Overview

Exercise\_13\_10 tests a GasMilageCalculator which takes in strings and calculates a vehicle’s gas milage.

### PROCESSING LOGIC

Testing Logic:

1. Run tryCatchMe()
2. Get user input for miles driven and gallons used
3. Try writing GasMilageCalculator.calculateMilage(string, string)
4. catch FormatException, if caught tell the user the wrong data was entered then run tryCatchMe()(return to step 1)
5. Accept any Key Input for user to exit.

GasMilageCalculator Logic:

1. Take in input of two strings: milesDriven and gallonsUsed
2. Convert milesDriven and gallonsUsed to doubles using Convert, if they are out of format they will be converted to 0.
3. if converted milesDriven times converted gallonsUsed (if one variable is 0 then the product will also be 0) does not equal 0 then return milesDriven divided by gallonsUsed
4. else throw new FormatException;

### DATA (INPUT/OUTPUT)

Input: string: milesDriven, string: gallonsUsed

Output: double: gasMilage

### COMPONENTS (SOURCE CODE NAMES, CLASSES, METHODS)

|  |
| --- |
| **Exercise\_13\_10** |
|  |
| +static Main(args[])  +static tryCatchMe(): void |

|  |
| --- |
| **GasMilageCalculator** |
|  |
| +static calculateMilage(string, string): double |

### TESTING

Scenario 1 – Basic test

Steps to test:

1. Start program
2. Enter “60”
3. Enter: “12”
4. Read: Gas milage = 5
5. Exit program

Expected reaction:

For the program as the steps say.

Actual result:

Expected reaction was actual result. Program works.

Scenario 1 – Exception Test

Steps to test:

1. Start program
2. Enter “60”
3. Enter: “f”
4. Read: “Wrong data was entered.”
5. Enter “up”
6. Enter “9”
7. Read: “Wrong data was entered.”
8. Enter “10”
9. Enter “5”
10. Read: “2”
11. Exit program

Expected reaction:

For the program to properly handle the FormatException and to continue allowing the user to enter new variables.

Actual result:

Expected reaction was actual result. Program works.

##### 