Calculator

Interfaces

ICalculator: exposes the method for the calculator

Calculate(Operator operation, double leftOperand, double rightOperand)

IUserCalculator: exposes the method for the usercalculator

PerformOperation(Operator oper, double leftOperand, double rightOperand)

Classes

Calculator: This class is implemented using singleton pattern to ensure that only one instance of the calculator class exists at a given point of time

UserCalculator: This class contains a private field for ICalculator which is set to the instance of the singleton object. This class contains an overloaded constructor to mock the behavior of ICalculator using dependency injection for unit tests.

Operation: This is the base class for the operations that we perform using the calculator

Add: This class inherits the operation class and overrides the Perform method to perform addition.

Subtract: This class inherits the operation class and overrides the Perform method to perform subtraction.

Multiply: This class inherits the operation class and overrides the Perform method to perform multiplication.

Divide: This class inherits the operation class and overrides the Perform method to perform division

Constants: This class contains the definition of the enumeration Operator which contains the four operations

Program: This class contains the main method which takes the input from the console user.

Implementation of concepts

Inheritance: The classes Add, Subtract, Multiply and Divide inherit from the Operation class and overload the abstract method Perform

Encapsulation: The Operator class contains the leftOperand and rightOperand as its properties which are set only during the instance creation.

Polymorphism: The abstract method Perform() in the Operator class exhibits different behavior depending on the class that overloads the method. However, the actual behavior is decided only at runtime depending on the object calling the Perform() method.

Singleton pattern: The Calculator class is implemented using the singleton pattern. The constructor is private and a static instance of the class can be obtain using the Instance property which internally creates a single instance of the class.

Factory Pattern: This can be observed in the Calculate method of the Calculator class. This method checks the value of the operator and depending upon which operator is present it creates the corresponding object. For example is the operator is ADD then it creates an instance of the Add class.

Unit testing

Scenarios tested:

- Addition
- Subtraction
- Multiplication
- Division
- Division by Zero should throw DivideByZeroException
- Overflow of double in case of Addition should throw ArithmaticException
- Overflow of double in case of Subtraction should throw ArithmaticException
- Overflow of double in case of Multiplication should throw ArithmaticException