The code is an implementation of a simple expenditure management system. It allows users to perform various operations on a list of expenses. Let's go through the code and provide a write-up explaining its functionality.

The code begins with the `main` method, which serves as the entry point for the program. It displays a welcome message and then calls the `optionsSelection` method.

The `optionsSelection` method presents a menu of options to the user and handles their selection. It initializes an array `arr` with the available options and an array `arr1` with corresponding option numbers. It then iterates over the options, printing each option from `arr` and waiting for user input. The user's choice is stored in the `options` variable.

Based on the user's choice, a `switch` statement is used to perform different actions. Let's go through each case:

1. Case 1: Review Expenditure

This case prints the list of expenses stored in the `expenses` ArrayList and then calls `optionsSelection` to display the menu again.

2. Case 2: Add Expenditure

This case prompts the user to enter a value for the new expense, which is then added to the `expenses` ArrayList. The updated list is printed, and `optionsSelection` is called.

3. Case 3: Delete Expenditure

This case asks for confirmation and deletes all expenses if the user confirms by selecting the same option. If the confirmation fails, an appropriate message is displayed. After deleting the expenses, `optionsSelection` is called.

4. Case 4: Sort Expenditures

This case calls the `sortExpenses` method, which sorts the expenses in ascending order using the `Collections.sort` method. The sorted list is printed, and `optionsSelection` is called.

5. Case 5: Search Expenditure

This case prompts the user to enter an expense to search for. It then checks if the expense is present in the `expenses` list and displays the appropriate message. After that, `optionsSelection` is called.

6. Case 6: Close the Application

This case calls the `closeApp` method, which simply displays a closing message.

The `closeApp` method displays a closing message to the user, indicating that the application is being closed.

The `searchExpenses` method prompts the user to enter an expense to search for and checks if it exists in the `arrayList` using the `contains` method. It then displays the appropriate message based on the search result.

The `sortExpenses` method sorts the `arrayList` in ascending order using the `Collections.sort` method. The sorted list is then printed.

Overall, this code provides a basic command-line interface for managing expenses. Users can review, add, delete, sort, and search for expenses. It uses ArrayLists to store and manipulate the expense data. However, there are some areas where the code could be improved, such as handling invalid user input or providing a more user-friendly interface.