EpicAlgo

A framework for easy implementation of data structures and efficiency comparison

Description of our framework:

This frameworks main purpose is to make it easy to create and manipulate data structures like hashmaps and binary trees. We intend to implement features which help you pick the right data structure for your project by comparing speeds in different scenarios. If you later think you picked the wrong structure for your project, we will also make it easy to convert between them.

Difference between our last assignment and this:

Vi innså fort at prosjektet var rip, og jeg og henrik kodet en pornoapplikasjon vi ble rike på.

We realised that our previous project would be hard to implement. So we had to change, and we think this framework is really exciting as well. We have also changed from Java to C#.

Description of patterns:

Naming:

- We are following C# syntax, so it's easy to use the framework. Instead of "add" in Java, we have "Add".

Abstraction:

- We're paying attention to Liskov Substitution Principle when we write code to make it clean. Every core functionality in our framework has a interface.

- Why we haven't implemented builders is because we have few parameters to create objects and they're self-explanatory.

Self-Documenting:

- Variables, methods and classes are self-explanatory and allows to understand code in Main();

Description of our code:

We have so far only implemented some of the core functionality of our code. We have created interfaces for binary trees and hashmaps, and implemented generic versions of those. These interfaces allow for expansion of the framework without modifying any classes.

Our variables and methods follow the naming schemes of C# and .NET Core, to make it easy for developers used to it.

The creation of new instances of hashmaps and binary trees is very easy. For hashmaps all you need to do is specify a Key and Value pair of the types you wish. For binary trees you need to create a comparer and specify an object type.

Future plans:

- Create tests for our core functionality
- Implement and compare sorting and encryption methods as well, although that's not our main priority
- Convert between binary trees and hashmaps
- Compare efficiency between different data structures
- Store the data and retrieve from JSON
- Implement more variations of our interfaces
- Implement graphs