“Practice-Project–Fix Bugs of the Application”

1).Bug Fix: Searching Technique

- **Steps Taken:**
  Identified missing code for searching.
  Implemented Binary Search.
  Added appropriate comments.

  Code snippet:

```
private static void searchExpenses(ArrayList<Integer> arrayList)
{
          sortExpenses(arrayList);
          int leng = arrayList.size();
          System.out.println("Enter the expense you need to
search:\t");
          Scanner scanner = new Scanner(System.in);
          int target = scanner.nextInt();
          // Binary Search
          int result = binarySearch(arrayList, target);
          if (result == -1) {
              System.out.println("Expense not found in the
list.");
          } else {
              System.out.println("Expense found at index " +
result);
          }
      }
```

2).Bug Fix: Sorting Predefined Array

- **Steps Taken:**
  Identified sorting bug.
  Implemented Arrays.sort().
  Added appropriate comments.
- **Code Snippet:**

```
private static void sortExpenses(ArrayList<Integer> arrayList) {
          int arrlength =  arrayList.size();
```

```
            //Using Arrays.sort() to sort the expenses in
Ascending order.
            Integer[] arr = arrayList.toArray(new Integer[0]);
            Arrays.sort(arr);
            // Converting the sorted array back to ArrayList
            arrayList.clear();
            arrayList.addAll(Arrays.asList(arr));
            System.out.println("Expenses sorted in ascending
order: " + arrayList);
      }
```

3).Algorithm: Binary Search

- **Explanation:**
- **Input:**

  -‘**arrayList**’: A sorted list of integers.

  -‘**target**’:The value to be searched in the list.

- **Initialize:**

  -‘**Left**’ to 0 (index of the leftmost element in the array).

  -‘**Right**’ to the length of the array minus 1 (index of the rightmost element).

- **Binary Search:**

  -While ‘**left**’ is less than or equal to ‘**right**’:

  - Calculate the middle index: ‘**mid = left + (right - left) / 2.**’
  - If the element at the middle index (‘**arrayList.get(mid)**’) is equal to the ‘**target**’:
    - Return ‘**mid**’ (index where the target is found).
  - If the element at the middle index is less than the ‘**target**’:
    - Update ‘**left**’ to ‘**mid + 1**’ (search in the right half).
  - If the element at the middle index is greater than the target:
    - Update ‘**right**’ to ‘**mid - 1**’ (search in the left half).

- **Output:**

  -If the target is not found, return **-1**.

4).Algorithm: Sorting

- **Explanation:**

  Sorting using Arrays.sort():

  -Input:
  - **'arrayList'**: ArrayList of Integer objects to be sorted.

  -Initialization:
  - Get the size of the **ArrayList: 'arrLength = arrayList.size()'**.

  -Convert to Array:
  - Convert the **ArrayList** to an array of Integer: **'arr = arrayList.toArray(new Integer[0])'.**

  -Sort Array:
  - Use **'Arrays.sort(arr)'** to sort the array of Integer in ascending order.

  -Convert Back to ArrayList:
  - Clear the original **ArrayList**: **'arrayList.clear()'**.
  - Add the sorted elements back to the ArrayList: **''arrayList.addAll(Arrays.asList(arr))''.**

  -Output:
  - The **arrayList** is now sorted in ascending order.

5).GitHub Repository Link

https://github.com/Srushti2680/Java-fsd.git