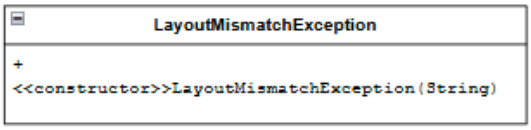
UML CLASSES:


FileManager
- month: String
- year: String
- C: Calendar
+ <<constructor>>FileManager()
+ getCurrentMonth(): String
+ getCurrentYear(): String
+ getFilePath(String, String): String
+ getDirectoryPath(String): String
+ filePathExists(String, String): boolean
+ directoryExists(String): boolean
+ makeDir(String): void
+ makeFile(String): void
+ rmvFile(String): void
+ updateFile(Entries): void
+ readEntries(String): Entries
+ entryStringToEntry(String): Entry
+ getNumberOfEntries(String): int
FinanceTracker
+ <<constructor>>FinaceTracker
+ runApp():void
+ main(String [] args):void
+ loadWorkingFile(FileManager,Entries)
Decoder
- encoded: String
- tokens: String []
- decoded: String
+ <<constructor>>Decoder(String):void
- parseEncoded(): void
- determineTypr(): void
- determineCategory(): void
- determineSubCategory(): void
- determineSubCategory2(): void
- determineSubCategory3(): void
- determineSubCategory4(): void
+ decodeAccount(): String
+ getDecoded(): String
Entires
- entries: ArrayList<Entry>
- fm: FileManager
+ <<constructor>>Entry(int entryNumber):void
+ addEntry(Entry e): void
+ removeEntry(int entryNumber): void
+ updateEntry(int entryNumber, Entry e): void
+ getEntryByNmEntry(int entryNumber): Entry

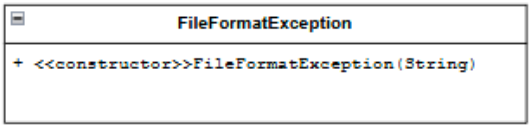

Entry
- entryNumber: int
- dollarAmount: double
- type: int
- category: int
- subcategory: int
- subcategory2: int
- subcategory3: int
- subcategory4: int
- account: int
- comment: String
- entryString: String
- textEntryString: String
+ i<<constructor>>Entry(int entryNumber):void
- setEntryString(): void
+ setAmount(double amount): void
+ setCategory(int category): void
+ setSubcategory(int subcategory): void
+ setSubcategory2(int subcategory2): void
+ setSubcategory3(int subcategory3): void
+ setSubcategory4(int subcategory4): void
+ setAccount(int account): void
+ setComment(String comment): void
+ getEntryNumber(): int
+ getAmount(): double
+ getCategory(): int
+ getSubcategory(): int
+ getSubcategory2(): int
+ getSubcategory3(): int
+ getSubcategory4(): int
+ getAccount(): int
+ getComment(): String
+ getEntryString(): String
+ getTextEntryString(): String

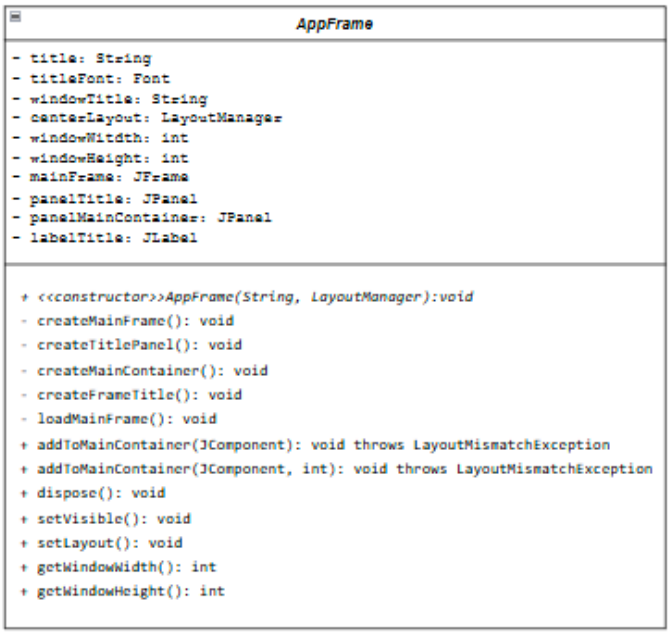

MenuFrame
- selection: int
- LEFT: int
- CENTER: int
- RIGHT: int
- BUTTON_FONT: Font
- topPanel: JPanel
- centerPanel: JPanel
- bottomPanel: JPanel
- newEntryButton: JButton
- editEntryButton: JButton
- deleteEntryButton: JButton
- viewCurrentMonthButton: JButton
- viewPastMonthButton: JButton
- newMonthButton: JButton
- clearMonthButton: JButton
+ <<constructor>>MenuFrame(String, LayoutManager)
- createPanels(): void
- createButtons(): void
- bevelBorderPanel(): JPanel
- standardButton(String, int): JButton
+ actionPerformed(ActionEvent): void
+ processChoice(): void
+ OnNewEntry(): void
+ OnEditEntry(): void
+ OnDeleteEntry(): void
+ OnViewCurrentMonth(): void
+ OnViewPastMonth(): void
+ OnNewMonth(): void
+ OnClearMonth(): void

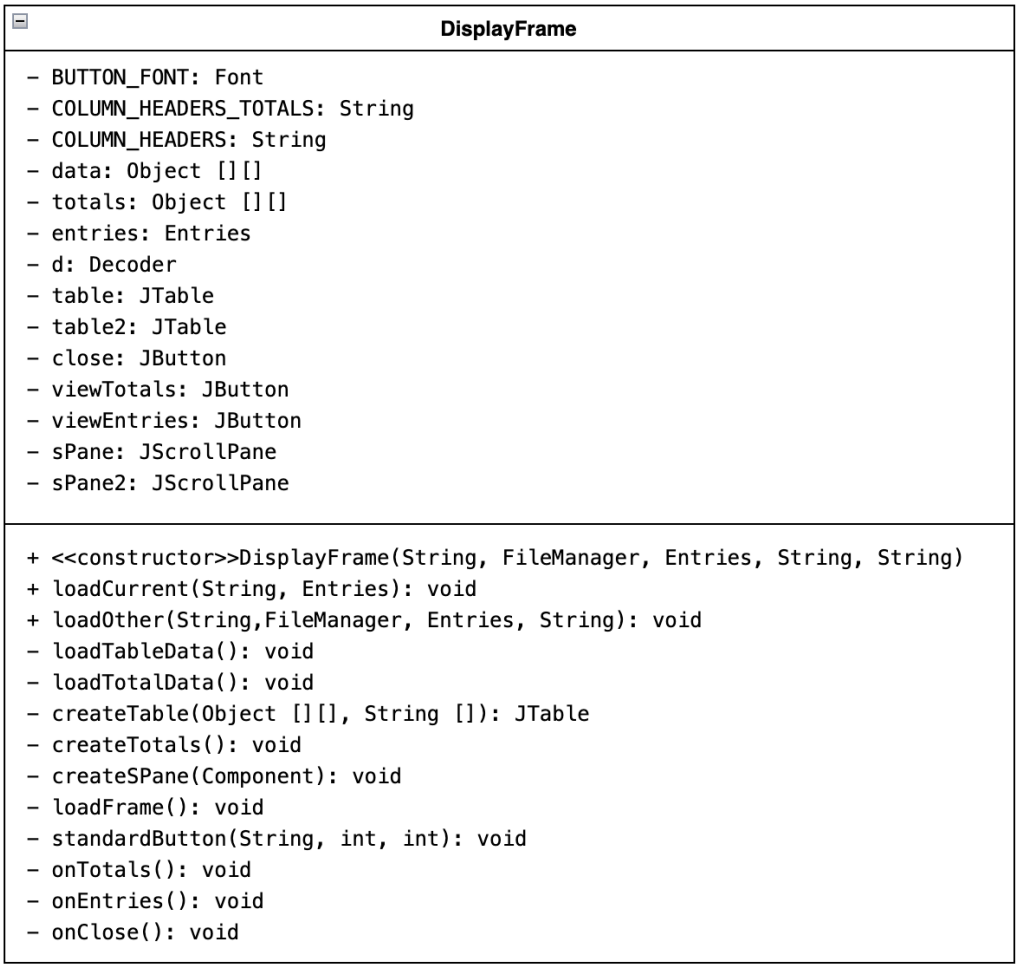

EntryFrame
- DEFAULT_FONT: Font
- TYPES: String[]
- CATEGORIES_INCOME: String[]
- CATEGORIES_EXPENSE: String[]
- ACCOUNTS: String[]
- SUBCAT1_EXPENSE_LIVING: String[]
- SUBCAT1_EXPENSE_PURCHASE: String[]
- SUBCAT1_EXPENSE_FLIGHT: String[]
- SUBCAT1_EXPENSE_SCHOOL: String[]
- SUBCAT2_LIVING_APPARTMENT: String[]
- SUBCAT2_LIVING_FOOD: String[]
- SUBCAT2_LIVING_TRANSPORTATION: String[]
- SUBCAT2_SCHOOL_FLIGHTTRAINING: String[]
- SUBCAT3_APPARTMENT_RENT: String[]
- SUBCAT3_APPARTMENT_EXPENSE: String[]
- SUBCAT3_FOOD_GROCERIES: String[]
- SUBCAT3_FOOD_FOODOUT: String[]
- SUBCAT3_FOOD_SNACKS: String[]
- SUBCAT4_RENT_UTILITIES: String[]
- SUBCAT4_FOODOUT_EATOUT: String[]
- hasSubCat: boolean
- hasSubCat1: boolean
- hasSubCat2: boolean
- hasSubCat3: boolean
- hasSubCat4: boolean
- panel1, panel2, panel3, panel4, panel5, panel6, panel7, panel8, panel9, panel10: JPanel
- label1, label2, label3, label4, label5, label6, label7, label8: JLabel
- textFieldAmount: JTextField
- textAreaComment: JTextArea
- comboBoxCategory, comboBoxSubCat1, comboBoxSubCat2,
comboBoxSubCat3, comboBoxSubCat4, comboBoxType, comboBoxAccount: JComboBox<String>
- submitButton, cancelButton: JButton
- entry: Entry
+ <<constructor>>EntryFrame(String, LayoutManager, Entry, Entries)
- loadEntry(): void
- resetCategory(): void
- resetSubCats(): void
- resetSubCats2On(): void
- resetSubCats3On(): void
- resetSubCats4ON(): void
- setCategories(): void
- setSubCat1(): void
- setSubCat2(): void
- setSubCat3(): void
- setSubCat4(): void
- initFlags(): void
- resetFlags(int): void
- initPanels(): void
- initTextField(): void
- initTextArea(): void
- initComboBoxes(): void
- createPanels(): void
- createLabels(): void
- createButtons():void
- createComboBoxs(): void
- createPanel(): JPanel
- creatLabel(String): JLabel
- creaTextField(): JTextField
- creatTextArea(): JTextArea
- createButton(String, int): JButton
- createComboBox(String[]): JComboBox<String>
- amountModified(): void
- commentModified(): void
- submitClicked(): void
- cancelClicked(): void
- typeSelected(): void
- categorySelected(): void
- subCat1Selected(): void
- subCat2Selected(): void
- subCat3Selected(): void
- subCat4Selected(): void
- accountSelected(): void


LayoutMismatchException
+
<<constructor>>LayoutMismatchException(String)


InvalidEntryException
+ <<constructor>>InvalidEntryException(String)


FileFormatException
+ <<constructor>>FileFormatException(String)


AppFrame
- title: String
- titleFont: Font
- windowTitle: String
- centerLayout: LayoutManager
- windowWidth: int
- windowHeight: int
- mainFrame: JFrame
- panelTitle: JPanel
- panelMainContainer: JPanel
- labelTitle: JLabel
+ <<constructor>>AppFrame(String, LayoutManager):void
- createMainFrame(): void
- createTitlePanel(): void
- createMainContainer(): void
- createFrameTitle(): void
- loadMainFrame(): void
+ addToMainContainer(JComponent): void throws LayoutMismatchException
+ addToMainContainer(JComponent, int): void throws LayoutMismatchException
+ dispose(): void
+ setVisible(): void
+ setLayout(): void
+ getWindowWidth(): int
+ getWindowHeight(): int


DisplayFrame
- BUTTON_FONT: Font
- COLUMN_HEADERS_TOTALS: String
- COLUMN_HEADERS: String
- data: Object [][]
- totals: Object [][]
- entries: Entries
- d: Decoder
- table: JTable
- table2: JTable
- close: JButton
- viewTotals: JButton
- viewEntries: JButton
- sPane: JScrollPane
- sPane2: JScrollPane
+ <<constructor>>DisplayFrame(String, FileManager, Entries, String, String)
+ loadCurrent(String, Entries): void
+ loadOther(String,FileManager, Entries, String): void
- loadTableData(): void
- loadTotalData(): void
- createTable(Object [][], String []): JTable
- createTotals(): void
- createSPane(Component): void
- loadFrame(): void
- standardButton(String, int, int): void
- onTotals(): void
- onEntries(): void
- onClose(): void

REFERENCES:

Oracle. (n.d.). *Trail: Creating a GUI with swing*. Trail: Creating a GUI With Swing (The JavaTM Tutorials). https://docs.oracle.com/javase/tutorial/uiswing/index.html

APPENDICES: