1.

Satisfied non-functional requirements:

1. Modulartiy
    a. Code separates concerns with ExpenseTrackerApp, ExpenseTrackerView, and Transaction
2. Usability
    a. In ExpenseTrackerView, the GUI has labeled fields and a table to display transactions

Violated non-functional requirements:

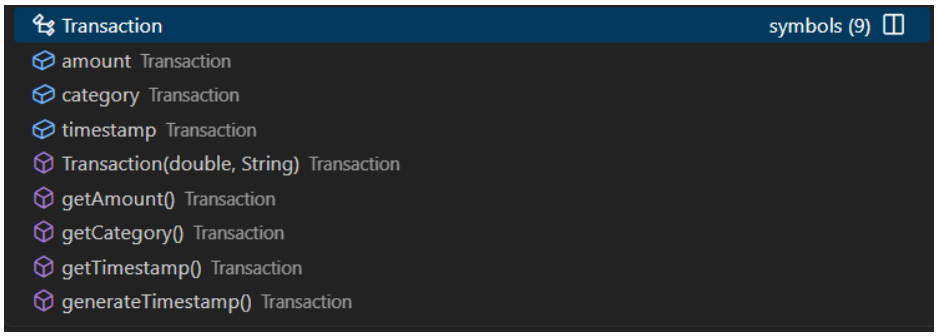
1. Testability
    a. In ExpenseTrackerApp, the logic for creating Transactions and updating the view is embedded in the event handler
    b. This violates testability because the log is mixed directly into the GUI event handler; therefore, it's difficult to test transactions without launching the GUI
    c. To fix this, we should have a controller class that handles the transaction logic, and the view should only forward user input to the controller, allowing for both to be tested separately
2. Extensibility
    a. In ExpenseTrackerView.addTransaction, the table row is updated with fixed columns directly
    b. This violates extensibility because if you want to add a new attribute, then you would have to rewrite parts of the view's code. This creates tight coupling between the transaction structure and UI
    c. To fix this, use TableModel abstraction and maps Transaction objects to rows dynamically; this way, new fields can be added to Transaction without modifying the view directly

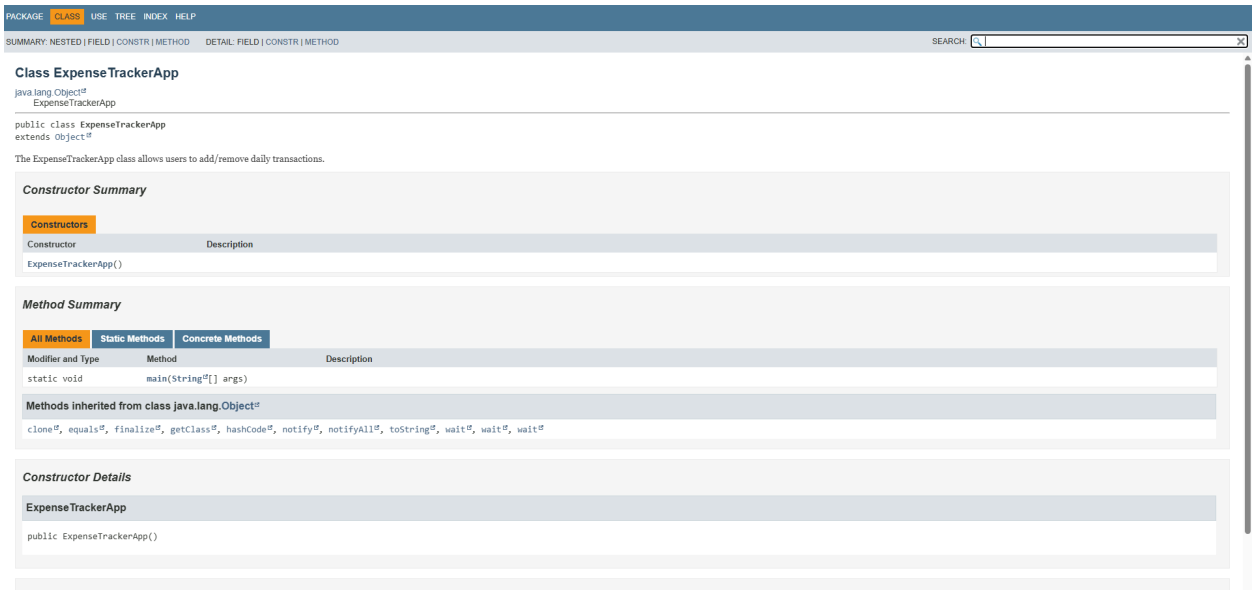
2.1

- Component A is a controller because, at the top, it has input fields for Amount and Category. At the bottom, it has an "Add Transaction" button to submit the entered information. These allow the user to control the application through user input.
- Component B is a view because we can see the transaction table with labeled columns. This shows that it visualizes stored data, but you cannot add or remove data from component B.

3.

Image of the API for Transaction





5.

How to implement the filtering feature in the Expense Tracker

- To implement a filtering feature by category, we could add a drop-down to ExpenseTrackerView and add options to the drop-down. We would also need to introduce a new field and have a getter method to retrieve the user's chosen category to filter. We then must extend the refreshTable() to this so that the table will accurately show the filters
- To implement a filtering feature by amount range, we need to add two fields, min and max value, and their getter methods, which would return the values or null if they are empty. We then must extend the refreshTable() to this so that the table will accurately show the range filters.
- To implement a filtering feature by Date, we need to add a text field for entering a date string, for example, 10-02-2025. We also need to add its getter method, and we need to modify refreshTable() to adjust to these filters.