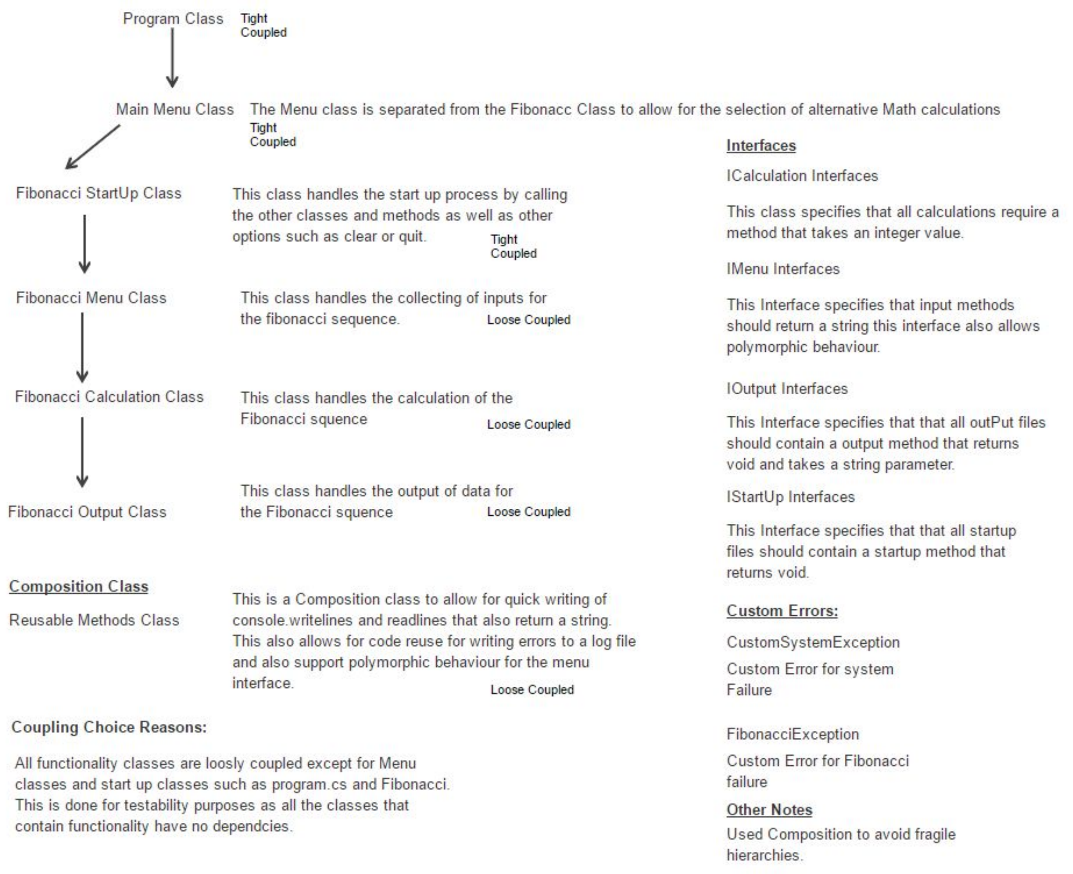
Fibonacci sequence Question.



The Menu class is separated from the Fibonacc Class to allow for the selection of alternative Math calculations

This class handles the start up process by calling the other classes and methods as well as other options such as clear or quit.

This class handles the collecting of inputs for the fibonacci sequence.

This class handles the calculation of the Fibonacci squence

This class handles the output of data for the Fibonacci squence

Composition Class

Reusable Methods Class

This is a Composition class to allow for quick writing of console.writelines and readlines that also return a string. This also allows for code reuse for writing errors to a log file and also support polymorphic behaviour for the menu interface.

Coupling Choice Reasons:

All functionality classes are loosly coupled except for Menu classes and start up classes such as program.cs and Fibonacci. This is done for testability purposes as all the classes that contain functionality have no dependcies.

Interfaces

ICalculation Interfaces

This class specifies that all calculations require a method that takes an integer value.

IMenu Interfaces

This Interface specifies that input methods should return a string this interface also allows polymorphic behaviour.

IOutput Interfaces

This Interface specifies that that all outPut files should contain a output method that returns void and takes a string parameter.

IStartUp Interfaces

This Interface specifies that that all startup files should contain a startup method that returns void.

Custom Errors:

CustomSystemException

Custom Error for system Failure

FibonacciException

Custom Error for Fibonacci failure

Other Notes

Used Composition to avoid fragile hierarchies.

(Overview Diagram available in Overview Files Folder in application)

Encapsulation

The Application is broken down into multiple classes, this is done to make every class responsible for just one task. I have also set all the fields to private as to hide the implementation details.

Composition

I choose composition for my code reuse as it was more appropriate than inheritance and doesn't cause any fragile hierarchies. There is a file called ReusableMethods methods that I use for my methods that I wish to reuse. I used this for:

- Error logging by writing to a text file
- To avoid rewriting a method used for polymorphic behaviour
- Reduce the amount of code written now and in the future as there is a reusable method for menu options that a user enters.

Interfaces
I used interfaces to specify some of the methods each class should have and to allow polymorphic behaviour. In this application I use this polymorphic behaviour for the menus. Currently there are only two menus but the system is built so that it can handle more being added in the future. I also used interfaces to ensure that all the classes are independent of each other to allow for easy unit tests (I created a quick unit test that completes a fake Fibonacci calculation Output.

Abstract class
I thought about using an abstract class for calculations but as of now there didn't seem to be any reason this would be needed as interfaces are being used for now.

Error Handling
There is a try catch block over the main method. When the catch block is hit a message is outputted to the user mentioning that they should contact support and a custom exception that specifies a system failure is displayed. This exception and a timestamp are then saved in a log file.

There is a try catch block over the FibonacciStartUp class StartUp method that has a custom exception that writes to a log file.

Defensive programming
All the if statements have else statements and inputs that are type in by the user are changed To Lower to avoid any issues with capitals.

What I would do in the future
Write more:

- unit tests, Custom errors
- add another calculation type
- make the system more loosely coupled where possible
- create more reusable code blocks.

.