Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 6

з дисципліни «Мова програмування Java»

Тема: «Collections»

Виконала:

Студентка групи ІА-31

Соколова Поліна

Київ – 2025

Мета роботи – практичне освоєння роботи з фреймворком колекцій в Java, доцільність вибору конкретних реалізацій та особливості їх роботи.

Завдання:
Напишіть консольний додаток, який:
- описує клас, який реалізує червоно-чорне дерево: додавання, обхід і видалення (варіативно) вузлів;
- створює екземпляр дерева і наповнює його елементами(також підтримує введення даних з клавіатури), як елементи можна використовувати числа:
• числа подаються в довільному порядку (можна використовувати масив, який заповнити випадковими числами; в цьому випадку масив потрібно вивести, щоб бачити порядок додавання елементів);
• числа подаються впорядковано, наприклад, по зростанню (можна впорядкувати масив, заповнений випадковими числами);
- відображає дерево.

Хід роботи

Реалізовано узагальнене червоно-чорне дерево RedBlackTree<T extends Comparable<T>>, яке підтримує базові операції над впорядкованою колекцією: вставку, пошук, видалення, обходи та графічне відображення структури дерева.

У класі дерева використано вкладений клас Node, який містить ключ, посилання на батьківський, лівий і правий вузли, а також колір вузла (RED або BLACK). Колір визначений у вигляді перерахування enum Color.

Для вставки нового елемента використовується типова логіка бінарного дерева пошуку: новий ключ порівнюється з поточним і переходить у ліве або праве піддерево. Дублікатів не допускається, перед вставкою виконується перевірка методом contains(). Після вставки викликається метод fixInsert(), який усуває можливі порушення властивостей червоно-чорного дерева. У цьому методі реалізовано всі основні випадки алгоритму: перекольоровування вузлів та необхідні обертання (rotateLeft та rotateRight).

Пошук елемента (contains і searchNode) реалізовано прямим спуском по дереву згідно правил бінарного дерева пошуку.

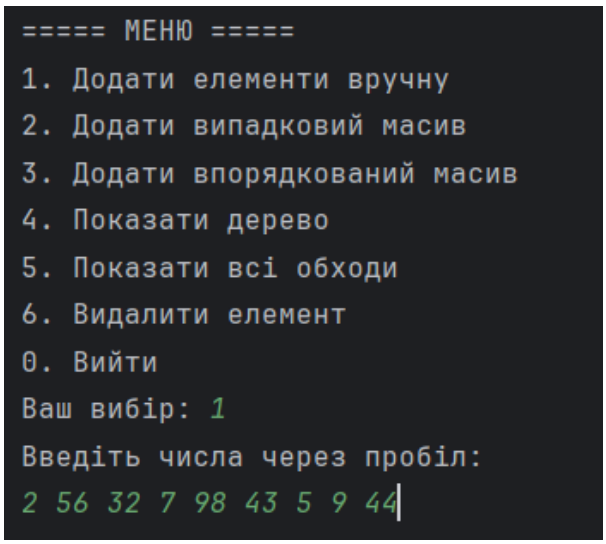
Видалення вузла реалізовано у методі delete(T key), який повертає логічне значення, що вказує на успішність операції. Видалення виконується за стандартним алгоритмом: залежно від кількості дітей вузла застосовується одна з трьох стратегій: видалення листка, заміна вузлом-нащадком або заміна вузлом-наступником. Допоміжний метод transplant використовується для коректної заміни піддерев. Після фізичного видалення викликається метод fixDelete(), який відновлює властивості червоно-чорного дерева. Реалізовано всі випадки алгоритму з урахуванням ситуацій, коли один або декілька нащадків можуть бути null. Для цього застосовані перевірки isBlack() та isRed().

Для відображення структури дерева реалізовано метод printTree(), який формує схематичне зображення з позначенням лівих і правих гілок (L та R), а також кольору вузлів (R або B).

Також реалізовано три класичні алгоритми обходу дерева: симетричний (inorder), прямий (preorder) та зворотний (postorder).

У головному класі Main створено консольне меню, яке дозволяє взаємодіяти з деревом: додавати значення вручну або генерувати випадкові/впорядковані набори чисел, відображати дерево і виконувати обходи, а також видаляти окремі елементи.

Результат:



```
Ваш вибір: 4
└── 32(B)
    ├──L─ 5(R)
    │   ├──L─ 2(B)
    │   └──R─ 7(B)
    │       └──R─ 9(R)
    └──R─ 56(R)
        ├──L─ 43(B)
        │   └──R─ 44(R)
        └──R─ 98(B)
```

```
Ваш вибір: 5
Inorder:   2 5 7 9 32 43 44 56 98
Preorder:  32 5 2 7 9 56 43 44 98
Postorder: 2 9 7 5 44 43 98 56 32
```

```
Ваш вибір: 6
Введіть значення для видалення: 43
Елемент видалено

===== МЕНЮ =====
1. Додати елементи вручну
2. Додати випадковий масив
3. Додати впорядкований масив
4. Показати дерево
5. Показати всі обходи
6. Видалити елемент
0. Вийти
Ваш вибір: 4
└── 32(B)
    ├──L─ 5(R)
    │   ├──L─ 2(B)
    │   └──R─ 7(B)
    │       └──R─ 9(R)
    └──R─ 56(R)
        ├──L─ 44(B)
        └──R─ 98(B)
```

```
Ваш вибір: 2
Кількість елементів: 5
Максимальне значення: 60
Згенеровано: 45 30 50 40 53

===== МЕНЮ =====
1. Додати елементи вручну
2. Додати випадковий масив
3. Додати впорядкований масив
4. Показати дерево
5. Показати всі обходи
6. Видалити елемент
0. Вийти
Ваш вибір: 4
└── 45(B)
    ├──L─ 30(B)
    │   └──R─ 40(R)
    └──R─ 50(B)
        └──R─ 53(R)
```

```
Ваш вибір: 3
Кількість елементів: 6
Максимальне значення: 90
Впорядкований масив: 7 37 62 69 71 79

===== МЕНЮ =====
1. Додати елементи вручну
2. Додати випадковий масив
3. Додати впорядкований масив
4. Показати дерево
5. Показати всі обходи
6. Видалити елемент
0. Вийти
Ваш вибір: 5
Inorder:   7 37 62 69 71 79
Preorder:  37 7 69 62 71 79
Postorder: 7 62 79 71 69 37
```

Висновок: у результаті виконання лабораторної роботи я реалізувала червоно-чорне дерево, яке підтримує основні операції впорядкованої колекції: вставку, пошук, видалення та обходи. Реалізовано візуалізацію дерева та створено консольне меню для взаємодії з користувачем. Отримана програма дозволяє на практиці розібратися з принципами роботи червоно-чорних дерев і особливостями їх використання у фреймворку колекцій Java.

Посилання на GitHub репозиторій:
https://github.com/sokolovapolina230/Java_labs/tree/main