Mohammad Reza Baghban Haghighi

378474

Abstract

design decisions and their advatanges and disadvantages for the project

LPP   
 Design Documentation

FHICT English Stream

# Introduction

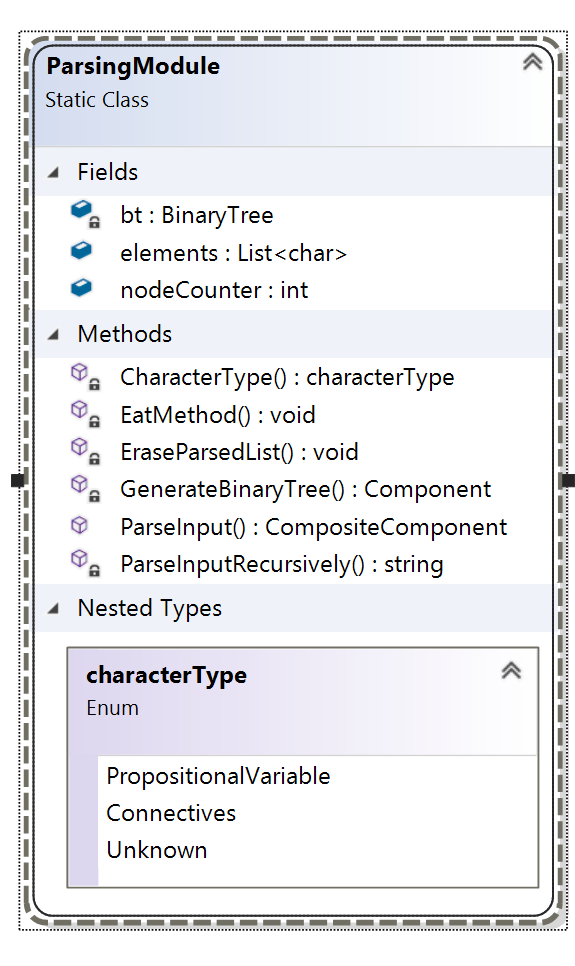
In this document I will try to elaborate my design decisions that I have taken in the process of designing, implementing and testing the LPP application.

# Assignment 1: Parse + Tree

1. **Parsing Module**

For designing parsing module, initially I was thinking either I need to use common class, singleton class or a simple static class with some public static method.

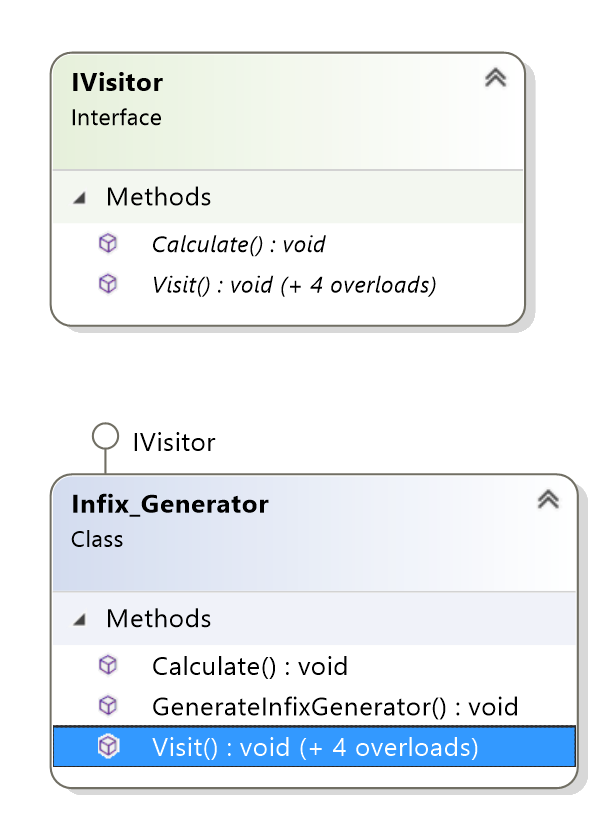
First of all, I did not use common class because I did not find any point of having multiple objects of Parsing Module especially since none of the potential objects does do have their own set of properties which should be distinguished from other objects of that class.

And between choosing singleton pattern and static common class, since singleton pattern usually comes with the concept of immutability and the fact that one universal object needs to be used within different classes. As we do not have none of these situations there is no need application for singleton pattern.

Lastly since one or two methods of parsing module is needed to being exposed to other classes, the last paradigm of having a static class is being choose.

Also for the sake of encapsulation and separation of concerns, all the methods of Parsing Module become private except **ParseInput**() method as the only gate for interacting with module where it will do following Operations:

1. Parse the input from prefix notation to extract all members by calling ParseInputRecursively() Method
2. Generate a binary tree out of given formula by interacting with BinaryTree object bt
3. Return the root of binary tree to the caller
4. **Using Visitor Design Pattern for functionalities | Infix Generator**

Based on the requirements of project where different functionalities needed to be added to software gradually which most of them perform some operations upon the binary tree of propositions it is good to use Visitor Design Pattern.

In this Design pattern new functionalities can be added without changing the implementation of pre-existing objects structure that were built using Composite Pattern and by following this approach the maintainability and extensibility of software design next to its flexibility would be increased considerably in loosely coupled architecture.

By doing this, instead to adding new methods to objects and made alternation to classes in object structure, Objects would be passed to individual Visitor classes that implemented IVistor to perform desired operations.

In Infix\_Generator is a concrete Visitor class who implement IVisitor Interface which would be used for generating infix formula of abstract proposition.

# Assignment 2: Truth table + Hash code

<PROVIDE YOUR ANSWERS HERE>

General notes:

* Describe your approach, detail upon challenges, what were your struggles, how you came across them.
* If you did not manage to implement, also mention all your attempts and why your solution could not be implemented.
* Describe (in existent) additional feature or smart computations that I might miss while assessing your code.
* All team work (2+ students) must be mentioned; e.i.: In class **Tree.cs** from **line 15-55** is code developed together with partner **X (PCN:xxxxxx)**.
* IMPORTNAT: Please be sure to mention all lines of code you worked together with another fellow student, otherwise plagiarism might be detected in your code and you will be sent to the examination board.

# Assignment 3: Simplify

<PROVIDE YOUR ANSWERS HERE>

General notes:

* Describe your approach, detail upon challenges, what were your struggles, how you came across them.
* If you did not manage to implement, also mention all your attempts and why your solution could not be implemented.
* Describe (in existent) additional feature or smart computations that I might miss while assessing your code.
* All team work (2+ students) must be mentioned; e.i.: In class **Tree.cs** from **line 15-55** is code developed together with partner **X (PCN:xxxxxx)**.
* IMPORTNAT: Please be sure to mention all lines of code you worked together with another fellow student, otherwise plagiarism might be detected in your code and you will be sent to the examination board.

# Assignment 4: Normalize

<PROVIDE YOUR ANSWERS HERE>

General notes:

* Describe your approach, detail upon challenges, what were your struggles, how you came across them.
* If you did not manage to implement, also mention all your attempts and why your solution could not be implemented.
* Describe (in existent) additional feature or smart computations that I might miss while assessing your code.
* All team work (2+ students) must be mentioned; e.i.: In class **Tree.cs** from **line 15-55** is code developed together with partner **X (PCN:xxxxxx)**.
* IMPORTNAT: Please be sure to mention all lines of code you worked together with another fellow student, otherwise plagiarism might be detected in your code and you will be sent to the examination board.

# Assignment 5: Nandify

<PROVIDE YOUR ANSWERS HERE>

General notes:

* Describe your approach, detail upon challenges, what were your struggles, how you came across them.
* If you did not manage to implement, also mention all your attempts and why your solution could not be implemented.
* Describe (in existent) additional feature or smart computations that I might miss while assessing your code.
* All team work (2+ students) must be mentioned; e.i.: In class **Tree.cs** from **line 15-55** is code developed together with partner **X (PCN:xxxxxx)**.
* IMPORTNAT: Please be sure to mention all lines of code you worked together with another fellow student, otherwise plagiarism might be detected in your code and you will be sent to the examination board.

# Software design

<PROVIDE YOUR ANSWERS HERE>

General notes:

* Give a general overview of your software design, your classes, interfaces, design patterns
* Argue why your software design is representative to the assignment
* Describe why the choice of classes\interfaces\design patterns or mention why you decide not to choose certain classes\interfaces\design patterns and kept the code simple.
* Describe (in existent) additional feature or smart computations that I might miss while assessing your code.

# GUI

<PROVIDE YOUR ANSWERS HERE>

General notes:

* Give a general overview of your GUI (what does each components do).
* Argue why your GUI is representative to the assignment (easy to use, straightforward or require experienced user).
* Describe why the choice of components or mention why you decide not to choose certain components and kept the GUI simple.
* Describe (in existent) additional feature or smart computations that I might miss while assessing your code.

# Testing

<PROVIDE YOUR ANSWERS HERE>

General notes:

* Give a general overview of your testing (what does each test).
* Argue why your testing is representative to the project (enough - more than 10 tests per assignment - for each components)
* Describe the choice of testing or mention why you decide not to test certain components.
* Describe (in existent) additional feature or smart computations that I might miss while assessing your code.

# Conclusions and future implementations

<PROVIDE YOUR ANSWERS HERE>

General notes:

* Conclude the report and mention future implementations, what could be improved.