Cover Page

Java Budget Tracker

https://github.com/OctavianIII/OOPBudgetTracker

Group 9
Alex Slep

14 December 2023

Table of Contents

Terminology Glossary

- DataNode - a class that serves as a single node in the linked lists of linked lists structure. Therefore it contains pointers to two other linked lists.
- Data - a singleton class that keeps the head of the Linked List data structure and thus serves as the actual data structure which contains DataNodes.

System Analysis

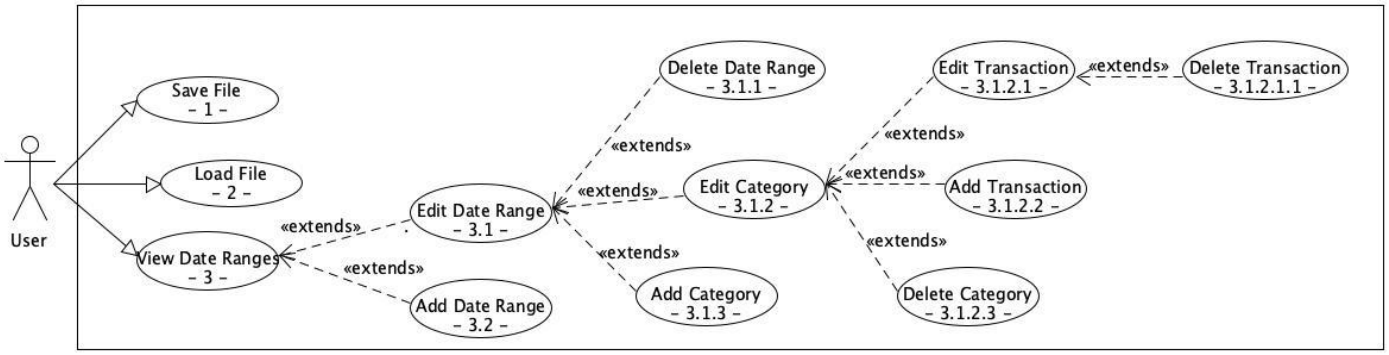
The goal of this project was to create an accounting application tailored to my needs. Prior to this project, I used Google Sheets for my budgeting. This naturally has a lot of shortfalls due to the broad scope. My goal was to create a more focused program that, due to its more specific focus, would better suit my needs.

In its simplest form, this program is a GUI interface for a tree structure that contains date ranges, which contain purchase categories, which in turn contain transactions. The GUI allows the user to easily navigate through the tree, adding and removing DataNodes. It also displays aggregate data about the user's spending as they navigate through the program. The other aspect of the program to note is the persistent storage system. As an accounting application, it is vital that the user be able to save their data and come back to it at a later date. To allow this functionality, the program provides the FileManager and DataParser classes, which work hand in hand to convert the Data to CSV format and to save it to a file, and to do the reverse upon loading a file.

An interesting benefit of this is that the CSV file the program saves can be imported into any spreadsheet program of the user's choice, allowing them to easily use the more powerful analytics tools of a spreadsheet should they ever have a need for a capability the program does not possess.

Moreover, the program is extremely reliable when the user is operating within defined bounds. However, little to no data entry validation is done, so it is quite easy to break the program by entering unexpected values. For example, since the data is stored as a CSV, entering commas in any of the fields corrupts the save. I decided against including data validation for the time being as it would add significant time to development. Also I'm the only one using the program regularly and I know how to avoid the issues it may cause and how to fix any that arise.

Use Case Diagram



UC Reference Name/Number:	Save File (- 1 -)
Overview	Saves the currently loaded data to a file
Related use cases:	None
Actors	User

UC Reference Name/Number:	Load File (- 2 -)
Overview	Loads budgeting data from a file on the user's system
Related use cases:	None
Actors	User

UC Reference Name/Number:	View Date Ranges (- 3 -)
Overview	Opens up the date ranges for viewing and editing
Related use cases:	Edit Date Range, Add Date Range
Actors	User

UC Reference Name/Number:	Edit Date Range (- 3.1 -)
Overview	Opens up a date range for viewing and editing
Related use cases:	View Date Ranges, Delete Date Range, Edit Category, Add Category
Actors	User

UC Reference Name/Number:	Delete Date Range (- 3.1.1 -)
Overview	Removes a date range
Related use cases:	Edit Date Range

Actors	User

UC Reference Name/Number:	Edit Category (- 3.1.2 -)
Overview	Opens up a category for viewing and editing
Related use cases:	Edit Date Range, Edit Transaction, Add Transaction
Actors	User

UC Reference Name/Number:	Edit Transaction (- 3.1.2.1 -)
Overview	Opens up a transaction for viewing and editing
Related use cases:	Edit Category, Delete Transaction
Actors	User

UC Reference Name/Number:	Delete Transaction (- 3.1.2.1.1 -)
Overview	Removes a transaction
Related use cases:	Edit Transaction
Actors	User

UC Reference Name/Number:	Add Transaction (- 3.1.2.2 -)
Overview	Adds a new transaction to the open category
Related use cases:	Edit Category
Actors	User

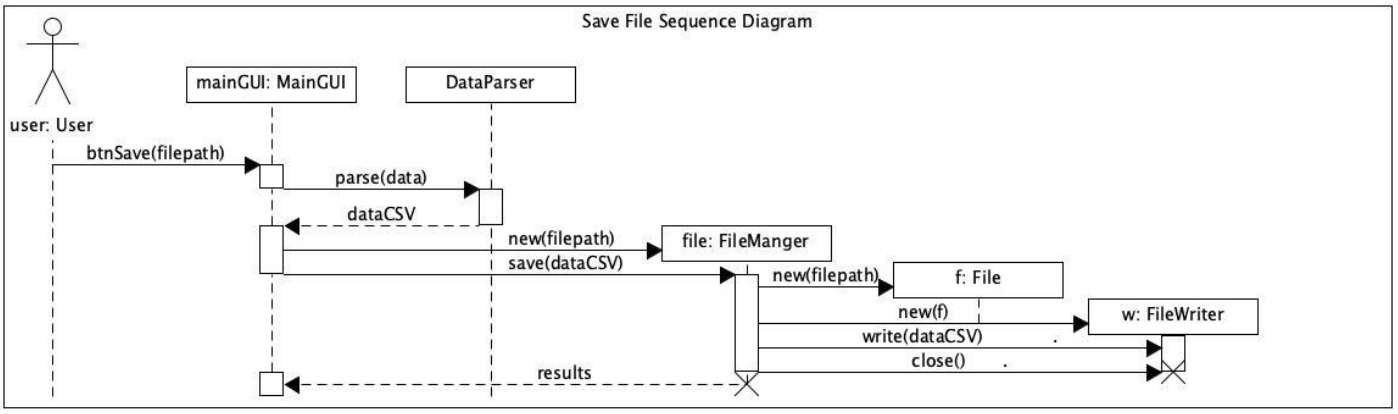
UC Reference Name/Number:	Delete Category (- 3.1.2.3 -)
Overview	Removes a category
Related use cases:	Edit Category
Actors	User

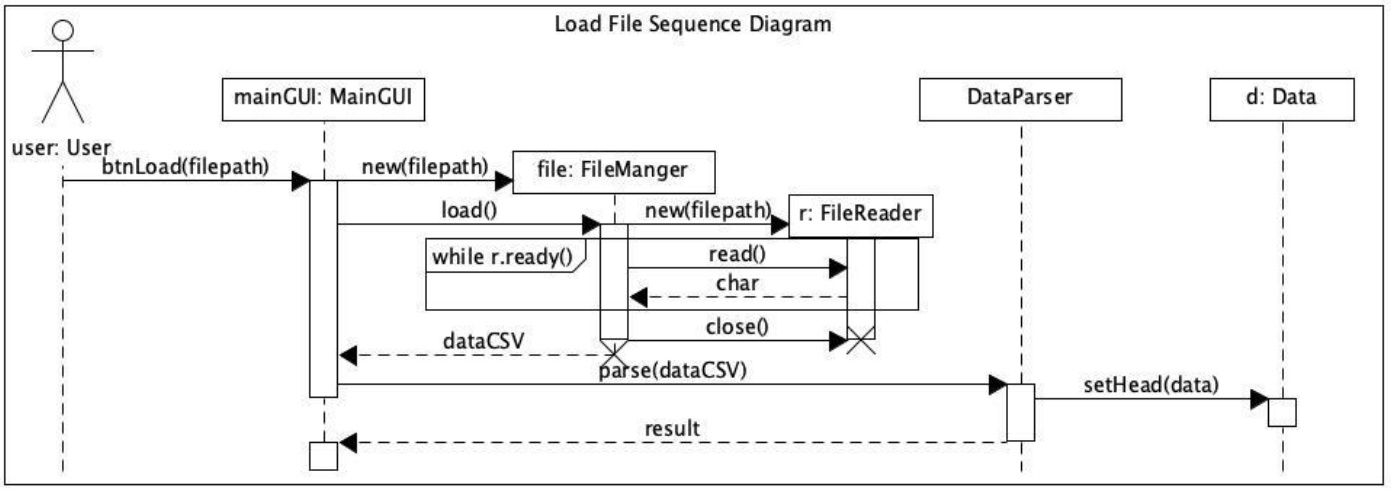
UC Reference Name/Number:	Add Category (- 3.1.3 -)
Overview	Adds a new category
Related use cases:	Edit Date Range
Actors	User

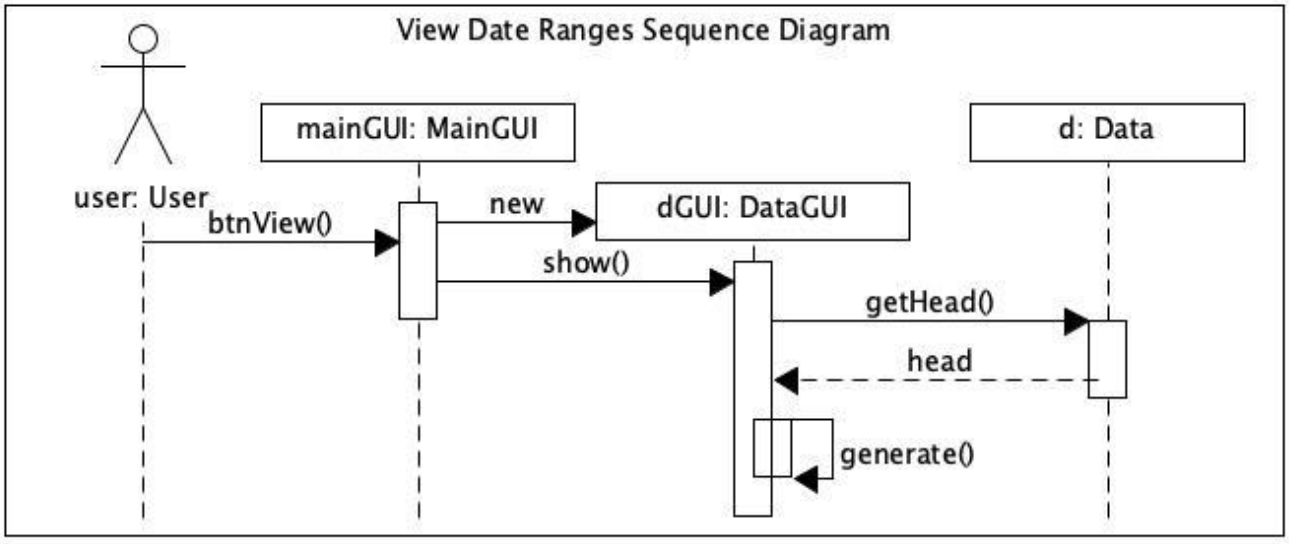
UC Reference Name/Number:	Add Date Range (- 3.2 -)

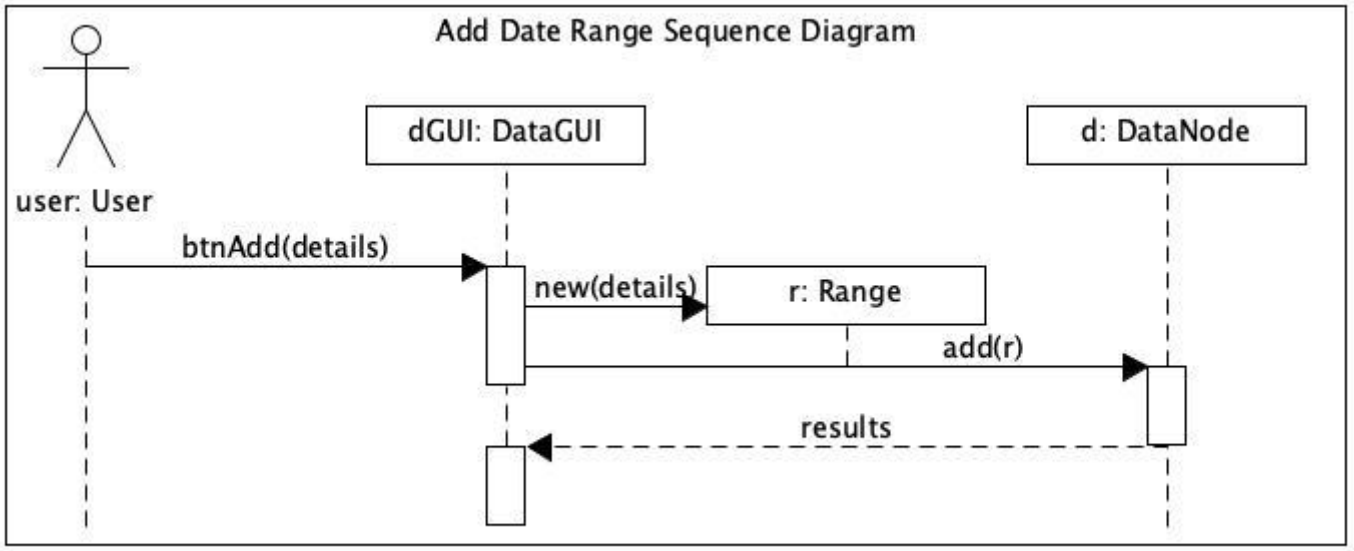
Overview	Adds a new date range
Related use cases:	View Date Ranges
Actors	User

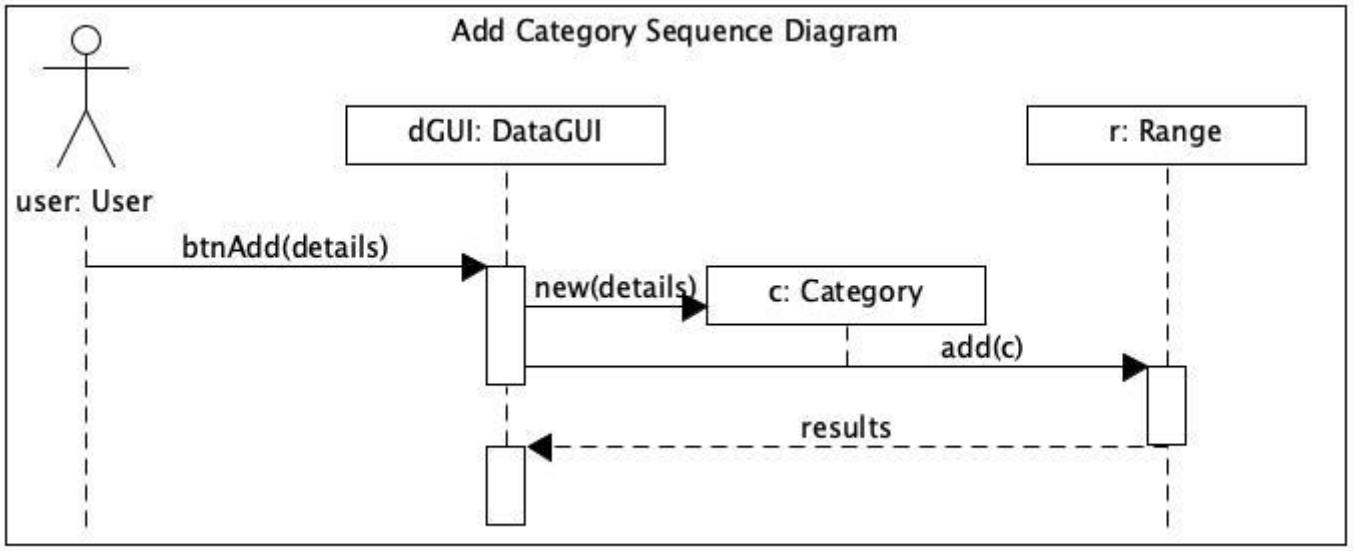
System Design

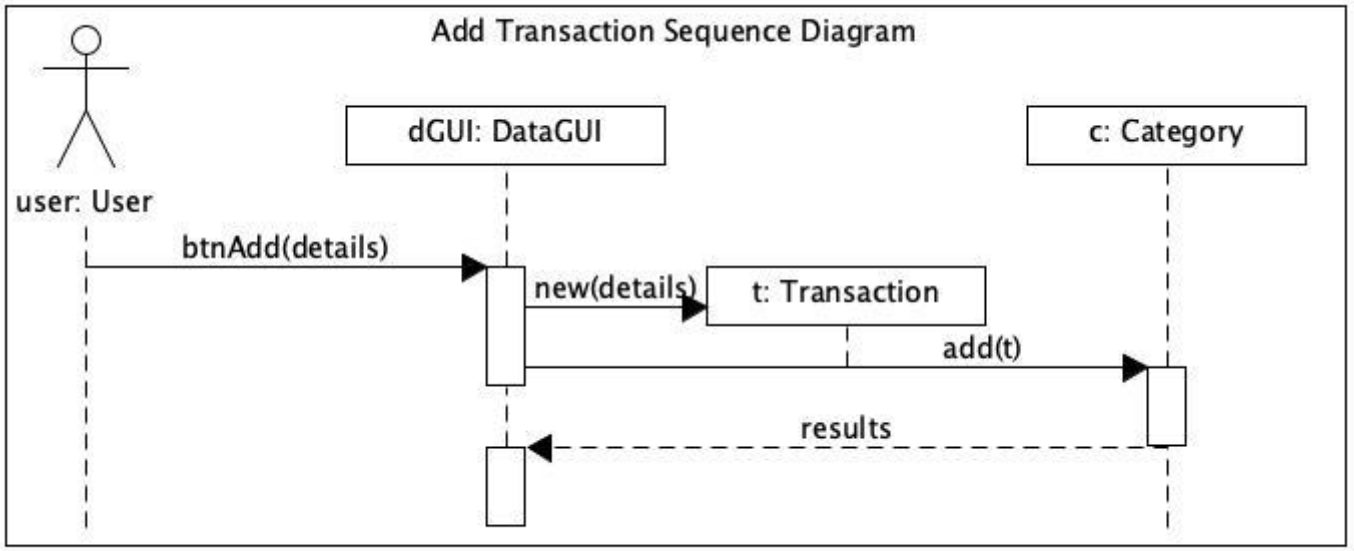


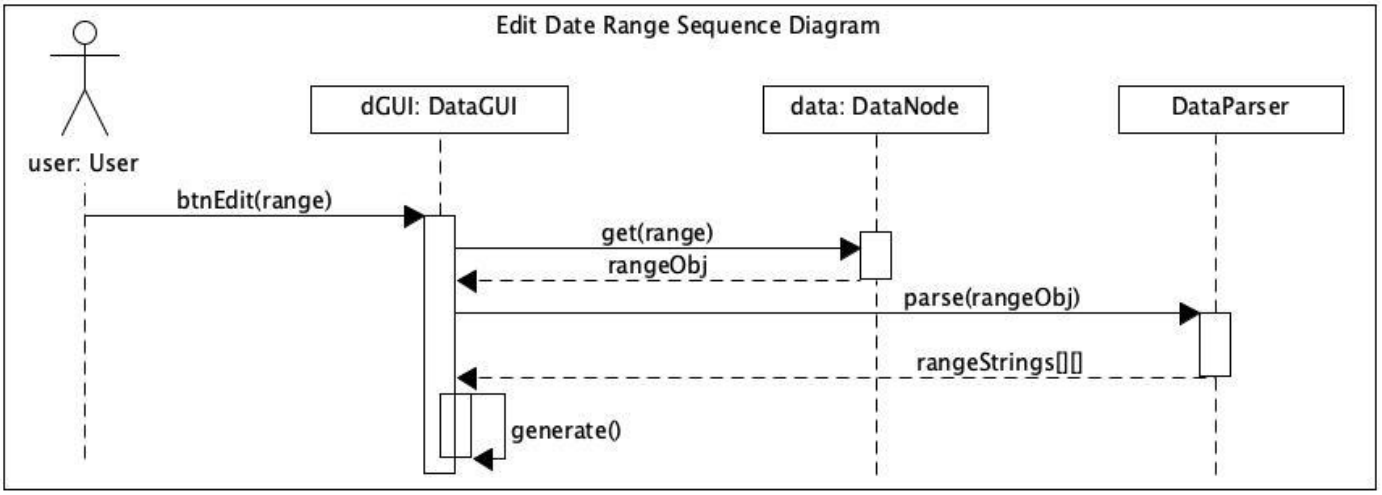


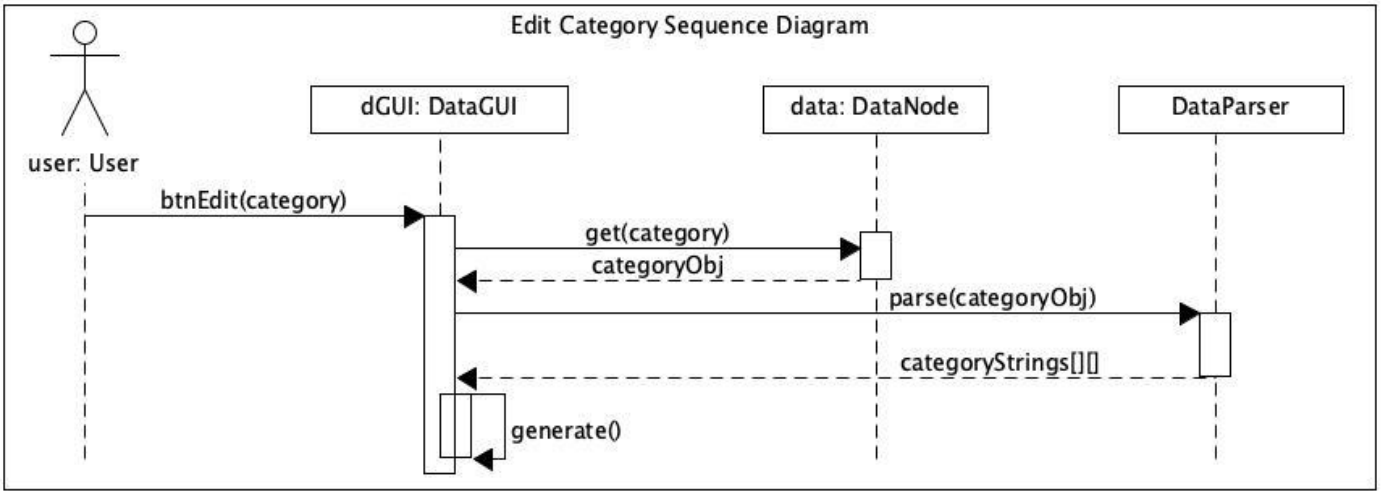


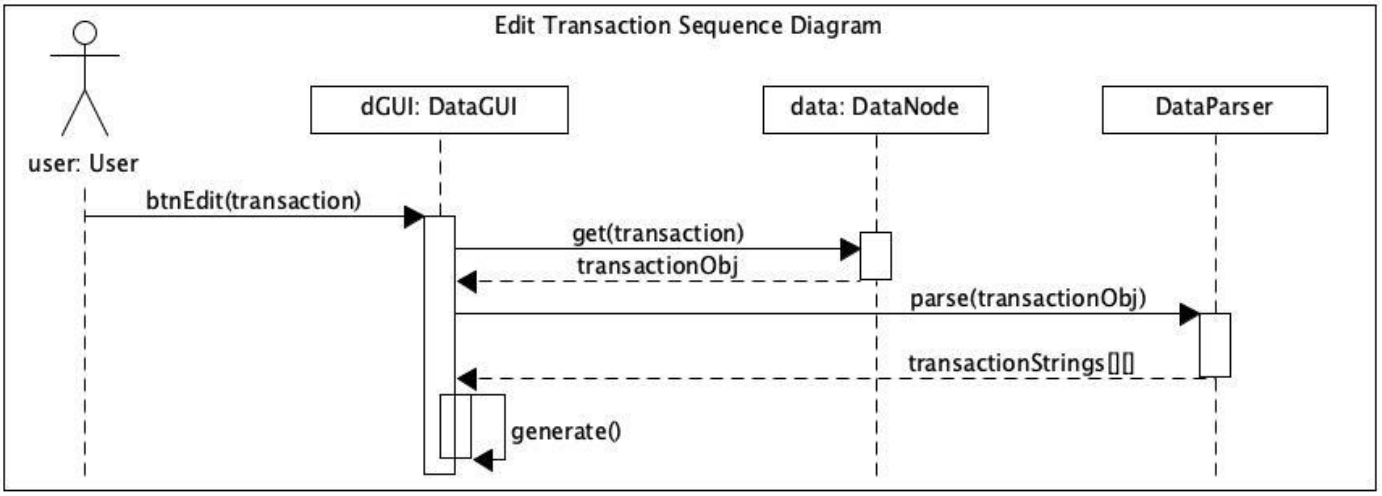

Add Date Range Sequence Diagram
user: User
dGUI: DataGUI
d: DataNode
btnAdd(details)
new(details)
r: Range
add(r)
results

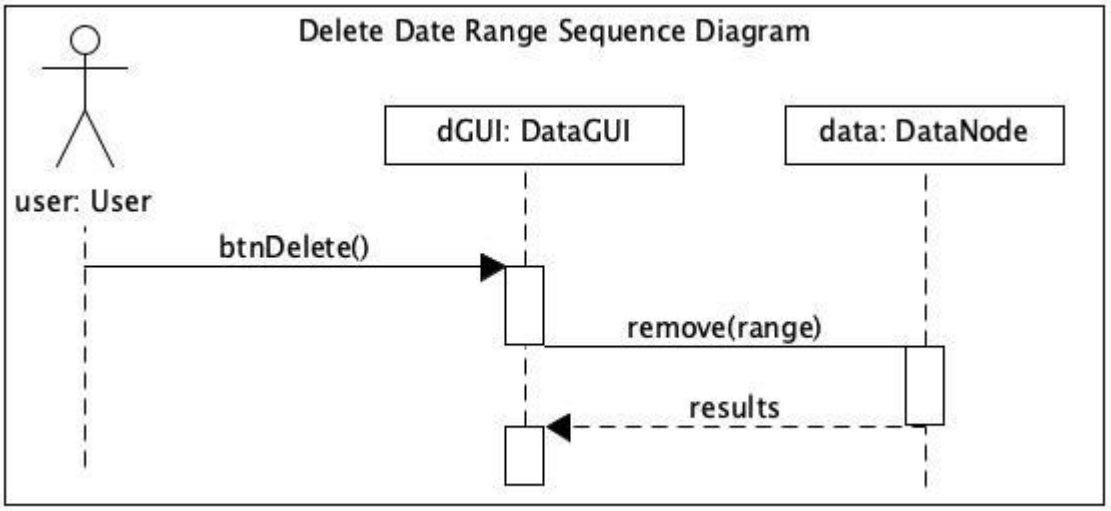

Add Category Sequence Diagram
user: User
dGUI: DataGUI
r: Range
btnAdd(details)
new(details)
c: Category
add(c)
results

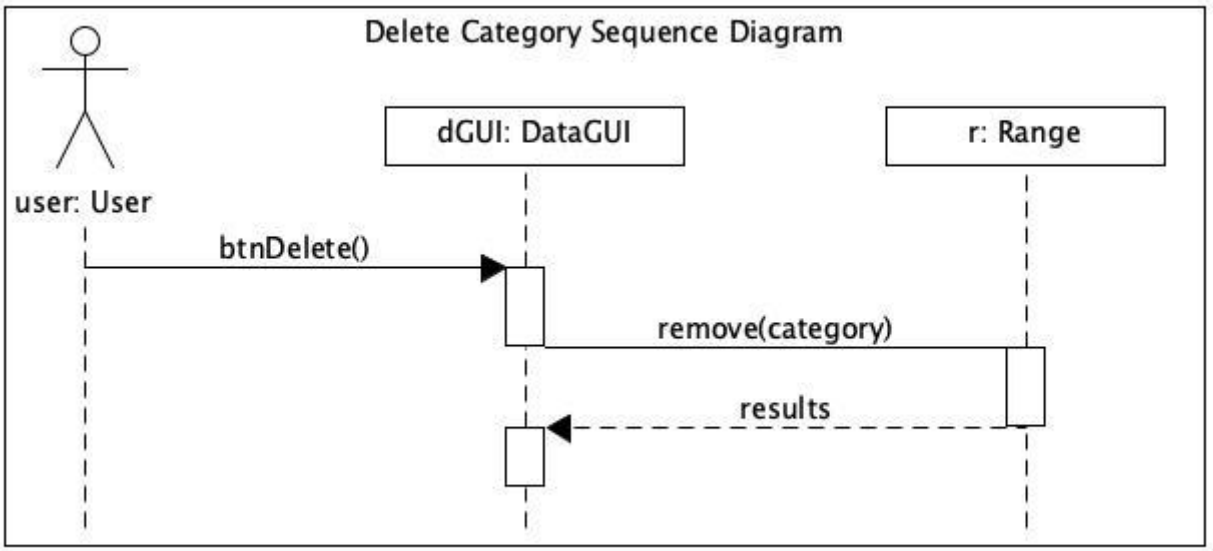

Add Transaction Sequence Diagram
user: User
dGUI: DataGUI
c: Category
btnAdd(details)
new(details)
t: Transaction
add(t)
results

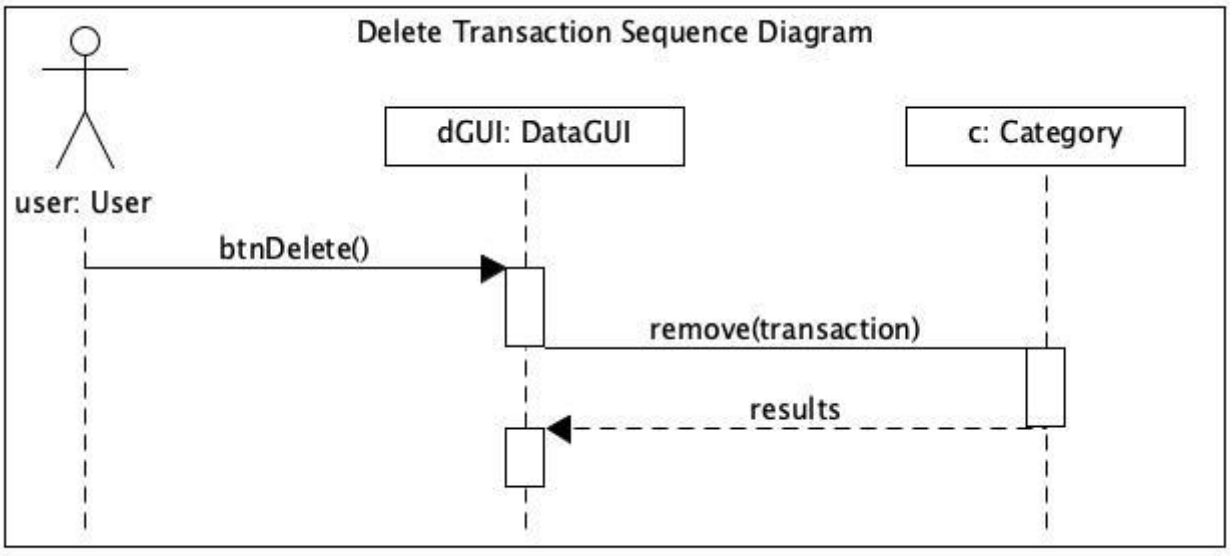

Edit Date Range Sequence Diagram
user: User
dGUI: DataGUI
data: DataNode
DataParser
btnEdit(range)
get(range)
rangeObj
parse(rangeObj)
rangeStrings[][]
generate()


Edit Category Sequence Diagram
user: User
dGUI: DataGUI
data: DataNode
DataParser
btnEdit(category)
get(category)
categoryObj
parse(categoryObj)
categoryStrings[][]
generate()

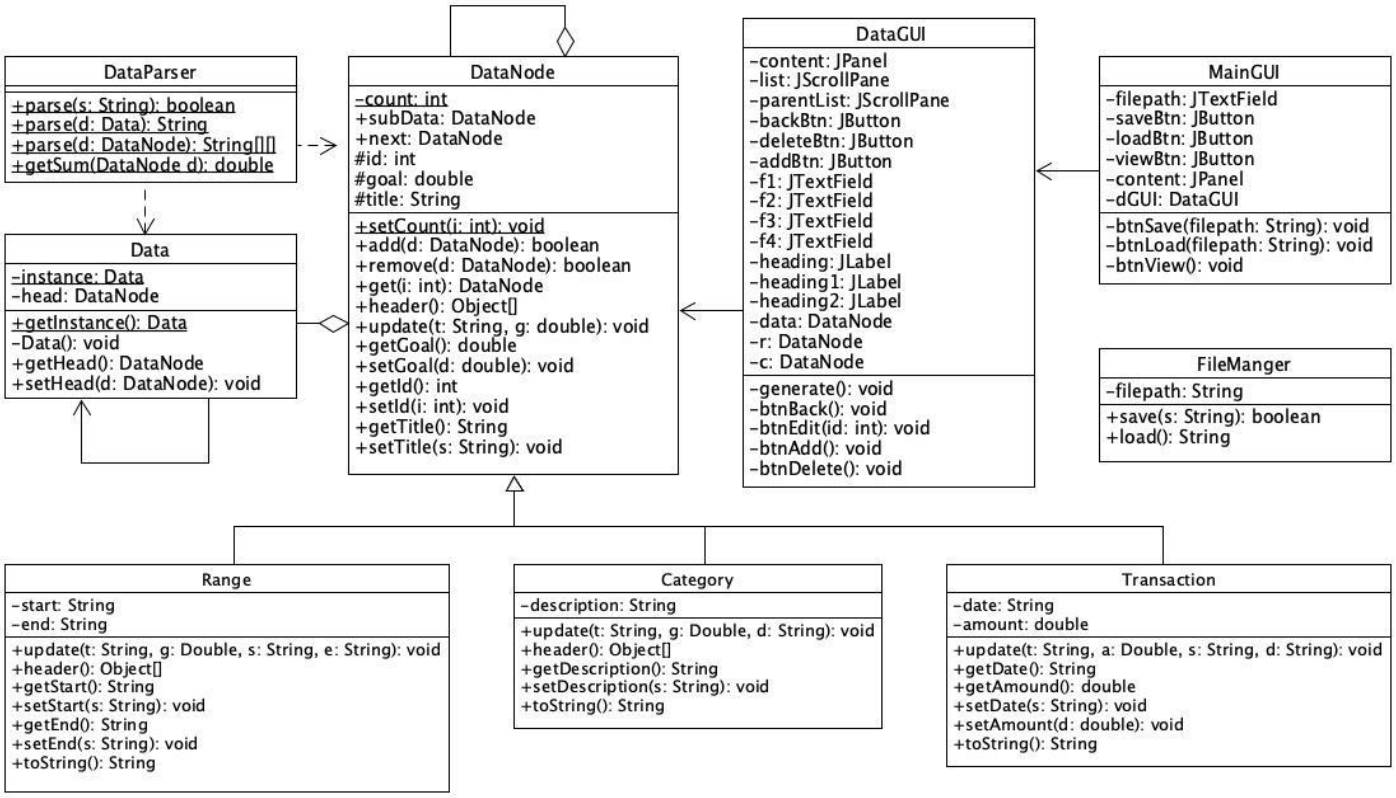

Edit Transaction Sequence Diagram
user: User
dGUI: DataGUI
data: DataNode
DataParser
btnEdit(transaction)
get(transaction)
transactionObj
parse(transactionObj)
transactionStrings[][]
generate()


Delete Date Range Sequence Diagram
user: User
dGUI: DataGUI
data: DataNode
btnDelete()
remove(range)
results


Delete Category Sequence Diagram
user: User
dGUI: DataGUI
r: Range
btnDelete()
remove(category)
results


Delete Transaction Sequence Diagram
user: User
dGUI: DataGUI
c: Category
btnDelete()
remove(transaction)
results

Class Diagram



Conclusion

Overall, this program is a complete execution of the initial proposal, with room for growth in the future. I am personally already using this program for my budgeting, and would like to add more features when I am not so pressed for time. Namely, I'm planning to add the ability to import other CSVs of transactions that are usually able to be generated from credit card and banking statements. The ability to automatically add transactions in this manner would make this program an incredible timesaver. I'm definitely looking forward to getting that up and running in the future. I'd also like to get some rudimentary data validation going, but that's not a pressing concern for my personal use of the program.