МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФРАСТРУКТУРИ ТА ТЕХНОЛОГІЙ
ІНСТИТУТ УПРАВЛІННЯ, ТЕХНОЛОГІЙ ТА ПРАВА
ФАКУЛЬТЕТ УПРАВЛІННЯ І ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

Лабораторна робота №4

з дисципліни «Фундаментальні комп’ютерні алгоритми»

з теми: «АЛГОРИТМИ РОБОТИ З НЕЛІНІЙНОЮ СТРУКТУРОЮ ДАНИХ «ДЕРЕВО»»

Варіант 13

Виконав:
Студент групи
ІПЗд-23121 маг.
Петренко Д.М.
Перевірив:
Доцент кафедри ІТ
Ткаченко О.А.

Київ – 2024

Лабораторна робота №4

Мета: набути практичних навичок зі створення й обробки бінарних дерев.

ЗАВДАННЯ:

Виконати такі дії:

- описати студента згідно з варіантом завдання (Прізвище, ім’я, курс, студентським квиток, служба в армії);
- описати бінарне дерево;
- створити екземпляр бінарного дерева;
- додавати елементи в кінець бінарного дерева (в якості листів);
- вивести вміст дерева згідно з заданим способом обходу дерева (Preorder).

ХІД РОБОТИ:

Створимо клас Student, де буде п’ять полів, які представляють інформацію про студента:

- lastName (Прізвище) - тип String, що зберігає прізвище студента.
- firstName (Ім’я) - тип String, що зберігає ім’я студента.
- course (Курс) - тип int, що вказує на курс, на якому навчається студент.
- studentId (Студентський квиток) - тип int, що є унікальним ідентифікатором студента.
- armyService (Служба в армії) - тип boolean, що вказує, чи служив студент в армії.

```
class Student {
    String lastName;
    String firstName;
    int course;
    int studentId;
    boolean armyService;

    public Student(String lastName, String firstName, int course, int studentId, boolean armyService) {
        this.lastName = lastName;
        this.firstName = firstName;
        this.course = course;
        this.studentId = studentId;
        this.armyService = armyService;
    }

    @Override
    public String toString() {
        return String.format(format:"(%s, %s, %d курс, ID: %d, Служив: %b)", lastName, firstName, course, studentId, armyService);
    }
}
```

Після оголошення класу Student, йде конструктор класу `public Student(String lastName, String firstName, int course, int studentId, boolean armyService).` Конструктор класу Student приймає п’ять параметрів, які відповідають полям класу, та ініціалізує ці поля. Використання ключового слова **this** допомагає відрізнити поля класу від параметрів конструктора.

@Override - це аннотація в Java, яка вказує компілятору, що метод, який слідує за аннотацією, перевизначає метод з класу-батька або інтерфейсу.

Метод **toString()** перевизначається з класу **Object**, який є базовим класом для всіх класів в Java.

Він використовується для отримання рядкового представлення об’єкта Student. Метод **String.format** використовується для форматування інформації про студента в читабельний рядок з використанням плейсхолдерів (%s для рядків, %d для цілих чисел та %b для булевих значень).

Клас TreeNode:

```
class TreeNode {
    Student student;
    TreeNode left;
    TreeNode right;

    public TreeNode(Student student) {
        this.student = student;
        this.left = null;
        this.right = null;
    }
}
```

TreeNode - це клас, який представляє вузол бінарного дерева. Розглянемо кожну частину цього класу:

Поля:

- **student**: Об’єкт типу Student, який містить інформацію про студента (прізвище, ім’я, курс, студентський квиток, служба в армії).
- **left**: Вказівник на лівого нащадка (лівий піддерево).
- **right**: Вказівник на правого нащадка (праве піддерево).

Конструктор:

- Конструктор приймає об’єкт Student і ініціалізує поле student цим об’єктом.
- Початкові значення для left та right встановлюються як null.

Клас BinaryTree:

```
class BinaryTree {
    TreeNode root;

    public BinaryTree() {
        this.root = null;
    }

    public void insert(Student student) {
        root = insertRec(root, student);
        System.out.println(x:"Вміст бінарного дерева після додавання:");
        printTree(root, indent:"", isLeft:true);
        System.out.println();
    }

    // Метод для відображення бінарного дерева у вигляді дерева консольного виводу
    public void printTree(TreeNode node, String indent, boolean isLeft) {
        if (node != null) {
            System.out.print(indent);
            if (isLeft)
                System.out.print(s:"<-- ");
            else
                System.out.print(s:"-->  ");
            System.out.println(node.student);
            printTree(node.right, indent + (isLeft ? "|   " : "    "), isLeft:false);
            printTree(node.left, indent + (isLeft ? "|   " : "    "), isLeft:true);
        }
    }
```

```
    private TreeNode insertRec(TreeNode root, Student student) {
        if (root == null) {
            root = new TreeNode(student);
            return root;
        }

        if (student.studentId < root.student.studentId)
            root.left = insertRec(root.left, student);
        else if (student.studentId > root.student.studentId)
            root.right = insertRec(root.right, student);

        return root;
    }

    // Прямий обхід дерева для виведення в консоль
    public void preorderTraversalPrint(TreeNode node) {
        if (node != null) {
            System.out.println(node.student);
            preorderTraversalPrint(node.left);
            preorderTraversalPrint(node.right);
        }
    }
}
```

Розглянемо все окремо:

```
class BinaryTree {
    TreeNode root;
```

Поле root: Це поле є вказівником на кореневий вузол бінарного дерева. Воно ініціалізується як null, що означає, що дерево спочатку порожнє.

```
public BinaryTree() {
    this.root = null;
}
```

Конструктор: Конструктор без параметрів створює порожнє бінарне дерево, встановлюючи поле root як null.

```
public void insert(Student student) {
    root = insertRec(root, student);
    System.out.println(x:"Вміст бінарного дерева після додавання:");
    printTree(root, indent:"", isLeft:true);
    System.out.println();
}
```

Метод insert: Цей метод відповідає за додавання нового студента до бінарного дерева. Він викликає приватний рекурсивний метод insertRec, передаючи йому кореневий вузол та об’єкт студента, якого потрібно додати.

Після вставки студента, метод виводить вміст бінарного дерева в консоль, використовуючи метод printTree. Це дозволяє візуально перевірити структуру дерева після кожного додавання.

```
// Метод для відображення бінарного дерева у вигляді дерева консольного виводу
public void printTree(TreeNode node, String indent, boolean isLeft) {
    if (node != null) {
        System.out.print(indent);
        if (isLeft)
            System.out.print(s:"<-- ");
        else
            System.out.print(s:"-->  ");
        System.out.println(node.student);
        printTree(node.right, indent + (isLeft ? "|   " : "    "), isLeft:false);
        printTree(node.left, indent + (isLeft ? "|   " : "    "), isLeft:true);
    }
}
```

Цей метод відповідає за відображення бінарного дерева у вигляді дерева консольного виводу. Розглянемо його детальніше:

1. *Параметри методу:*
   - ***node***: Поточний вузол, з якого починається відображення.
   - ***indent***: Рядок, що визначає відступ для поточного рівня вузлів.
   - ***isLeft***: Логічне значення, яке вказує, чи є поточний вузол лівим нащадком.
2. *Базовий випадок (якщо node дорівнює null):*
   - Якщо node є null, обхід завершується.
3. *Виведення поточного вузла в консоль:*
   - Виводимо відступ (indent), який залежить від рівня вузла.
   - Якщо вузол є лівим нащадком, виводимо стрілку <--.
   - Якщо вузол є правим нащадком, виводимо стрілку -->.
   - Виводимо інформацію про студента, який міститься в поточному вузлі.

4. Рекурсивний виклик для лівого та правого піддерева:

- Рекурсивно викликаємо printTree для правого піддерева (node.right), збільшуючи відступ на один рівень.
- Рекурсивно викликаємо printTree для лівого піддерева (node.left), також збільшуючи відступ на один рівень.

Цей метод дозволяє вивести структуру бінарного дерева в зручному для сприйняття форматі, де стрілки вказують на зв'язки між вузлами, а інформація про студентів виводиться поруч з відповідними вузлами.

```
private TreeNode insertRec(TreeNode root, Student student) {
    if (root == null) {
        root = new TreeNode(student);
        return root;
    }

    if (student.studentId < root.student.studentId)
        root.left = insertRec(root.left, student);
    else if (student.studentId > root.student.studentId)
        root.right = insertRec(root.right, student);

    return root;
}
```

Ця частина коду в Java є реалізацією приватного методу insertRec(), який використовується для вставки нового вузла з даними студента в бінарне дерево пошуку. Розглянемо кожен крок цього методу детально:

1. Параметри методу:

- **root**: Поточний вузол дерева, який може бути коренем піддерева або null.
- **student**: Об'єкт класу Student, який містить дані студента, що потрібно вставити в дерево.

2. Перевірка на null:

- Якщо root дорівнює null, це означає, що ми досягли позиції в дереві, де можемо вставити новий вузол.
- Створюємо новий вузол з об'єктом student і повертаємо його як корінь піддерева.

3. Вставка в ліве або праве піддерево:

- Якщо studentId об’єкта student менше, ніж studentId поточного вузла (root), то виконуємо рекурсивний виклик insertRec для лівого піддерева (root.left).
- Якщо studentId об’єкта student більше, ніж studentId поточного вузла, то виконуємо рекурсивний виклик insertRec для правого піддерева (root.right).

4. Збереження властивостей бінарного дерева пошуку:

- Властивості бінарного дерева пошуку вимагають, щоб усі ліві нащадки вузла мали значення менше, ніж сам вузол, а всі праві нащадки — більше.
- Цей метод забезпечує дотримання цих властивостей під час вставки, порівнюючи studentId і розміщуючи студентів у відповідні позиції в дереві.

5. Повернення кореня піддерева:

- Після вставки нового вузла або проходження до відповідного місця в дереві, метод повертає корінь піддерева.
- Це дозволяє оновити посилання на ліві та праві піддерева в батьківських вузлах.

Загалом, цей метод insertRec є ключовим для побудови та підтримки структури бінарного дерева пошуку, де кожен вузол має унікальний ідентифікатор studentId, що використовується для визначення його позиції в дереві.

Метод рекурсивний, що означає, що він викликає сам себе з новими параметрами до тих пір, поки не буде знайдено правильне місце для вставки нового студента.

```java
// Прямий обхід дерева для виведення в консоль
public void preorderTraversalPrint(TreeNode node) {
    if (node != null) {
        System.out.println(node.student);
        preorderTraversalPrint(node.left);
        preorderTraversalPrint(node.right);
    }
}
```

Ця частина коду в Java реалізує метод preorderTraversalPrint(), який виконує прямий обхід (preorder traversal) бінарного дерева та виводить інформацію про студентів, збережених у вузлах дерева.

Розглянемо цей процес детально:

1. Параметр методу:

- **node**: Поточний вузол дерева, з якого починається обхід.

2. Перевірка на null:

- Перш за все, метод перевіряє, чи поточний вузол не є null. Якщо вузол є null, це означає, що ми досягли кінця гілки, і метод завершується.

3. Виведення інформації про студента:

- Якщо вузол не є null, метод виводить інформацію про студента, яка міститься в цьому вузлі, використовуючи System.out.println(node.student). Це відображає дані студента в консоль.

4. Рекурсивний виклик для лівого піддерева:

- Після виведення інформації про студента, метод рекурсивно викликає сам себе для лівого нащадка поточного вузла (node.left). Це означає, що метод спочатку відвідає всі вузли лівого піддерева, перш ніж перейти до правого піддерева.

5. Рекурсивний виклик для правого піддерева:

- Після завершення обходу лівого піддерева, метод рекурсивно викликає сам себе для правого нащадка поточного вузла (node.right).

Клас Main:

```
public class Main {
    Run | Debug
    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();

        // Додавання студентів у бінарне дерево
        tree.insert(new Student(lastName:"Петров", firstName:"Іван", course:3, studentId:123, armySer…false));
        tree.insert(new Student(lastName:"Іванов", firstName:"Петро", course:2, studentId:456, armySer…true));
        tree.insert(new Student(lastName:"Сидорова", firstName:"Марія", course:4, studentId:789, armySer…false));
        tree.insert(new Student(lastName:"Ковальчук", firstName:"Олексій", course:1, studentId:111, armySer…true));
        tree.insert(new Student(lastName:"Бондаренко", firstName:"Анна", course:3, studentId:222, armySer…false));
        tree.insert(new Student(lastName:"Коваленко", firstName:"Ігор", course:2, studentId:333, armySer…true));
        tree.insert(new Student(lastName:"Мельник", firstName:"Олена", course:4, studentId:444, armySer…false));

        // Виведення вмісту дерева згідно з прямим обходом у консоль
        System.out.println(x:"\nПрямий обхід:");
        tree.preorderTraversalPrint(tree.root);
    }
}
```

1. Створення об’єкту BinaryTree:

- Спочатку створюється об’єкт класу BinaryTree з іменем tree.

2. Додавання студентів у бінарне дерево:

- Викликаються методи insert для додавання об’єктів Student до бінарного дерева.
- Кожен студент має свої дані, такі як прізвище, ім’я, курс, студентський квиток та інформацію про службу в армії.

3. Виведення вмісту дерева згідно з прямим обходом:

- Після додавання студентів в бінарне дерево викликається метод preorderTraversalPrint, який виводить дані про студентів в порядку прямого обходу (preorder traversal).
- Прямий обхід означає, що спочатку виводиться інформація про поточний вузол, а потім рекурсивно виводяться дані лівого та правого піддерева.

РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ:

Вводимо початкові дані студентів:

```
// Додавання студентів у бінарне дерево
tree.insert(new Student(lastName:"Петров", firstName:"Іван", course:3, studentId:123, armySer…false));
tree.insert(new Student(lastName:"Іванов", firstName:"Петро", course:2, studentId:456, armySer…true));
tree.insert(new Student(lastName:"Сидорова", firstName:"Марія", course:4, studentId:789, armySer…false));
tree.insert(new Student(lastName:"Ковальчук", firstName:"Олексій", course:1, studentId:111, armySer…true));
tree.insert(new Student(lastName:"Бондаренко", firstName:"Анна", course:3, studentId:222, armySer…false));
tree.insert(new Student(lastName:"Коваленко", firstName:"Ігор", course:2, studentId:333, armySer…true));
tree.insert(new Student(lastName:"Мельник", firstName:"Олена", course:4, studentId:444, armySer…false));
```

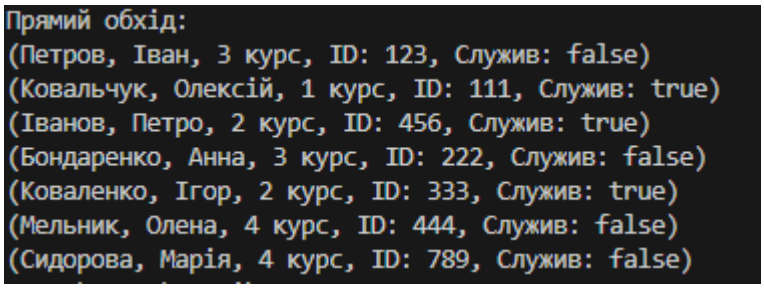
Натискаємо команду Run та отримуємо результат:

1. Виконується додавання студента до бінарного дерева:

```
Вміст бінарного дерева після додавання:
<-- (Петров, Іван, 3 курс, ID: 123, Служив: false)

Вміст бінарного дерева після додавання:
<-- (Петров, Іван, 3 курс, ID: 123, Служив: false)
    -->  (Іванов, Петро, 2 курс, ID: 456, Служив: true)

Вміст бінарного дерева після додавання:
<-- (Петров, Іван, 3 курс, ID: 123, Служив: false)
    -->  (Іванов, Петро, 2 курс, ID: 456, Служив: true)
        -->  (Сидорова, Марія, 4 курс, ID: 789, Служив: false)

Вміст бінарного дерева після додавання:
<-- (Петров, Іван, 3 курс, ID: 123, Служив: false)
    -->  (Іванов, Петро, 2 курс, ID: 456, Служив: true)
        -->  (Сидорова, Марія, 4 курс, ID: 789, Служив: false)
    <-- (Ковальчук, Олексій, 1 курс, ID: 111, Служив: true)

Вміст бінарного дерева після додавання:
<-- (Петров, Іван, 3 курс, ID: 123, Служив: false)
    -->  (Іванов, Петро, 2 курс, ID: 456, Служив: true)
        -->  (Сидорова, Марія, 4 курс, ID: 789, Служив: false)
        <-- (Бондаренко, Анна, 3 курс, ID: 222, Служив: false)
    <-- (Ковальчук, Олексій, 1 курс, ID: 111, Служив: true)
```

```
Вміст бінарного дерева після додавання:
<-- (Петров, Іван, 3 курс, ID: 123, Служив: false)
    -->  (Іванов, Петро, 2 курс, ID: 456, Служив: true)
        -->  (Сидорова, Марія, 4 курс, ID: 789, Служив: false)
        <-- (Бондаренко, Анна, 3 курс, ID: 222, Служив: false)
            -->  (Коваленко, Ігор, 2 курс, ID: 333, Служив: true)
    <-- (Ковальчук, Олексій, 1 курс, ID: 111, Служив: true)

Вміст бінарного дерева після додавання:
<-- (Петров, Іван, 3 курс, ID: 123, Служив: false)
    -->  (Іванов, Петро, 2 курс, ID: 456, Служив: true)
        -->  (Сидорова, Марія, 4 курс, ID: 789, Служив: false)
        <-- (Бондаренко, Анна, 3 курс, ID: 222, Служив: false)
            -->  (Коваленко, Ігор, 2 курс, ID: 333, Служив: true)
                -->  (Мельник, Олена, 4 курс, ID: 444, Служив: false)
    <-- (Ковальчук, Олексій, 1 курс, ID: 111, Служив: true)
```

2. Виводиться список студентів отриманих прямим обходом:



ВИСНОВОК:

В ході лабораторної роботи було набуто практичних навичок зі створення й обробки бінарних дерев. Було створено програму на мові програмування Java, що виконує опис студента (за відповідним варіантом), описує бінарне дерево, створює екземпляр бінарного дерева, виводячи покрокове додавання студентів. Дана програма виконує виведення вмісту дерева згідно з заданим способом обходу дерева (прямим).