Programming Assignment 4: Data Driven

Due Date

- See Piazza for any changes to due date and time
    - Friday ***April 29*** before midnight
    - Grading the next day
- Submit all files and directories to Perforce
    - Create a directory called: PA4 in your student directory
    - /student/<yourname>/PA4/…
        - Need to create and run CleanMe.bat before submission
        - Any additional files that are generated will incur point deduction

Goals

- C# refresher
- Data Driven messaging
- Queuing of dissimilar messages

Assignments

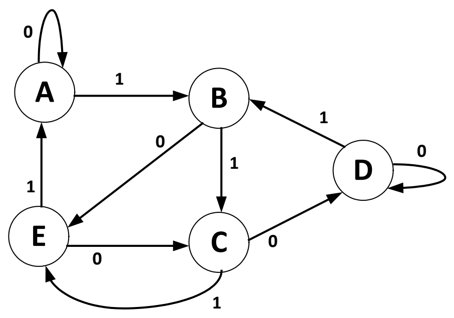
1. **Create a C# solution**
    a. C# console application
    b. Name the project **DataDriven**

2. **Create a simple calculator class**
    a. Simple calculator class
        i. Contains only one data element, ***int*** result
        ii. This member private value
            1. Contains the resulting answer of the calculator operations
    b. Class needs to have the following public methods
        i. Add
            1. Input ***int*** value
            2. Adds the input value to the internal result
            3. Store answer in the result field
        ii. Sub
            1. Input ***int*** value
            2. Subtract the input value from the internal result
            3. Store answer in the result field
        iii. Set
            1. Input ***int*** value
            2. Overwrites the internal result
            3. Store answer in the result field

- iv. GetAnswer
  1. Returns the current result of the calculator ***int***
- v. Mult
  1. Input ***int*** value
  2. Multiplies the input value with the existing result
  3. Store answer in the result field
- vi. Dump
  1. Private method to print the interactions of the operations
- vii. DoWork
  1. Input a user define data type
  2. Executes a specific method described inside the data type
  3. Uses the data in the data type as the data for the function

3. Create a Finite State Machine Class

- a. Simple Finite State Machine (FSM) Class
  - i. Contains only one data element, ***FSM_STATE*** state
  - ii. This member private value
    1. Contains the current state
  - iii. ***FSM_STATE*** is a user define state
    1. Suggestion use enumerations
- b. State machine needs to follow this relationship
  - i. There is only one input 1/0
  - ii. There are 5 states, A,B,C,D,E



- c. Class needs to have the following public methods
  - i. Set
    1. Input the user defined, ***FSM_STATE***
    2. This function, overwrite the internal current state

- ii. Advance
    1. Input type ***Byte***, the input stimulus for the state machine
    2. Base on the FSM diagram, this function advances to the next appropriate state
    3. Internally storing the result in the current state variable
- iii. GetState
    1. Returns the current state variable, ***FSM_STATE***
- iv. Dump
    1. Private method to print the interactions of the operations
- v. DoWork
    1. Input a user define data type
    2. Executes a specific method described inside the data type
    3. Uses the data in the data type as the data for the function

4. Calculator: Tests

- a. Phase 1 – Explicit Calls
    - i. Write / modified the supplied tests to mimic the output from Calc_Tests()
        1. The ordering of the functions and the input data must match!
    - ii. In the test function
        1. Call the methods explicitly
    - iii. Output should be similar to the supplied output.
- b. Phase 2 – Data Driven
    - i. Write / modified the supplied tests to mimic the output from Calc_Tests()
        1. The ordering of the functions and the input data must match!
    - ii. In the test function
        1. Call the methods using the DoWork() with data
    - iii. Output should be similar to the supplied output.
- c. Phase 3 – Queued Data Driven
    - i. Write / modified the supplied tests to mimic the output from Calc_Tests()
        1. The ordering of the functions and the input data must match!
    - ii. In the test function
        1. The data of each transactions is stored in a queue of your choice
        2. All the transactions are added to the queue first
        3. Then the queue is processed, call the DoWork() to process all the queued operations
    - iii. Output should be similar to the supplied output.

5. **Finite State Machine: Tests**
    a. Phase 1 – Explicit Calls
        i. Write / modified the supplied tests to mimic the output from FSM_Tests()
            1. The ordering of the functions and the input data must match!
        ii. In the test function
            1. Call the methods explicitly
        iii. Output should be similar to the supplied output.
    b. Phase 2 – Data Driven
        i. Write / modified the supplied tests to mimic the output from FSM_Tests()
            1. The ordering of the functions and the input data must match!
        ii. In the test function
            1. Call the methods using the DoWork() with data
        iii. Output should be similar to the supplied output.
    c. Phase 3 – Queued Data Driven
        i. Write / modified the supplied tests to mimic the output from FSM_Tests()
            1. The ordering of the functions and the input data must match!
        ii. In the test function
            1. The data of each transactions is stored in a queue of your choice
            2. All the transactions are added to the queue first
            3. Then the queue is processed, call the DoWork() to process all the queued operations
        iii. Output should be similar to the supplied output.

6. **Interleaved Tests**
    a. Queued Interleaved Data Driven
        i. Write / modified the supplied tests to mimic the output from InterLeaved()
            1. The ordering of the functions and the input data must match!
        ii. In the test function
            1. The data of each transactions is stored in a queue of your choice
            2. All the transactions are added to the queue first
            3. Then the queue is processed, calling the DoWork() to process all the queued operations
            4. It's up to you how to store and process the dissimilar class operations
        iii. Output should be similar to the supplied output.

7. **Store the result**
    a. Create an output similar to the supplied file
    b. Store the file
        i. <your name>.txt is suffice

8. **Submit your whole project to perforce directory**
    a. Solution and files
    b. Do not include self generated files
        i. you will lose points if you submit extra (unnecessary) files
    c. Include the output.txt file of your results
    d. Include a *cleanMe.bat* to delete all temp files/dir

Validation

Simple check list to make sure that everything is checked in correctly

- Did you create the solution?
    - Added Calculator, FSM, Test classes
    - Additional classes to make it all work
- Does the output for Calculator have all 3 phases: Explicit, Data Driven, Queue?
- Does the output for FSM have all 3 phases: Explicit, Data Driven, Queue?
- Does the output for Interleaved work and match the sample output?
- Did you create a text file with your output?
- Is it at level 4 warnings and clean?

Hints

Most assignments will have hints in a section like this.

- Baby steps, use an very incremental process
    - Big steps will prevent you from finishing task
- Rewatch the lecture demo on this assignment
- Learn C#

Troubleshooting

- Pretty easy assignment if you do it in the order defined