31.03.2022

FIX BUGS OF THE APP PROJECT SOURCE CODE

Prepared by: PRiYANKA DAS

JAVA CODE :

```
import java.util.*;

public class Fix_The_Bugs {
   /*
       While Integers shouldn't be used for handling money, I
wasn't sure if changing the list to BigDecimal
       was allowed or wanted for this assignment, so I left the
List as a List of Integers.
    */
   private static List<Integer> expenses = new ArrayList<>();
   private static final String ERROR_MESSAGE = "ERROR:
Please enter a valid integer!";

   private final static StringBuilder arr = new StringBuilder()
         .append("1. I wish to review my expenditure\n")
         .append("2. I wish to add my expenditure\n")
         .append("3. I wish to delete my expenditure\n")
         .append("4. I wish to sort the expenditures\n")
         .append("5. I wish to search for a particular
expenditure\n")
         .append("6. Close the application\n");

   public static void main(String[] args) {
System.out.println("\n**********************************
********\n");
       System.out.println("\tWelcome to TheDesk \n");

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

```
  addInitialExpenses();
        System.out.println("Current expenses: " +
expenses);
        optionsSelection();
    }
    private static void addInitialExpenses() {
        expenses.add(1000);
        expenses.add(2300);
        expenses.add(45000);
        expenses.add(32000);
        expenses.add(110);
    }
    private static void optionsSelection() {
        int optionSelected = 1;
        do {
            System.out.print(arr);
            Scanner sc = new Scanner(System.in);
            /*
                Ensures valid user input by catching an
InputMismatchException which continues to the next
iteration
                to allow the user to try again.
             */
            try {
                optionSelected = sc.nextInt();
            } catch (InputMismatchException e) {
                System.out.println("\n" + ERROR_MESSAGE +
"\n");
                continue;
            }
```

```
 switch (optionSelected) {
            case 1:
               System.out.println("Your saved expenses
are listed below: \n");
               System.out.println(expenses + "\n");
               break;
            case 2:
               addExpenditure(sc);
               break;
            case 3:
               deleteExpenses(optionSelected, sc);
               break;
            case 4:
               sortExpenses(expenses);
               break;
            case 5:
               searchExpenses(expenses, sc);
               break;
            case 6:
               closeApp();
               break;
/*
               Allow the user to pick another option if
they make a non-existent choice by accident.
             */
            default:
               System.out.println("\nYou have made an
invalid choice!\nChoose '6' if you wish to exit.\n");
               break;
         }
```

```
 // The loop will not end unless option 6 is selected.
      } while (true);
   }

   // exits the program
   private static void closeApp() {
      System.out.println("Closing your application... \nThank
you!");
      System.exit(0);
   }

   private static void searchExpenses(List<Integer> arrayList,
Scanner sc) {
      System.out.println("Enter the expense you need to
search:\t");

      int key = -1;
      try {
         key = sc.nextInt();
      } catch (InputMismatchException e) {
         System.out.println("\n" + ERROR_MESSAGE);
         return;
      }

      System.out.println(arrayList);
// Searching to see if an element exists in the List can be easily
accomplished through using the indexOf method
 int index = arrayList.indexOf(key);

 if (index < 0)
 System.out.println(key + " is not present in the list!");
 else
 System.out.println("Value " + "[ " + key + " ]" + " has been found
at index: " + index);
}
```

```
  //Sorts in ascending order using the existing functionality
given to us by Collections
    private static void sortExpenses(List<Integer> arrayList) {
        Collections.sort(arrayList);
        System.out.println("\n" + arrayList);
    }

    private static void deleteExpenses(int optionSelected,
Scanner sc) {
        System.out.println("You are about the delete all your
expenses! " +
                "\nConfirm again by selecting the same
option...\n");
        int con_choice = -1;

        try {
            con_choice = sc.nextInt();
        } catch (InputMismatchException e) {
            System.out.println(ERROR_MESSAGE);
            return;
        }
 // Empty the list if the user wants to delete all the expenses
        if (con_choice == optionSelected) {
            expenses.clear();
            System.out.println(expenses + "\n");
            System.out.println("All your expenses are erased!\n");
        } else {
```

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

    private static void addExpenditure(Scanner sc) {
        System.out.println("Enter the value to add your
Expense: \n");
        int value;
        try {
            value = sc.nextInt();
        } catch (InputMismatchException e) {
            System.out.println(ERROR_MESSAGE);
            return;
        }
        System.out.println("Your value is updated\n");
        expenses.add(value);
        System.out.println(expenses + "\n");
    }
}
```

THE END