As a developer, fix the bugs in the application using the appropriate algorithmic techniques.

**package** practice.project;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

/\*System.out.println("Hello World!");\*/

System.***out***.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

System.***out***.println("\tWelcome to TheDesk \n");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

*optionsSelection*();

}

**private** **static** **void** optionsSelection() {

String[] arr = {"1. I wish to review my expenditure",

"2. I wish to add my expenditure",

"3. I wish to delete my expenditure",

"4. I wish to sort the expenditures",

"5. I wish to search for a particular expenditure",

"6. Close the application"

};

**int**[] arr1 = {1,2,3,4,5,6};

**int** slen = arr1.length;

**for**(**int** i=0; i<slen;i++){

System.***out***.println(arr[i]);

// display the all the Strings mentioned in the String array

}

ArrayList<Integer> arrlist = **new** ArrayList<Integer>();

ArrayList<Integer> expenses = **new** ArrayList<Integer>();

expenses.add(1000);

expenses.add(2300);

expenses.add(45000);

expenses.add(32000);

expenses.add(110);

expenses.addAll(arrlist);

System.***out***.println("\nEnter your choice:\t");

Scanner sc = **new** Scanner(System.***in***);

**int** options = sc.nextInt();

**for**(**int** j=1;j<=slen;j++){

**if**(options==j){

**switch** (options){

**case** 1:

System.***out***.println("Your saved expenses are listed below: \n");

System.***out***.println(expenses+"\n");

*optionsSelection*();

**break**;

**case** 2:

System.***out***.println("Enter the value to add your Expense: \n");

**int** value = sc.nextInt();

expenses.add(value);

System.***out***.println("Your value is updated\n");

expenses.addAll(arrlist);

System.***out***.println(expenses+"\n");

*optionsSelection*();

**break**;

**case** 3:

System.***out***.println("You are about the delete all your expenses! \nConfirm again by selecting the same option...\n");

**int** con\_choice = sc.nextInt();

**if**(con\_choice==options){

expenses.clear();

System.***out***.println(expenses+"\n");

System.***out***.println("All your expenses are erased!\n");

} **else** {

System.***out***.println("Oops... try again!");

}

*optionsSelection*();

**break**;

**case** 4:

*sortExpenses*(expenses);

*optionsSelection*();

**break**;

**case** 5:

*searchExpenses*(expenses);

*optionsSelection*();

**break**;

**case** 6:

*closeApp*();

**break**;

**default**:

System.***out***.println("You have made an invalid choice!");

**break**;

}

}

}

}

**private** **static** **void** closeApp() {

System.***out***.println("Closing your application... \nThank you!");

}

//searching using linear search technique

**private** **static** **void** searchExpenses(ArrayList<Integer> arrayList) {

**int** leng = arrayList.size();

System.***out***.println("Enter the expense you need to search:\t");

Scanner sc = **new** Scanner(System.***in***);

**int** expenseToSearch = sc.nextInt();

**boolean** found = **false**;

**for** (**int** i = 0; i < leng; i++) {

**if** (arrayList.get(i) == expenseToSearch) {

found = **true**;

System.***out***.println("Expense found at index " + i);

**break**;

}

}

**if** (!found) {

System.***out***.println("Expense not found.");

}

}

//sorting using bubble sort technique

**private** **static** **void** sortExpenses(ArrayList<Integer> arrayList) {

**int** arrlength = arrayList.size();

**int**[] expensesArray = **new** **int**[arrlength];

**for** (**int** i = 0; i < arrlength; i++) {

expensesArray[i] = arrayList.get(i);

}

// Sort the expenses in ascending order

**for** (**int** i = 0; i < arrlength - 1; i++) {

**for** (**int** j = 0; j < arrlength - i - 1; j++) {

**if** (expensesArray[j] > expensesArray[j + 1]) {

**int** temp = expensesArray[j];

expensesArray[j] = expensesArray[j + 1];

expensesArray[j + 1] = temp;

}

}

}

// Update the sorted expenses back to the ArrayList

**for** (**int** i = 0; i < arrlength; i++) {

arrayList.set(i, expensesArray[i]);

}

System.***out***.println("Expenses sorted in ascending order.");

System.***out***.println(arrayList);

}

}

Algorithm for linear search

1. Get the length of the ArrayList using arrayList.size().

2. Read the expense to search from the user using sc.nextInt().

3. Initialize a boolean variable found as false to keep track of whether the expense is found.

4. Iterate over the elements of the ArrayList using a for loop from index 0 to leng - 1.

• Within each iteration:

• Check if the current element (arrayList.get(i)) is equal to the expense being searched (expenseToSearch).

• If a match is found, set found to true, print the index where the expense is found (i), and exit the loop using break.

5. After the loop, check the value of found.

• If found is false, output "Expense not found."

• If found is true, the code will have already printed the index where the expense is found.

Algorithm for Bubble Sort

1. Get the length of the ArrayList using arrayList.size().

2. Create an array expensesArray with a length equal to arrlength.

3. Copy the elements from the ArrayList to expensesArray using a loop:

• Iterate from 0 to arrlength - 1.

• For each iteration, assign arrayList.get(i) to expensesArray[i].

4. Perform the Bubble Sort to sort the expensesArray in ascending order:

• Iterate from 0 to arrlength - 2 (outer loop i).

• Within the outer loop, iterate from 0 to arrlength - i - 2 (inner loop j).

• Compare expensesArray[j] with expensesArray[j + 1].

• If expensesArray[j] is greater than expensesArray[j + 1], swap them using a temporary variable.

5. After the sorting is complete, update the sorted expenses back to the ArrayList:

• Iterate from 0 to arrlength - 1.

• For each iteration, assign expensesArray[i] back to arrayList using arrayList.set(i, expensesArray[i]).

6. Output the message "Expenses sorted in ascending order."

7. Print the sorted ArrayList.