Algorithm:

1. Start the program.
2. Display a welcome message to the user.
    - Print "Welcome to TheDesk."
3. Call the **optionsSelection** method to present a menu of options to the user.
4. **optionsSelection** Method:
    - Initialize an array **arr** with menu options.
    - Initialize an integer array **arr1** with option numbers.
    - Determine the length of **arr1** and store it in **slen**.
    - Display the menu options with their respective numbers using a for loop.
    - Create an empty ArrayList **expenses** to store expense values.
    - Create a Scanner object **sc** to accept user input.
    - Read the user's choice into the variable **options**.
5. Use a switch-case statement to handle the user's choice:
    - Case 1:
        - Display all saved expenses from the **expenses** ArrayList.
        - Call **optionsSelection** recursively.
    - Case 2:
        - Prompt the user to enter an expense value.
        - Read the user input into the variable **value**.
        - Add **value** to the **expenses** ArrayList.
        - Display a confirmation message.
        - Call **optionsSelection** recursively.
    - Case 3:
        - Display a confirmation message and ask the user to confirm the deletion by entering 3 again.
        - Read the user input into the variable **con_choice**.
        - If **con_choice** is 3, clear all expenses in the **expenses** ArrayList.
        - Display an empty expenses list.
        - Display a message indicating that all expenses are erased.
        - If **con_choice** is not 3, display an error message.

        - Call **optionsSelection** recursively.
    - Case 4:
        - Call the **sortExpenses** method to sort the expenses in ascending order.
        - Call **optionsSelection** recursively.
    - Case 5:
        - Call the **searchExpenses** method to search for a particular expense in the **expenses** ArrayList.
        - Call **optionsSelection** recursively.
    - Case 6:
        - Call the **closeApp** method to close the application.
        - End the program.
    - Default:
        - Display an error message for an invalid choice.
6. **closeApp** Method:
    - Display a closing message.
    - End the program.
7. **searchExpenses** Method:
    - Accept the **arrayList** containing expenses as a parameter.
    - Prompt the user to enter the expense they want to search for.
    - Read the user input into the variable **expenseToSearch**.
    - Initialize a boolean variable **found** to **false**.
    - Loop through the **arrayList** and check if any expense matches **expenseToSearch**.
    - If a match is found, print the index of the expense.
    - If no match is found, print a message indicating that the expense was not found.
8. **sortExpenses** Method:
    - Accept the **arrayList** containing expenses as a parameter.
    - Sort the **arrayList** in ascending order using **Collections.sort**.
    - Display the sorted list of expenses.
9. End the program.

I had pushed the Updated Code to git below is the URL link

https://github.com/R-NandaKumar/Java-Fsd

(press ctrl + click)