**package** Project;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Scanner;

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

//moved arrays to class level for access throughout app/class

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

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

**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() {

//added expenses before while loop to prevent them from being

//added again from method call

*expenses*.add(1000);

*expenses*.add(2300);

*expenses*.add(45000);

*expenses*.add(32000);

*expenses*.add(110);

*expenses*.addAll(*arrlist*);

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 array unnecessary and moved scanner object creation outside loop

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

//replaced for loop with while loop and added a key to exit loop

**while**(**true**){

//for loop to display options whenever loop is re-entered

**for**(**int** i=0; i<arr.length;i++){

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

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

}

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

//moved scanner object inside loop to get user's input every time

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

**switch** (options){

**case** 1: //added if statement for when expenses are empty

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

**if**(*expenses*.isEmpty()) {

System.***out***.println("You have no saved expenses\n");

}**else** {

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

}

**break**;

**case** 2: //no changes made to adding values

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");

**break**;

**case** 3: //no changes made to deletion of list

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!");

}

**break**;

**case** 4: //added method call to sort arraylist

System.***out***.println("Sorting expenses form least to greatest.\n");

*sortExpenses*(*expenses*,0,*expenses*.size()-1);

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

**break**;

**case** 5: //added method call to search for value in array list

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

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

*binarySearch*(searchValue);

**break**;

**case** 6: //close scanner object before closing app

sc.close();

*closeApp*();

**break**;

**default**: //left unchanged

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

**break**;

}

}

}

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

// added systen call to exit application

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

System.*exit*(0);

}

**private** **static** **void** binarySearch(**int** value) {

// Binary search through sorted array

// passes in value from main to start search

System.***out***.println("Sorting values first before searching...");

*sortExpenses*(*expenses*,0,*expenses*.size()-1);

System.***out***.println("Sorting complete. Begining search...\n");

**int** check = *searchExpenses*(*expenses*,0,*expenses*.size()-1,value);

//if value is found then return it, otherwise tell user could not find it

**if**(check == -1) {

System.***out***.println("Could not find "+value+" expense.\n");

}

**else** {

System.***out***.println("Found "+value+" expense.\n");

}

}

**private** **static** **int** searchExpenses(ArrayList<Integer> arrayList, **int** start, **int** finish, **int** value) {

//Complete the method

//implement binary search for array list

// do this until start is greater then finish

**if**(finish >= start) {

// determine middle for searching

**int** mid = start + ((finish - start)/2);

// return value if found at middle

**if**(arrayList.get(mid) == value) {

**return** mid;

}

// discard top half and search bottom half if the middle is greater than search value

**if**(arrayList.get(mid) > value) {

**return** *searchExpenses*(arrayList,start,mid-1,value);

}

// discard bottom half and search top half if the middle is less than search value

**return** *searchExpenses*(arrayList,mid+1,finish,value);

}

// return -1 if value is never found in array list

**return** -1;

}

**private** **static** **void** sortExpenses(ArrayList<Integer> arrayList, **int** start, **int** finish) {

//Complete the method. The expenses should be sorted in ascending order.

//implement quicksort algorithm

// continue until start is greater than finish

**if**(start < finish) {

// create an index and pass it to partition

**int** index = *partition*(arrayList,start,finish);

*sortExpenses*(arrayList, start, index-1);

*sortExpenses*(arrayList, index+1, finish);

}

}

**private** **static** **int** partition(ArrayList<Integer> arrayList, **int** start, **int** finish) {

// create a pivot from the value at the end of the array

**int** pivot = arrayList.get(finish);

// start index at beginning of the array

**int** index = start - 1;

// continue until start and finish cross

**for**(**int** i=start; i<finish; i++){

// if values above and below the pivot are found then swap them

**if**(arrayList.get(i) < pivot) {

index++;

**int** temp = arrayList.get(index);

arrayList.set(index,arrayList.get(i));

arrayList.set(i,temp);

}

}

// replace the pivot with the last swapped value

**int** temp = arrayList.get(index+1);

arrayList.set(index+1,arrayList.get(finish));

arrayList.set(finish,temp);

// return the index for the next partition

**return** index+1;

}

}