APPLICATION OF TREE DATA STRUCTURES FOR SYSTEMS MODELING

Vasilyeva Marina Alekseevna, Filipchenko Konstantin Mikhailovich

Russian University of Transport (MIIT), Moscow, Russia

Contribution on the State of the Art

https://doi.org/10.7251/JIT2202152A UDC: 629.7.014.9:623.746.2-519

Abstract: The authors developed a binary tree search library. The article presents UML diagrams of the developed classes. The authors supposed Abstract factory design pattern for the opportunity of using search trees' node classes inheritance. The unit tests one developed. This library can be used in the creating the scheduling technical maintenance calculation automated system and the metro train energy optimal trajectory calculation system.

Keywords: Tree structure, interval tree, transport system modeling.

The presence of adequate models allows testing hypotheses in the management of such complex transport facilities as the subway. The development of an automated system is associated with both the verification of certain algorithms, the choice of architectural solutions for a software product, and the choice of a data structure that is used in the implementation of the selected algorithm. To store objects in RAM, standard data structures such as arrays, lists, stack or queue are usually used [1]. To implement specific tasks, the authors, each in the development of their specific automated system, faced the need to use the data structure of a binary search tree [2] or its variation - a segment tree [3].

The binary search tree allows searching and inserting elements with algorithmic complexity $O(log_2(n))$ if the tree is balanced, otherwise, the tree may degenerate into a list, and the algorithmic complexity of the above operations will increase to $O(n)$.

Well-known self-balancing binary trees are the AVL tree and the red-black tree. The AVL tree is named after its creators, the Soviet mathematicians G. M. Adelson-Velsky and E. M. Landis. Balancing the AVL tree is achieved by single and double left and right turns. In a red-black tree, node color and single rotations are involved in balancing.

A segment tree (or interval tree) is a data structure that uses a balanced binary tree and stores intervals as values [3].

Based on the foregoing, UML diagrams [4] of interfaces and classes of the library were developed.

The hierarchy of library interfaces is shown below (Figure 1).

To implement the binary tree library, the authors developed a hierarchy of node base classes (Figure 2) and node implementation classes (Figure 3).

The hierarchy of base classes of the tree library and their specific implementations are presented below (Figure 4, Figure 5).

The hierarchy of base classes of the segment tree library and their specific implementation are presented below (Figure 6, Figure 7).

Each tree class aggregates the class of its particular node. When implementing node classes, the authors used the architectural solution Abstract Factory [5]. The hierarchy of base classes and their specific implementation are presented below (Figure 8, Figure 9).

The developed solution (solution) contains four projects: Demo for demonstrating the work of the library, Stage Library - a specific use of the library of interval trees for working with speed limits on the subway section, Trees - a library of binary trees and Trees. Tests - a library of tests (Figure 10).

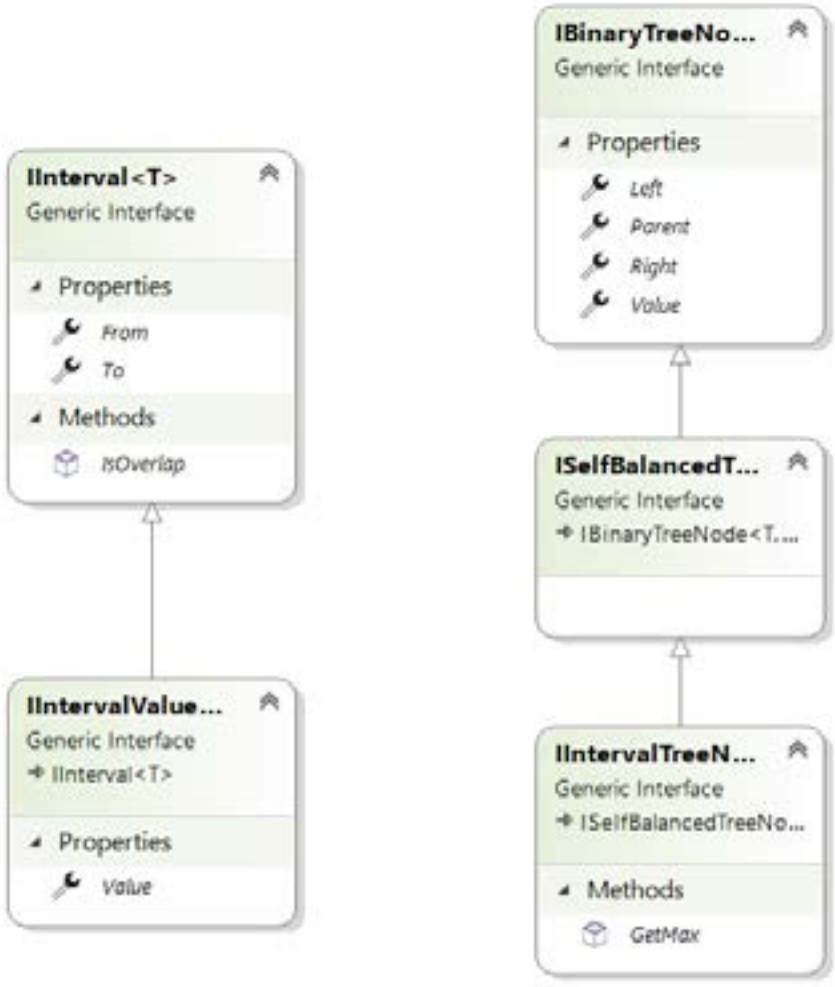

Figure 1 *– Hierarchy of library interfaces*

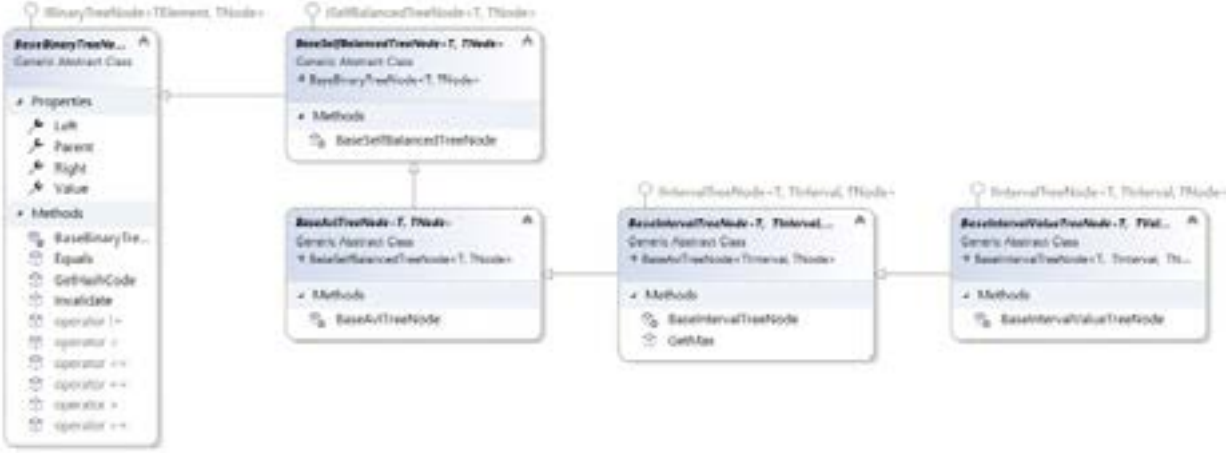

Figure 2 *– The base class hierarchy of the binary search tree node*

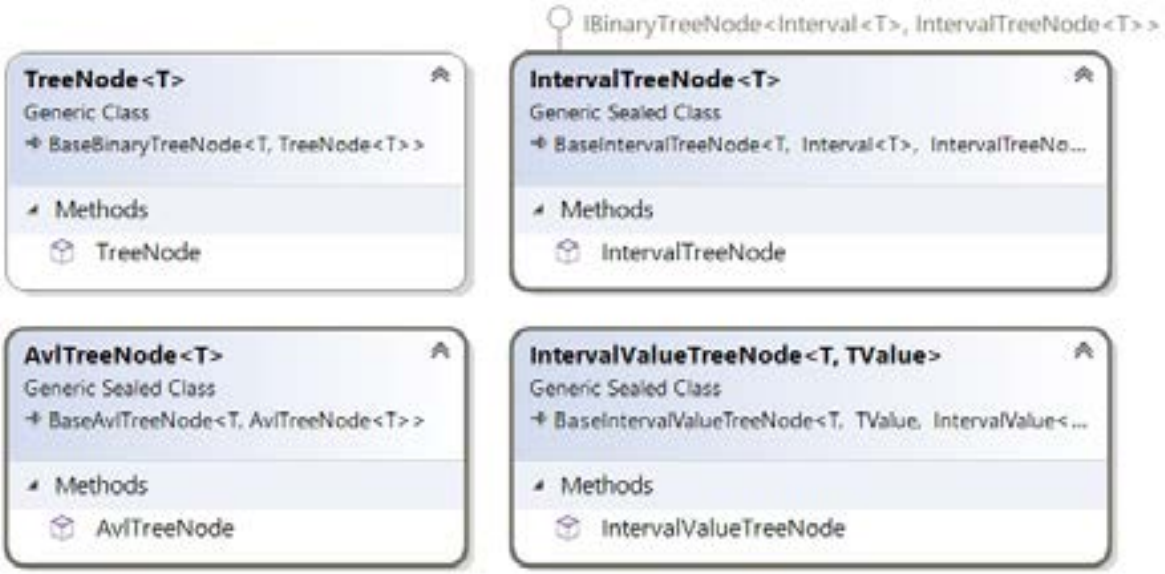

Figure 3 *– Binary search tree node implementation classes*

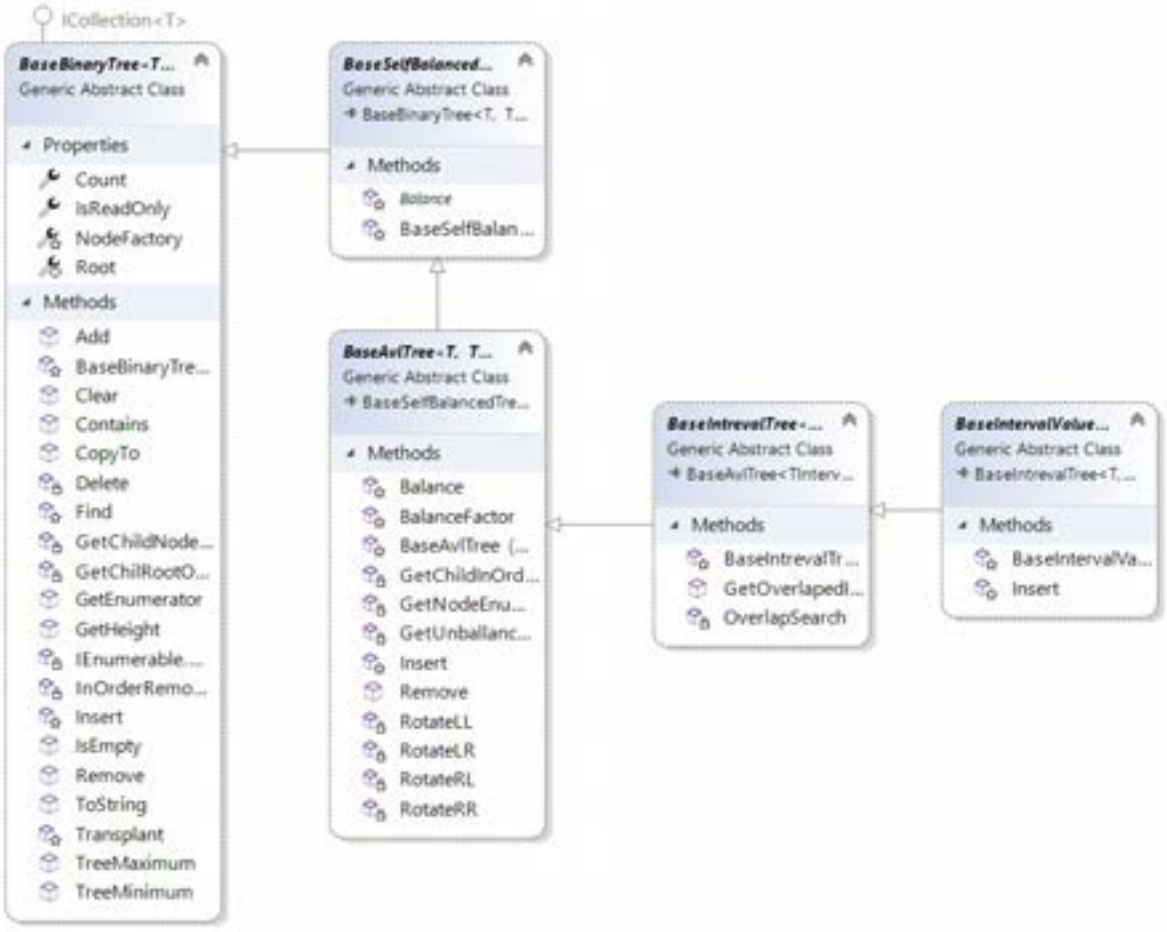

Figure 4 *– Hierarchy of the base classes of the binary search tree library*

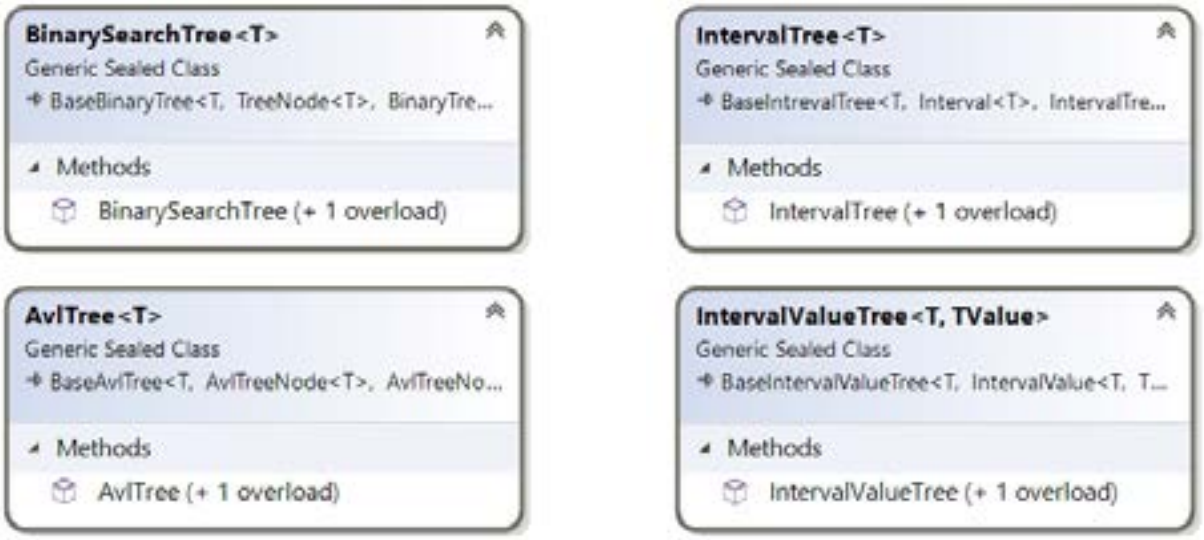

Figure 5 *– Binary search tree implementation classes*

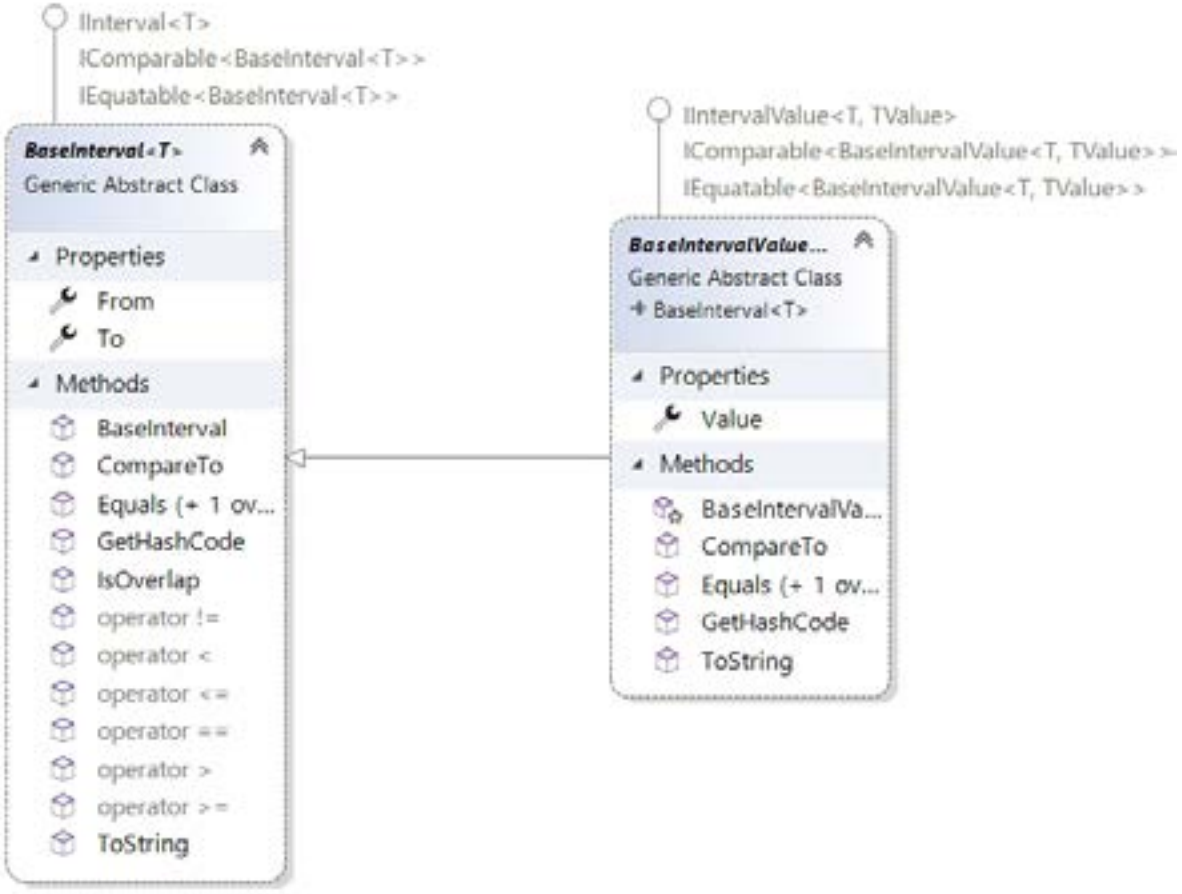

Figure 6 *– Hierarchy of base classes of interval trees*

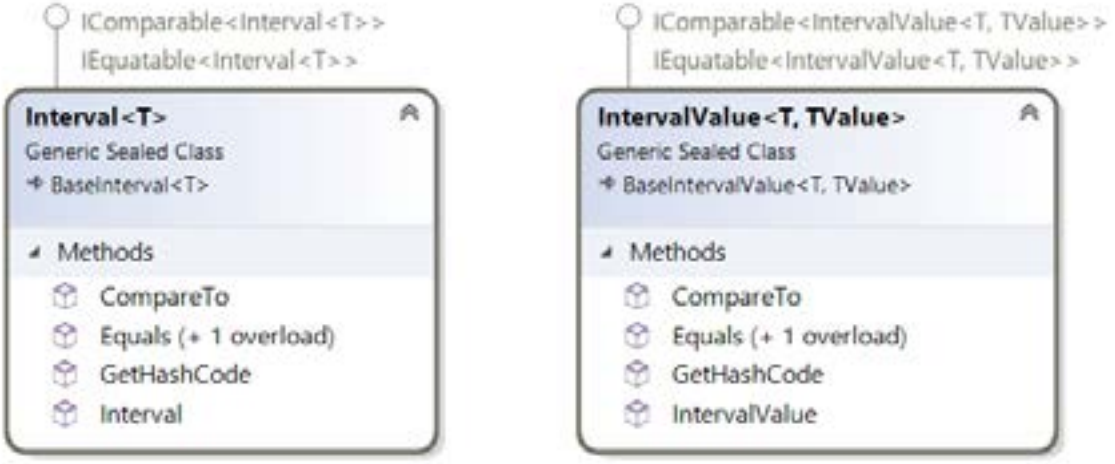

Figure 7 *– Class library of interval trees*

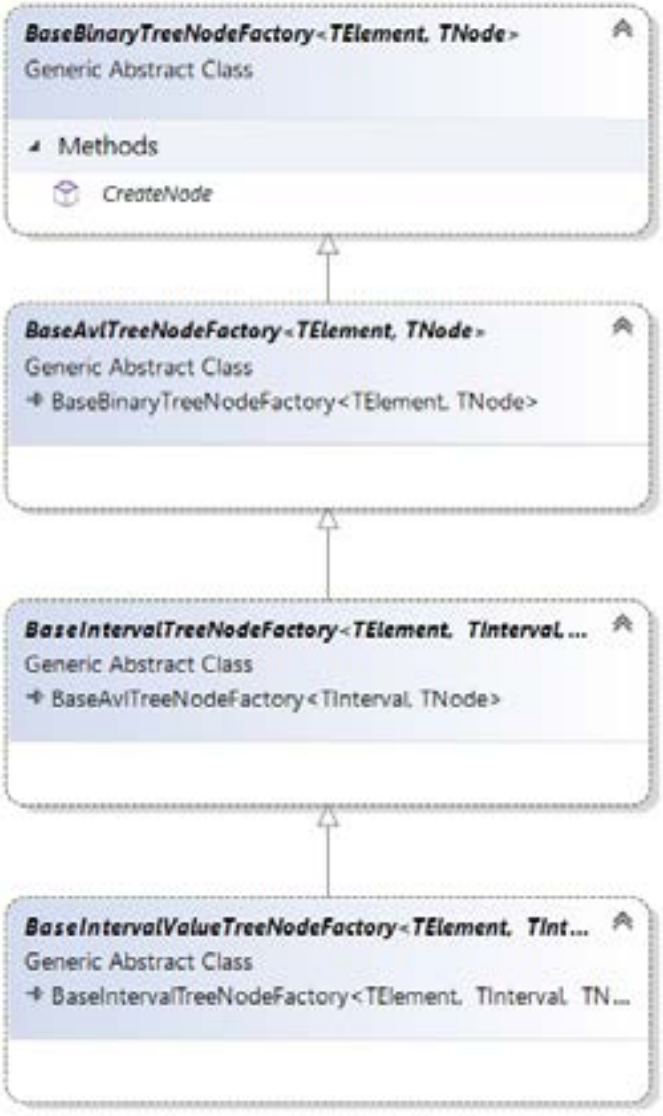

Figure 8 *– Hierarchy of base classes of abstract node factories of binary search trees*

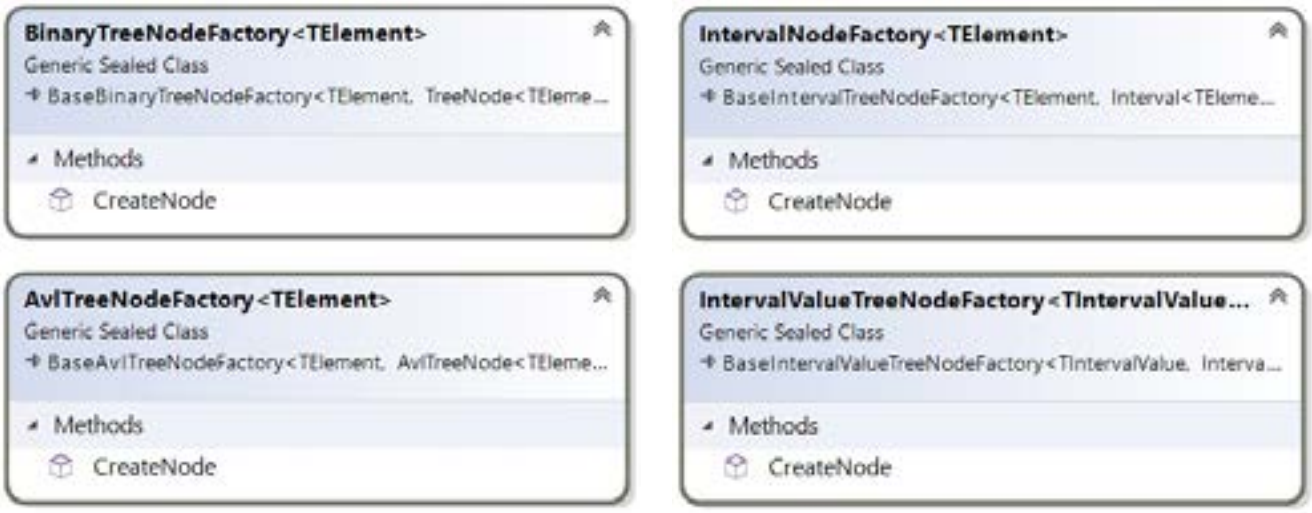

***Figure 9 – Hierarchy of** concrete implementations of binary search tree node factory classes*

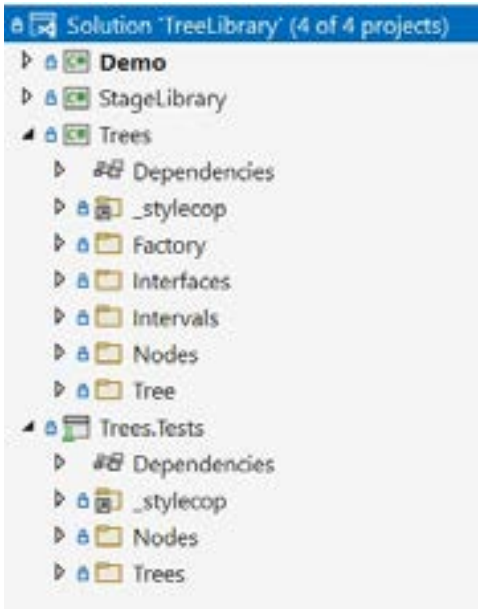

***Figure 10** – Developed solution with projects*

Automated tests were written for the developed classes [6]. All public methods are covered by tests with various outcomes (working with primitive data types or objects, whether the method throws an exception with incorrect input parameters, etc.).

A screenshot of the Test Inspector is shown below (Figure 11). As can be seen from the figure, 87 tests were developed, and all of them passed.

As a result of the work done, the authors created a library of binary search trees and their variety - interval trees. This library is being introduced into automated systems ASER (automated system for selecting energy-optimal operating modes) and AS GTO (automated system "Maintenance Schedule").

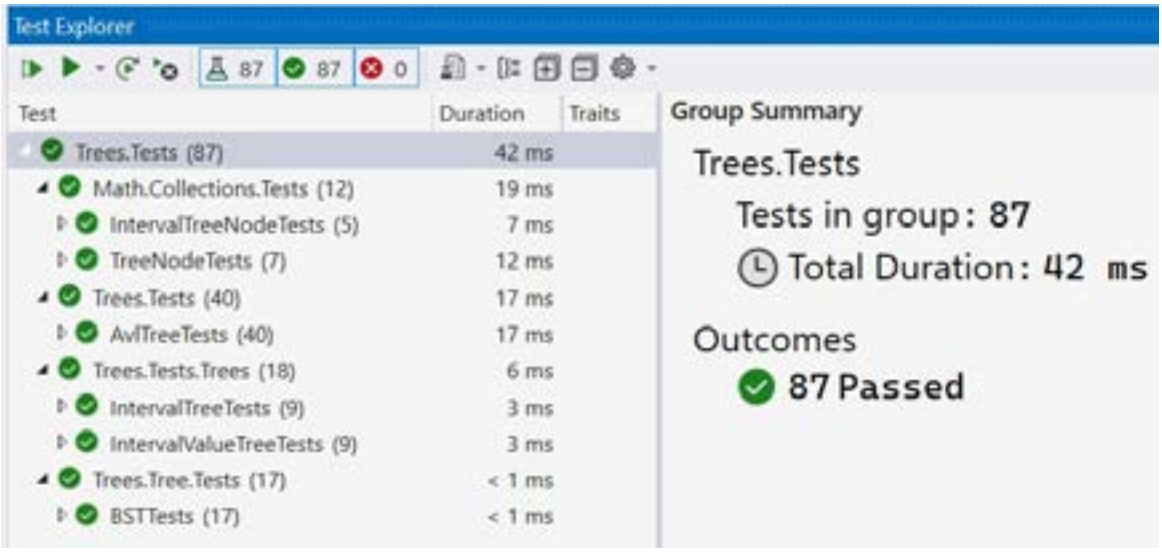

***Figure 11** – Test Inspector*

INFORMATION ABOUT AUTHORS

Vasilyeva Marina Alekseevna - Associate Professor of the Department of Information Management and Security, Russian University of Transport (MIIT).
academic degree: candidate of technical sciences
academic title: associate professor
e-mail: marina_paley@mail.ru

Filipchenko Konstantin Mikhailovich - developer of business applications at Commonwealth Partnership LLC

Применение древовидных структур данных для моделирования систем

Васильева Марина Алексеевна, Филипченко Константин Михайлович

Российский университет транспорта (МИИТ), Москва, marina_paley@mail.ru

Оригинальная научная статья

Аннотация: Авторами разработана библиотека бинарных деревьев поиска. В статье представлены UML-диаграммы разработанных классов. Предложено архитектурное решение Абстрактная фабрика для возможности использования наследования классов узлов бинарных деревьев. Разработаны классы тестов на библиотеку. Применение разработанной библиотеки предложено при создании автоматизированных систем построения графиков технического обслуживания и выбора энергооптимальных траекторий движения поездов метрополитена.

Ключевые слова: Древовидные структуры, дерево отрезков, моделирование транспортных систем.

Наличие адекватных моделей позволяет проводить проверку гипотез при управлении такими сложными транспортными объектами как метрополитен. Разработка автоматизированной системы связана как с проверкой тех или иных алгоритмов, выбора архитектурных решений программного продукта так и с выбором структуры данных, которая используется при реализации выбранного алгоритма. Для хранения в оперативной памяти объектов обычно используют стандартные структуры данных такие как массивы, списки, стек или очередь [1]. Для реализации конкретных задач авторы, каждый при разработке своей конкретной автоматизированной системы, столкнулся с необходимостью использования структуры данных бинарное дерево поиска [2] или её разновидностью – деревом отрезков [3].

Двоичное дерево поиска позволяет производить поиск и вставку элементов с алгоритмической сложностью $O(log_2(n))$ в том случае, если дерево является сбалансированным, иначе, дерево может выродиться в список, и алгоритмическая сложность вышеуказанных операций увеличиться до $O(n)$.

Широко известными самобалансирующимися бинарными деревьями являются АВЛ-дерево и красно-черное дерево. АВЛ-дерево названо в честь его создателей – советских математиков Г. М. Адельсона-Вельского и Е. М. Ландиса. Балансировка АВЛ-дерева достигается одинарными и двойными левыми и правыми поворотами. В красно-черном дереве в балансировке участвует цвет узла и одиночные повороты.

Дерево отрезков (или дерево интервалов) представляет собой структуру данных, использующую сбалансированное двоичное дерево и хранящее интервалы в качестве значений [3].

Исходя из вышеизложенного были разработаны UML-диаграммы [4] интерфейсов и классов библиотеки.

Иерархия интерфейсов библиотеки представлена ниже (Рисунок 1).

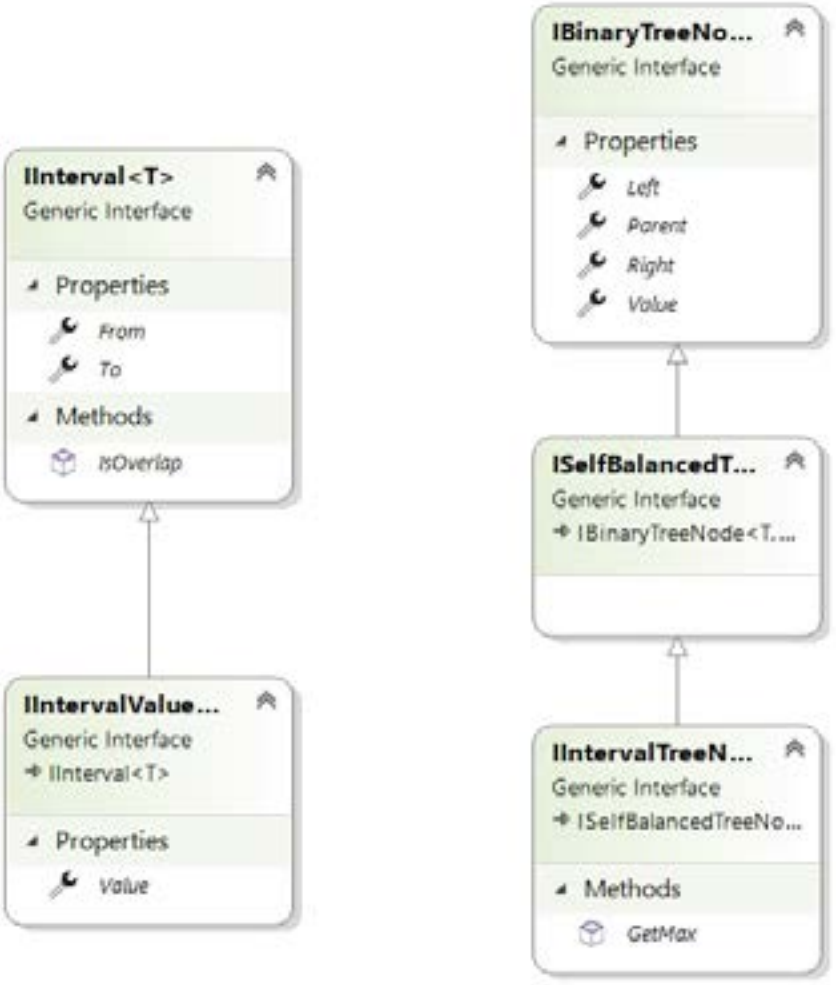

Рисунок 1 *– Иерархия интерфейсов библиотеки*

Для реализации библиотеки бинарных деревьев авторы разработали иерархию базовых классов узлов (Рисунок 2) и классы реализации узлов (Рисунок 3).

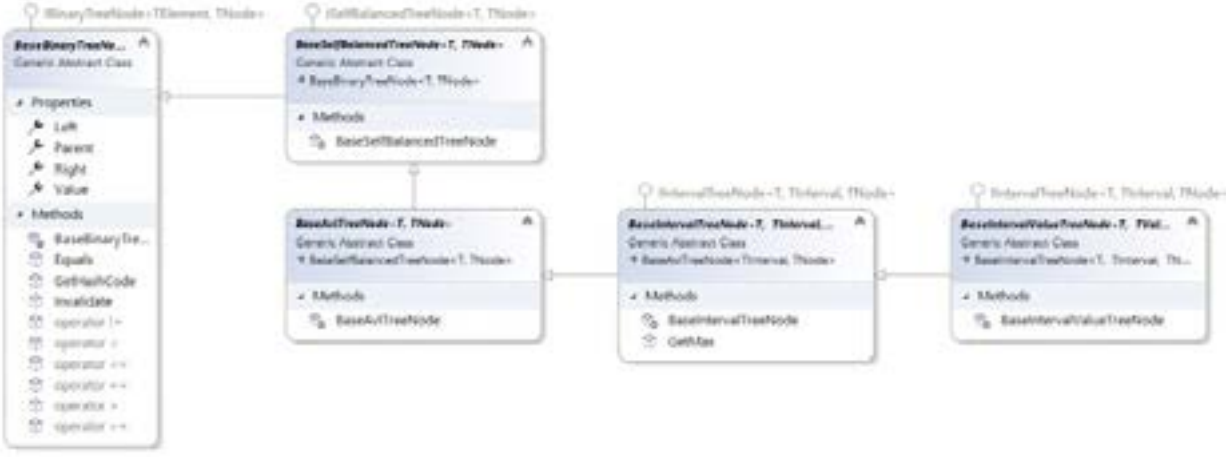

Рисунок 2 *– Иерархия базовых классов узла двоичного дерева поиска*

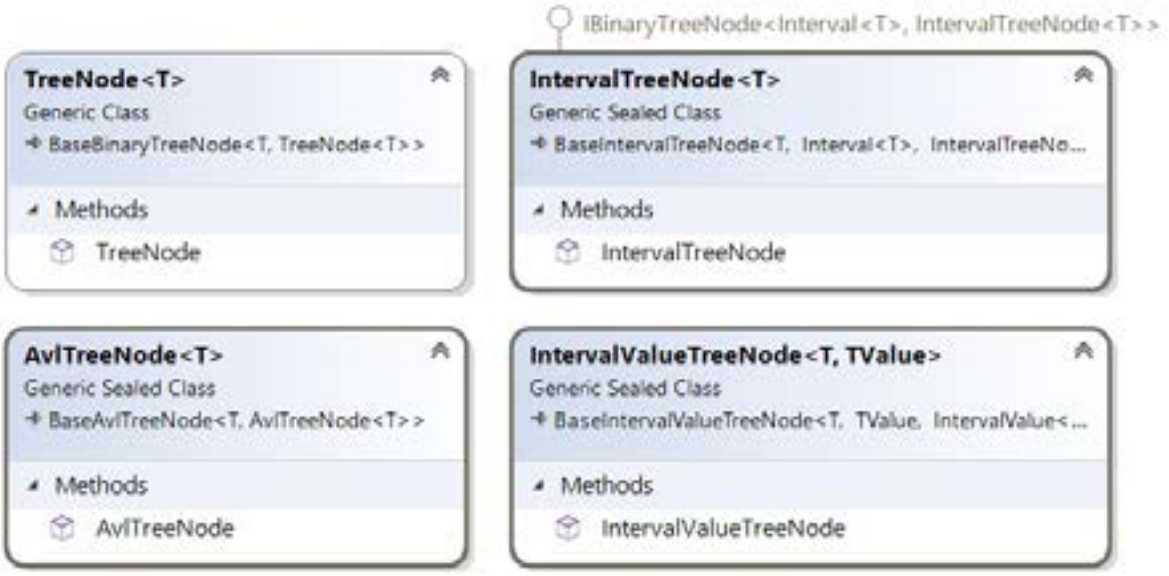

Рисунок 3 *– Классы реализации узла бинарного дерева поиска*

Иерархия базовых классов библиотеки деревьев и их конкретных реализаций представлены ниже (Рисунок 4, Рисунок 5).

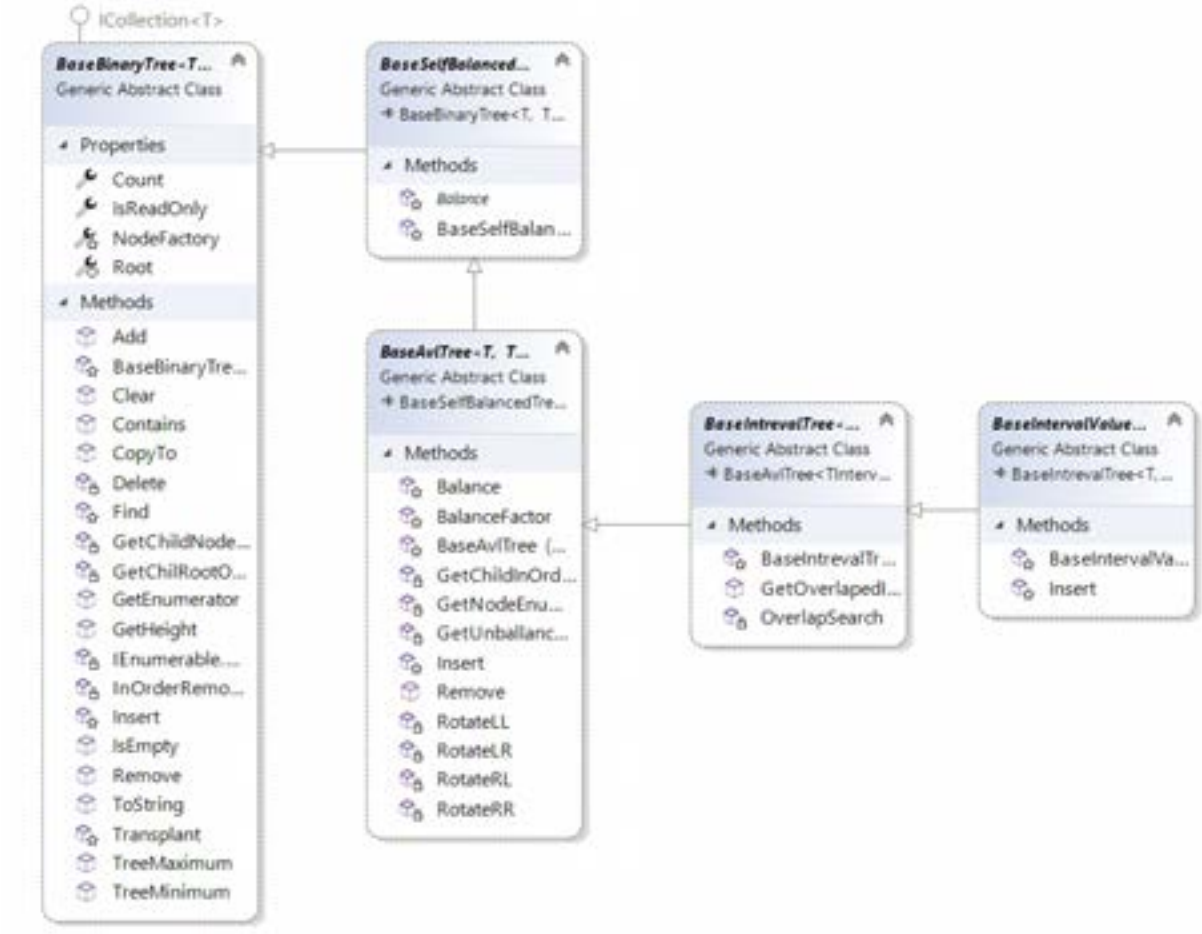

Рисунок 4 *– Иерархия базовых классов библиотеки бинарных деревьев поиска*

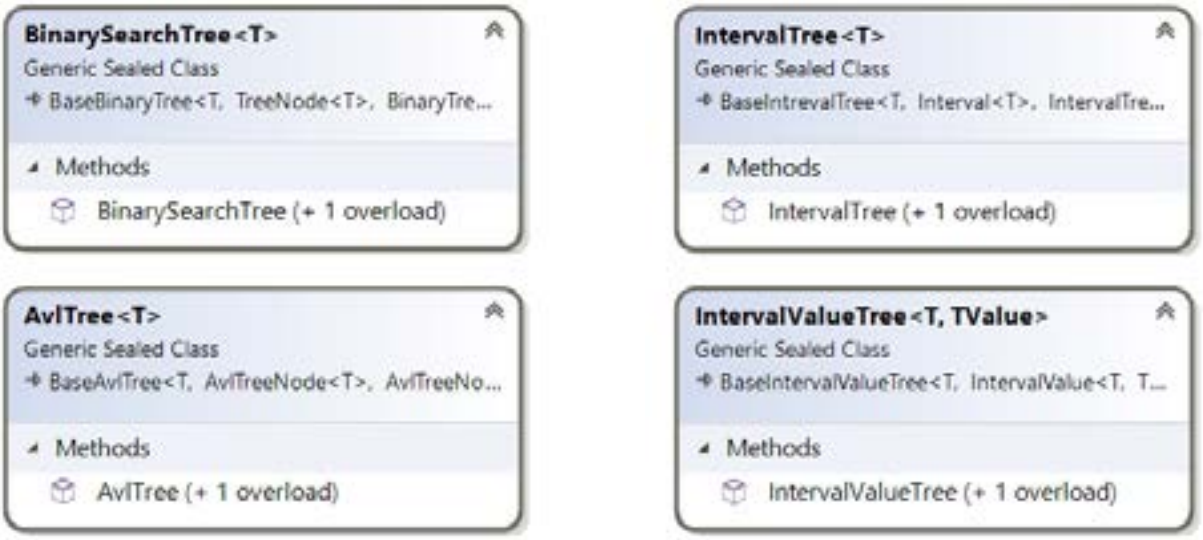

Рисунок 5 *– Классы реализации двоичных деревьев поиска*

Иерархия базовых классов библиотеки деревьев отрезков и их конкретная реализация представлены ниже (Рисунок 6, Рисунок 7).

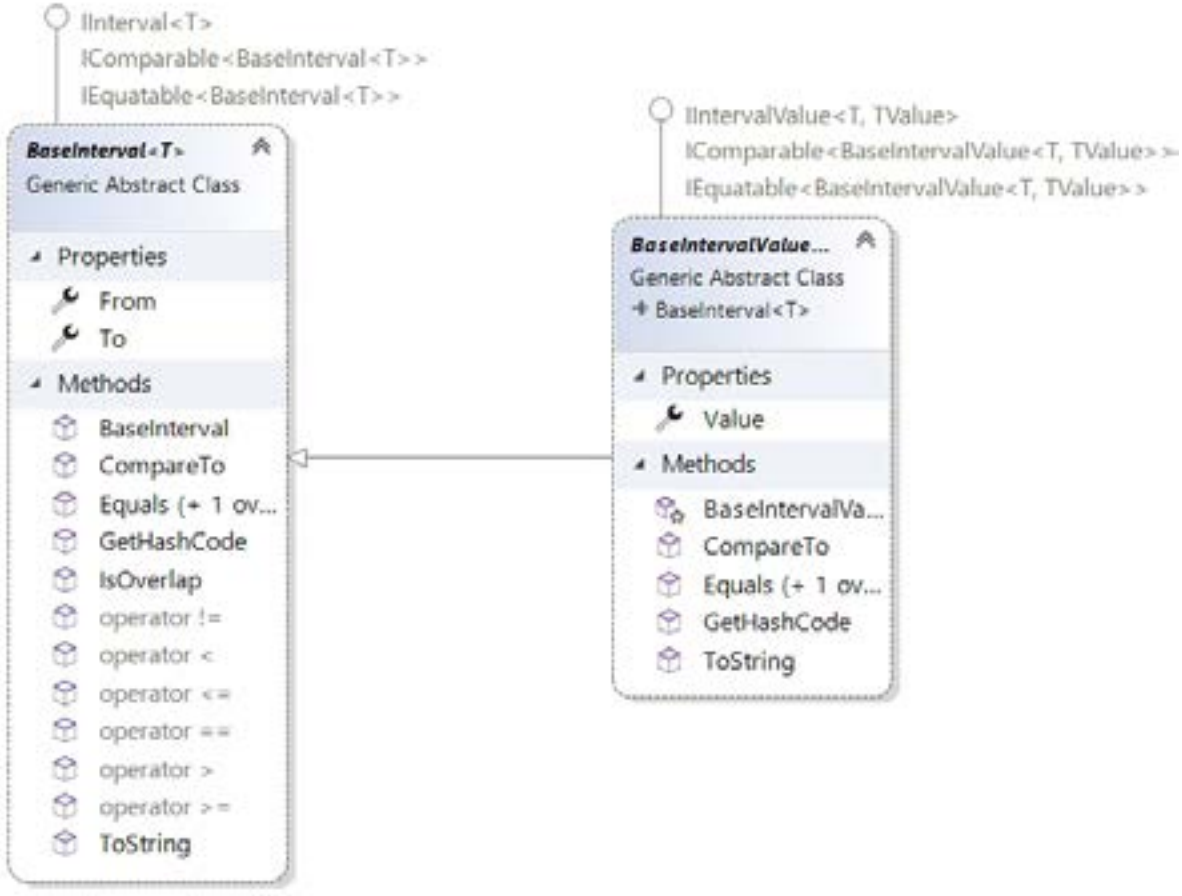

Рисунок 6 *– Иерархия базовых классов деревьев интервалов*

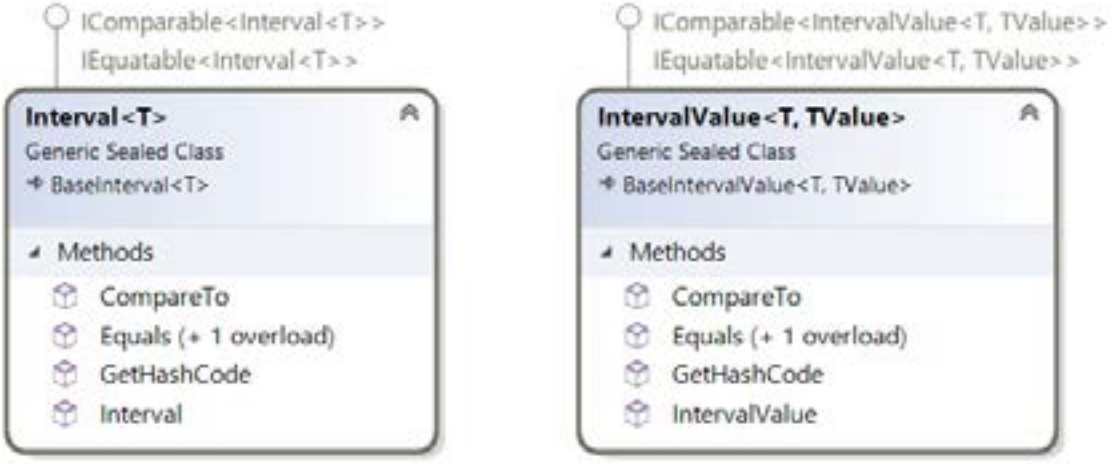

Рисунок 7 *– Библиотека классов деревьев интервалов*

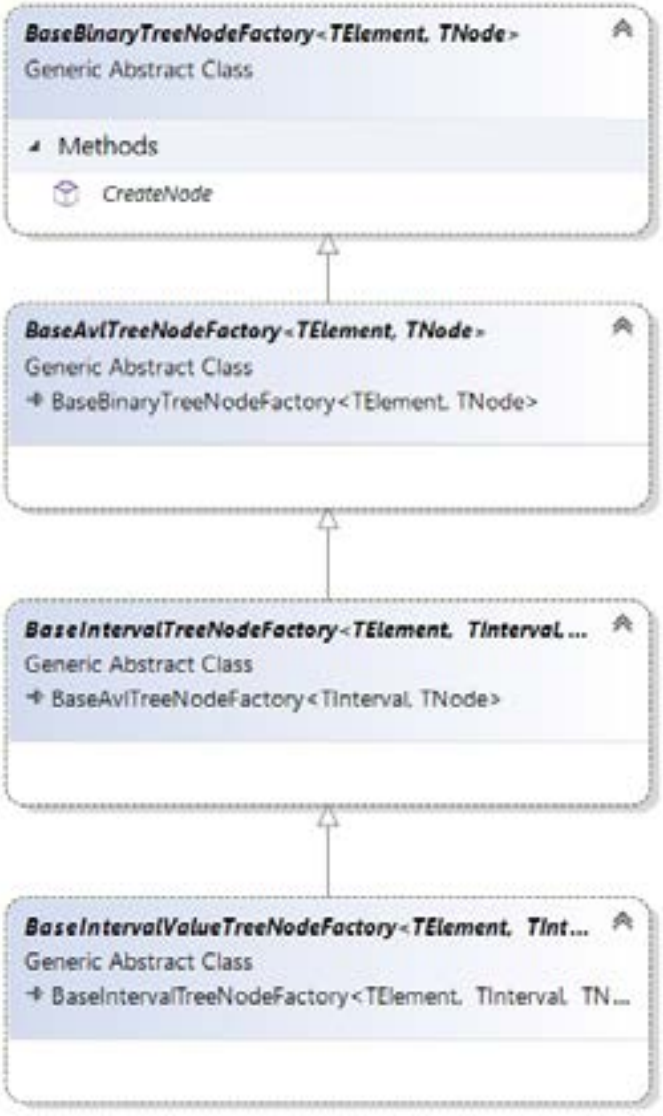

Рисунок 8 *– Иерархия базовых классов абстрактных фабрик узлов бинарных деревьев поиска*

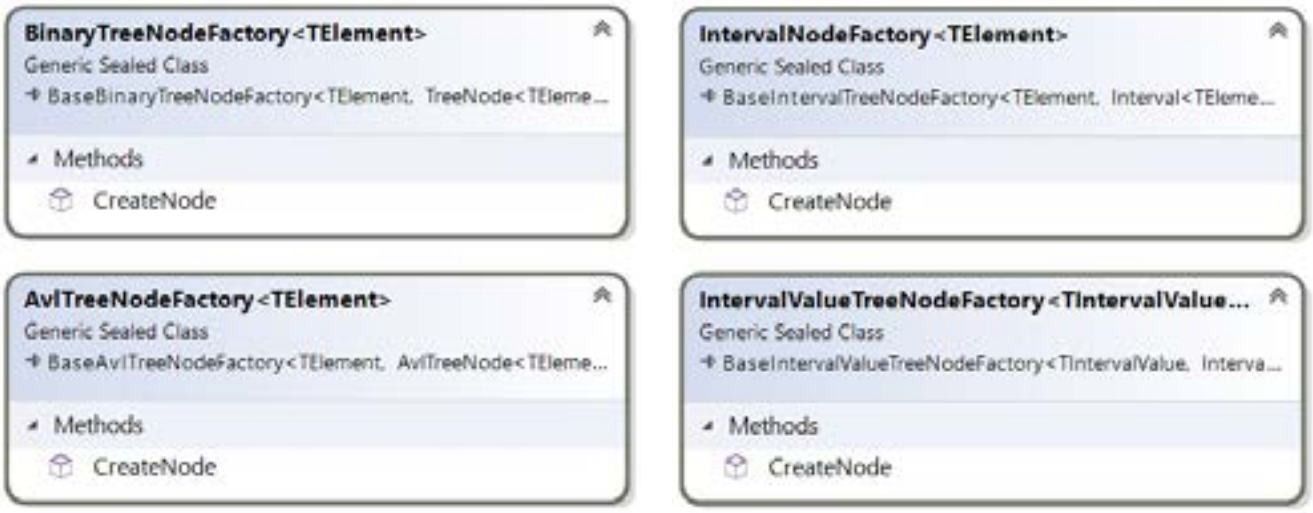

Рисунок 9 *– Иерархия конкретных реализаций классов фабрик узлов бинарных деревьев поиска*

Каждый класс дерева агрегирует класс своего конкретного узла. При реализации классов узлов авторы использовали архитектурное решение Абстрактная фабрика [5]. Иерархия базовых классов и их конкретная реализация представлены ниже (Рисунок 8, Рисунок 9).

Разработанное решение (`solution`) содержит четыре проекта: `Demo` для демонстрации работы библиотеки, `StageLibrary` – конкретное использование библиотеки деревьев интервалов для работы с ограничениями скорости на перегоне метрополитена, `Trees` – библиотека двоичных деревьев и `Trees.Tests` – библиотека тестов (Рисунок 10).

На разработанные классы написаны автоматизированные тесты [6]. Все публичные методы

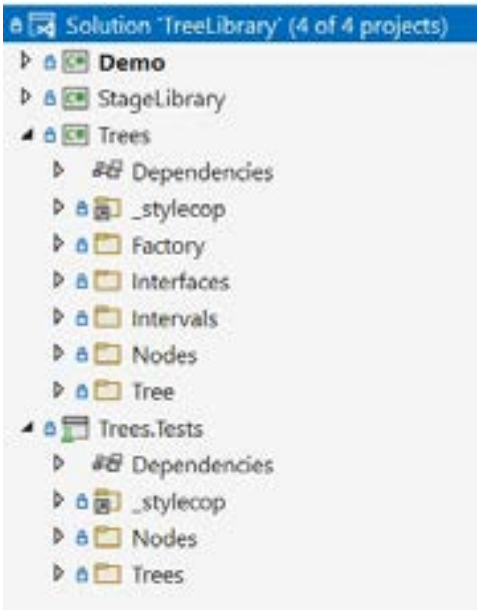

Рисунок 10 *– Разработанное решение с проектами*

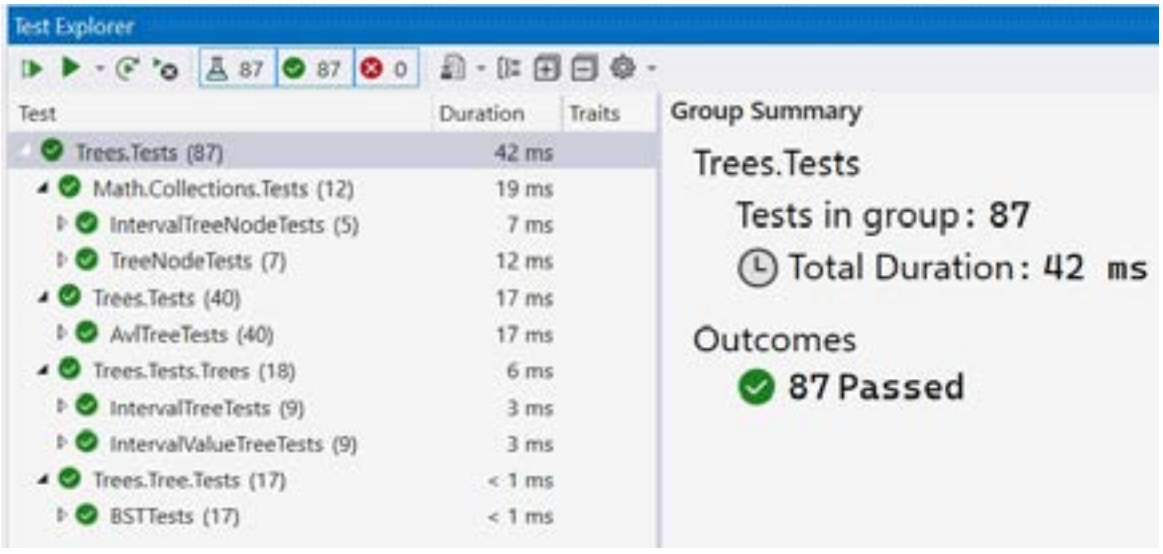

Рисунок 11 *– Инспектор тестов*

покрыты тестами с различными вариантами исхода (работа с примитивными типами данных или объектами, выбрасывает ли метод исключение при неправильных входных параметрах и т.д.).

Снимок экрана, на котором представлен Инспектор тестов представлен ниже (Рисунок 11). Как видно из рисунка, разработано 87 тестов, и все они прошли.

В результате проделанной работы авторами создана библиотека бинарных деревьев поиска и их разновидностью – деревьев интервалов. Данная библиотека внедряется в автоматизированные системы АСЭР (автоматизированная система выбора энергооптимальных режимов ведения) и АС ГТО (автоматизированная система «График технического обслуживания»).

References / Список использованных источников

[1] Рихтер Д. CLR via C# программирование на платформе Microsoft.NET FRAMEWORK 4.5 на языке C#. СПб: ООО Издательство «Питер», 2017.

[2] Двоичное дерево поиска [Электронный ресурс] // Википедия: [сайт]. [2021]. URL: https://ru.wikipedia.org/wiki/Двоичное_дерево_поиска (дата обращения: 22.05.2021).

[3] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 3-е изд. Москва: ООО «И.Д. Вильямс», 2014. 1328 с.

[4] Фаулер М. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. 3-е изд. Санкт-Петербург: Символ-Плюс, 2018. 192 с.

[5] Тепляков С. Паттерны проектирования на платформе.NET. СПб.: Питер, 2016. 320 с.

[6] Кент Б. Экстремальное программирование. Разработка через тестирование TDD. Спб.: Питер, 2020. 224 с.

Received: May 19, 2022 / Получено: 19 мая 2022 г.
Accepted: November 12, 2022 / Принято: 12 ноября 2022 г.

Сведения об авторах

Васильева Марина Алексеевна – доцент кафедры «Управление и защита информации» Российского университета транспорта (МИИТ).
ученая степень: кандидат технических наук
ученое звание: доцент
e-mail: marina_paley@mail.ru

Филипченко Константин Михайлович – разработчик бизнес-приложений ООО «Комонвелс Партнершип».

For citation

Vasilyeva Marina Alekseevna, Filipchenko Konstantin Mikhailovich, Application of Tree Data Structures for Systems Modeling, *JITA – Journal of Information Technology and Applications, Banja Luka*, Pan-Europien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 12(2022) 2:152-165, (UDC: 629.7.014.9:623.746.2-519), (DOI: 10.7251/JIT2202152A), Volume 12, Number 2, Banja Luka, December (65-172), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004