Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

З лабораторної роботи №3

Варіант – 9

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Спадкування та інтерфейси»

Виконав: ст. гр. КІ-305

Заставний Р.А.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів-2023

Мета роботи: ознайомитися з спадкуванням та інтерфейсами у мові Java.

ЗАВДАННЯ

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті `Група.Прізвище.Lab3` та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

Варіант завдання:

9. Дерево

Код програми:

Main.java

```
package KI305.Zastavnyi.Lab3;

import java.io.IOException;
/**
 *
 * Main class implements main method for Tree class possibilities demonstration
 *
 * @author Roman Zastavnyi
 *
 */
public class Main {
	public static void main(String[] args) throws IOException {
		// Створення екземпляра класу Tree
		Tree tree = new Tree();

		tree.getStatus(); // Виклик методу інформації про дерево
		tree.Death(); //Виклик методу смерті дерева

		// Створення іншого екземпляру класу Tree з конкретними параметрами
		Tree tree1 = new Tree("Вишня","Коричневий",33, "Листопадний");
		tree1.getStatus(); // Виклик методу інформації про дерево
		tree1.finish();

		// Створення екземпляра класу Tree з більшою кількістю параметрів
		Tree tree2 = new Tree("Яблуня", new Size(1, 2, 3), "Зелений",
				new Fruit(true, "Зелений", "Стиглі"), new Leaf("Серцеподібна", true,
"Зелений"), 16, 10, "Солоденьке");

		tree2.getStatus();
		System.out.println("Назва рослини tree2: "+ tree2.getName()); // Вивід назви
дерева tree2
		tree2.finish();
	}
}
```

CheckHumidity.java

```java
package KI305.Zastavnyi.Lab3;

/**
 * Інтерфейс CheckHumidity визначає метод для перевірки вологості перед зростанням рослини.
 */
public interface CheckHumidity {

    /**
     * Метод HumidityToGrow повертає true, якщо вологість відповідає умовам для зростання рослини, інакше - false.
     *
     * @return true, якщо вологість дозволяє рослині рости, false - якщо недостатньо вологості.
     */
    boolean HumidityToGrow();
}
```

GrowUp.java

```java
package KI305.Zastavnyi.Lab3;
/**
 * Інтерфейс GrowUp визначає метод для зростання рослини.
 * Розширює інтерфейс CheckHumidity для перевірки вологості перед зростанням.
 */
public interface GrowUp extends CheckHumidity {

    /**
     * Метод Grow визначає процес зростання рослини.
     */
    void Grow();
}
```

Fruit.java

```java
package KI305.Zastavnyi.Lab3;

import java.io.PrintWriter;

import static java.lang.System.out;

/**
 * Клас Fruit представляє характеристики плоду рослини.
 * Включає в себе інформацію про наявність, колір та стиглість плоду.
 * Також надає метод для отримання та виведення статусу плоду.
 */
public class Fruit {
    // Властивості плоду
    private boolean existence;
    private String color;
    private String ripeness;

    /**
     * Конструктор за замовчуванням ініціалізує плід значеннями за замовчуванням.
     */
    public Fruit() {
        this.existence = false;
        this.color = "Немає";
```

```
        this.ripeness = "Немає";
    }

    /**
     * Конструктор із вказаною наявністю, коліром та стиглістю плоду.
     *
     * @param existence Наявність плоду.
     * @param color     Колір плоду.
     * @param ripeness  Стиглість плоду.
     */
    public Fruit(boolean existence, String color, String ripeness) {
        this.existence = existence;
        this.color = color;
        this.ripeness = ripeness;
    }

    /**
     * Метод, який повертає стан наявності плоду.
     *
     * @return true, якщо плід існує, false, якщо плід відсутній.
     */
    public boolean isExistence() {
        return existence;
    }

    /**
     * Метод для встановлення стану наявності плоду.
     *
     * @param existence true, якщо плід існує, false, якщо плід відсутній.
     */
    public void setExistence(boolean existence) {
        this.existence = existence;
    }

    /**
     * Метод, який повертає колір плоду.
     *
     * @return Колір плоду.
     */
    public String getColor() {
        return color;
    }

    /**
     * Метод для встановлення коліра плоду.
     *
     * @param color Колір плоду.
     */
    public void setColor(String color) {
        this.color = color;
    }

    /**
     * Метод, який повертає стиглість плоду.
     *
     * @return Стиглість плоду.
     */
    public String getRipeness() {
        return ripeness;
    }

    /**
     * Метод для встановлення стиглістю плоду.
     *
```

```
     * @param ripeness Стиглість плоду.
     */
    public void setRipeness(String ripeness) {
        this.ripeness = ripeness;
    }

    /**
     * Метод для отримання та виведення статусу плоду в консоль і у файл.
     *
     * @param fout Об'єкт PrintWriter для запису в файл.
     */
    public void getStatus(PrintWriter fout) {
        if (this.existence) {
            out.println("Плоди: Є");
            out.println("Колір: " + this.color);
            out.println("Стиглість: " + this.ripeness);

            fout.println("Плоди: Є");
            fout.println("Колір: " + this.color);
            fout.println("Стиглість: " + this.ripeness);
        } else {
            out.println("У рослини немає плодів.");

            fout.println("У рослини немає плодів.");
        }
    }
}
```

Leaf.java

```
package KI305.Zastavnyi.Lab3;

import java.io.PrintWriter;

import static java.lang.System.out;
/**
 * Клас Leaf представляє характеристики листка рослини.
 * Містить інформацію про форму, стан та колір листка.
 */
public class Leaf {
    // Властивості листка
    private String shape;
    private boolean state;
    private String color;

    /**
     * Конструктор за замовчуванням ініціалізує листок значеннями за замовчуванням.
     */
    public Leaf() {
        this.shape = "Немає";
        this.state = false;
        this.color = "Немає";
    }

    /**
     * Конструктор з вказаним станом ініціалізує листок формою та кольором за замовчу-
ванням,
     * але з вказаним станом.
     *
     * @param state Стан листка.
     */
    public Leaf(boolean state) {
```

```java
        this.shape = "Немає";
        this.state = state;
        this.color = "Немає";
    }

    /**
     * Конструктор із вказаною формою, станом та кольором листка.
     *
     * @param shape Форма листка.
     * @param state Стан листка.
     * @param color Колір листка.
     */
    public Leaf(String shape, boolean state, String color) {
        this.shape = shape;
        this.state = state;
        this.color = color;
    }

    /**
     * Метод, який повертає форму листка.
     *
     * @return Форма листка.
     */
    public String getShape() {
        return shape;
    }

    /**
     * Метод для встановлення форми листка.
     *
     * @param shape Форма листка.
     */
    public void setShape(String shape) {
        this.shape = shape;
    }

    /**
     * Метод, який повертає стан листка.
     *
     * @return Стан листка.
     */
    public boolean getState() {
        return state;
    }

    /**
     * Метод для встановлення стану листка.
     *
     * @param state Стан листка.
     */
    public void setState(boolean state) {
        this.state = state;
    }

    /**
     * Метод, який повертає колір листка.
     *
     * @return Колір листка.
     */
    public String getColor() {
        return color;
    }

    /**
```

```
    * Метод для встановлення коліра листка.
    *
    * @param color Колір листка.
    */
   public void setColor(String color) {
      this.color = color;
   }

   /**
    * Метод для отримання та виведення статусу листка в консоль і у файл.
    *
    * @param fout Об'єкт PrintWriter для запису в файл.
    */
   public void getStatus(PrintWriter fout) {
      if (state) {
         out.println("Листя: Є");
         fout.println("Листя: Є");
      } else {
         out.println("Листя: Немає");
         fout.println("Листя: Немає");
      }
      out.println("Форма: " + this.shape);
      out.println("Колір: " + this.color);

      fout.println("Форма: " + this.shape);
      fout.println("Колір: " + this.color);
   }
}
```

Plant.java

```
/**
 * KI305.Zastavnyi.Lab2package
 */
package KI305.Zastavnyi.Lab3;
import java.io.*;
import static java.lang.System.out;
/**
 * Клас, що представляє абстрактний об'єкт "Рослина". Реалізує функціонал рослини та
містить абстрактні методи,
       * які повинні бути реалізовані в конкретних рослинах.
       *
       * @author Roman Zastavnyi
      * @version 1.0
      */
public abstract class Plant {
   private String name;           // Ім'я рослини
   private Size size;             // Розмір рослини
   private String color;          // Колір рослини
   private Fruit fruit;           // Плід рослини
   private Leaf leaf;             // Листок рослини
   private double humidity;       // Вологість рослини
   private int countOfLeaves;     // Кількість листків
   private PrintWriter fout;      // Об'єкт для запису у файл

   /**
    * Конструктор за замовчуванням. Ініціалізує всі властивості рослини на значення за
замовчуванням та створює
    * об'єкт PrintWriter для запису у файл "Log.txt".
    *
    * @throws IOException
```

```
    */
    public Plant() throws IOException {
        this.name = "Немає";
        this.size = new Size();
        this.color = "Немає";
        this.fruit = new Fruit();
        this.leaf = new Leaf();
        this.humidity = 0;
        this.countOfLeaves = 0;
        fout = new PrintWriter(new FileWriter("Log.txt", true));
    }

    /**
     * Конструктор, що дозволяє встановити ім'я, колір та кількість листків рослини при створенні об'єкта.
     *
     * @param _name         Ім'я рослини
     * @param _color        Колір рослини
     * @param countOfLeaves  Кількість листків
     * @throws IOException
     */
    public Plant(String _name, String _color, int countOfLeaves) throws IOException {
        this.name = _name;
        this.size = new Size();
        this.color = _color;
        this.fruit = new Fruit();
        this.leaf = new Leaf(true);
        this.humidity = 0;
        this.countOfLeaves = countOfLeaves;
        fout = new PrintWriter(new FileWriter("Log.txt", true));
    }

    /**
     * Конструктор, що дозволяє встановити всі властивості рослини при створенні об'єкта.
     *
     * @param name          Ім'я рослини
     * @param size          Розмір рослини
     * @param color         Колір рослини
     * @param fruit         Плід рослини
     * @param leaf          Листок рослини
     * @param humidity      Вологість рослини
     * @param countOfLeaves  Кількість листків
     * @throws IOException
     */
    public Plant(String name, Size size, String color, Fruit fruit, Leaf leaf, double humidity, int countOfLeaves) throws IOException {
        this.name = name;
        this.size = size;
        this.color = color;
        this.fruit = fruit;
        this.leaf = leaf;
        this.humidity = humidity;
        this.countOfLeaves = countOfLeaves;
        fout = new PrintWriter(new FileWriter("Log.txt", true));
    }

    /**
     * Метод, який повертає PrintWriter об'єкт для запису у файл.
     *
     * @return PrintWriter об'єкт
     */
    public PrintWriter getFout()
    {
        return fout;
```

```java
}

/**
 * Метод, що повертає розмір рослини.
 *
 * @return Розмір рослини
 */
public Size getSize() {
    return size;
}

/**
 * Метод, що встановлює розмір рослини.
 *
 * @param size Розмір рослини
 */
public void setSize(Size size) {
    this.size = size;
}

/**
 * Метод, що повертає плід рослини.
 *
 * @return Плід рослини
 */
public Fruit getFruit() {
    return fruit;
}

/**
 * Метод, що встановлює плід рослини.
 *
 * @param fruit Плід рослини
 */
public void setFruit(Fruit fruit) {
    this.fruit = fruit;
}

/**
 * Метод, що повертає листок рослини.
 *
 * @return Листок рослини
 */
public Leaf getLeaf() {
    return leaf;
}

/**
 * Метод, що встановлює листок рослини.
 *
 * @param leaf Листок рослини
 */
public void setLeaf(Leaf leaf) {
    this.leaf = leaf;
}

/**
 * Метод, що встановлює ім'я рослини.
 *
 * @param name Ім'я рослини
 */
public void setName(String name) {
    this.name = name;
}
```

```java
    /**
     * Метод, що встановлює колір рослини.
     *
     * @param color Колір рослини
     */
    public void setColor(String color) {
        this.color = color;
    }

    /**
     * Метод, що встановлює вологість рослини.
     *
     * @param humidity Вологість рослини
     */
    public void setHumidity(double humidity) {
        this.humidity = humidity;
    }

    /**
     * Метод, що встановлює кількість листків рослини. Якщо передано від'ємне значення, виводить повідомлення про помилку.
     * Якщо передано значення 0, встановлює кількість листків на 0 та встановлює стан листка на true.
     *
     * @param countOfLeaves Кількість листків
     */
    public void setCountOfLeaves(int countOfLeaves) {
        if (countOfLeaves > 0)
            this.countOfLeaves = countOfLeaves;
        else if (countOfLeaves == 0) {
            this.countOfLeaves = countOfLeaves;
            this.leaf.setState(true);
        } else {
            out.println("Вказано неправильне значення кількості листків");
        }
    }

    /**
     * Метод, що повертає ім'я рослини.
     *
     * @return Ім'я рослини
     */
    public String getName() {
        return name;
    }

    /**
     * Метод, що повертає колір рослини.
     *
     * @return Колір рослини
     */
    public String getColor() {
        return color;
    }

    /**
     * Метод, що повертає вологість рослини.
     *
     * @return Вологість рослини
     */
    public double getHumidity() {
        return humidity;
    }
```

```java
    /**
     * Метод, що повертає кількість листків рослини.
     *
     * @return Кількість листків
     */
    public int getCountOfLeaves() {
        return countOfLeaves;
    }

    /**
     * Абстрактний метод, що представляє процес зростання рослини.
     */
    public void GrowUp() {
        if(this.humidity > 0) {
            size.changeSizeToGrow();
            setCountOfLeaves(this.countOfLeaves + 1);
        }
        else {
            out.println("Рослина не може рости: недостатньо вологості.");
        }
    }
    /**
     * Метод, що виводить статус рослини на консоль та у файл за допомогою PrintWriter.
     */
    public void getStatus() {
        out.println("Рослина: " + this.name);
        out.println("Колір: " + this.color);
        out.println("Вологість: " + this.humidity);

        fout.println("Рослина: " + this.name);
        fout.println("Колір: " + this.color);
        fout.println("Вологість: " + this.humidity);

        size.getStatus(fout);

        out.println("Кількість листків: " + this.countOfLeaves);

        fout.println("Кількість листків: " + this.countOfLeaves);

        leaf.getStatus(fout);
        fruit.getStatus(fout);
        out.println();
        fout.append("\n");
    }

    /**
     * Абстрактний метод, що представляє процес фотосинтезу рослини.
     *
     * @param countLight Кількість світла для фотосинтезу
     */
    public void photosynthesis(double countLight) {
        if (countLight > 0 && this.humidity >= countLight * 0.1 && this.humidity >
0&&this.countOfLeaves > 0) {
            this.humidity -= countLight * 0.1;
            out.println("Сфотосинтезовано " + countLight * 1.2 + " кисню.");
        } else if(this.humidity <= 0){
            out.println("Недостатньо вологості для фотосинтезу.");
        }else if(this.countOfLeaves <=0){
            out.println("Недостатньо листя для фотосинтезу.");
        }else {
            out.println("Недостатньо світла для фотосинтезу.");
        }
    }
```

```
    /**
     * Метод, що представляє процес "смерті" рослини, встановлюючи всі властивості рослини на відповідні значення.
     */
    public void Death() {
        size.changeSizeToDeath();
        setColor("Немає");
        setCountOfLeaves(0);
        setHumidity(0);
        finish();
    }

    /**
     * Метод, що викликається при завершенні роботи з рослиною та закриває PrintWriter.
     */
    public void finish() {
        fout.close();
    }
}
```

Size.java

```
package KI305.Zastavnyi.Lab3;

import java.io.PrintWriter;

import static java.lang.System.out;

/**
 * Клас, що представляє розміри об'єкта. Кожен розмір може мати ширину, висоту та довжину.
 * Для спрощення можливості створення об'єктів за замовчуванням та з заданими параметрами є конструктори.
 */
public class Size {
    private double width;   // Ширина
    private double height;  // Висота
    private double length;  // Довжина

    /**
     * Конструктор за замовчуванням. Встановлює всі властивості розміру на значення 0.
     */
    public Size() {
        this.width = 0;
        this.height = 0;
        this.length = 0;
    }

    /**
     * Конструктор, який дозволяє встановити ширину, висоту та довжину розміру при створенні об'єкта.
     *
     * @param width  Ширина
     * @param height Висота
     * @param length Довжина
     */
    public Size(double width, double height, double length) {
        this.width = width;
        this.height = height;
        this.length = length;
    }
```

```java
/**
 * Метод, що повертає ширину розміру.
 *
 * @return Ширина
 */
public double getWidth() {
    return width;
}

/**
 * Метод, що встановлює ширину розміру.
 *
 * @param width Ширина
 */
public void setWidth(double width) {
    this.width = width;
}

/**
 * Метод, що повертає висоту розміру.
 *
 * @return Висота
 */
public double getHeight() {
    return height;
}

/**
 * Метод, що встановлює висоту розміру.
 *
 * @param height Висота
 */
public void setHeight(double height) {
    this.height = height;
}

/**
 * Метод, що повертає довжину розміру.
 *
 * @return Довжина
 */
public double getLength() {
    return length;
}

/**
 * Метод, що встановлює довжину розміру.
 *
 * @param length Довжина
 */
public void setLength(double length) {
    this.length = length;
}

/**
 * Метод, що виводить статус розміру на консоль та у файл за допомогою PrintWriter.
 *
 * @param fout PrintWriter об'єкт для запису у файл
 */
public void getStatus(PrintWriter fout) {
    out.println("Ширина: " + this.width);
    out.println("Довжина: " + this.length);
    out.println("Висота: " + this.height);
```

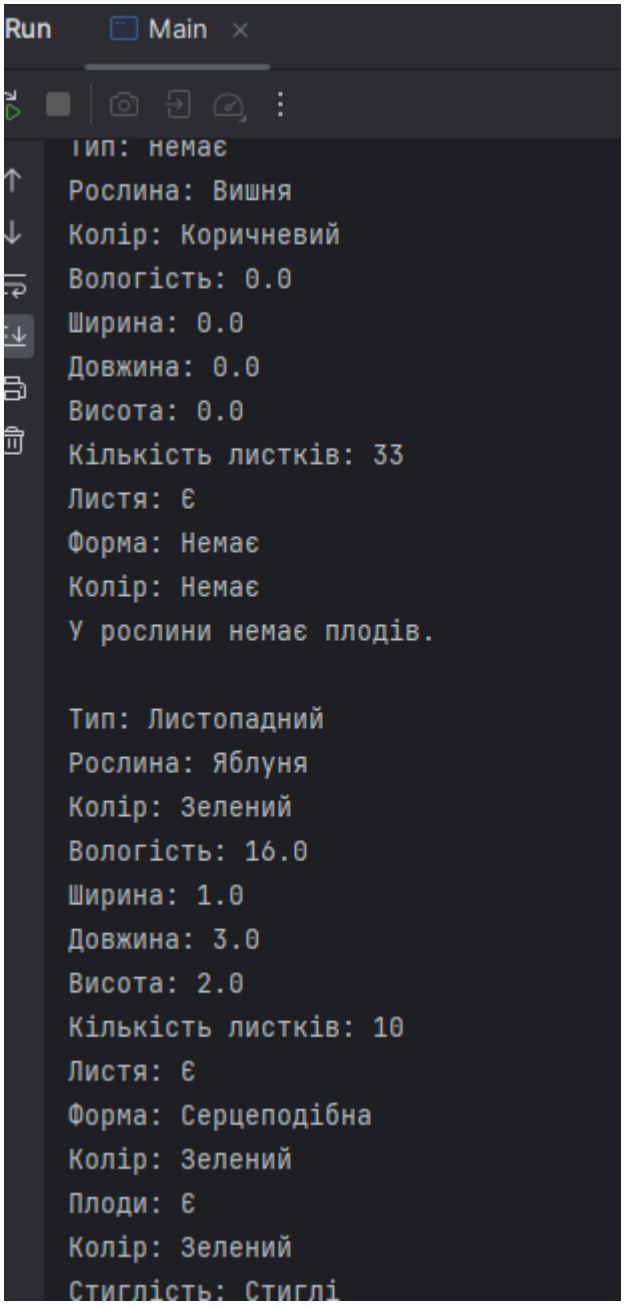
```
        fout.println("Ширина: " + this.width);
        fout.println("Довжина: " + this.length);
        fout.println("Висота: " + this.height);
    }

    /**
     * Метод, що змінює розмір на зростання.
     */
    public void changeSizeToGrow() {
        setWidth(this.width + 1.1);
        setHeight(this.height + 1.1);
        setLength(this.length + 1.1);
    }

    /**
     * Метод, що змінює розмір на "Смерть", встановлюючи всі властивості розміру на значення 0.
     */
    public void changeSizeToDeath() {
        setWidth(0);
        setHeight(0);
        setLength(0);
    }
}
```

Tree.java

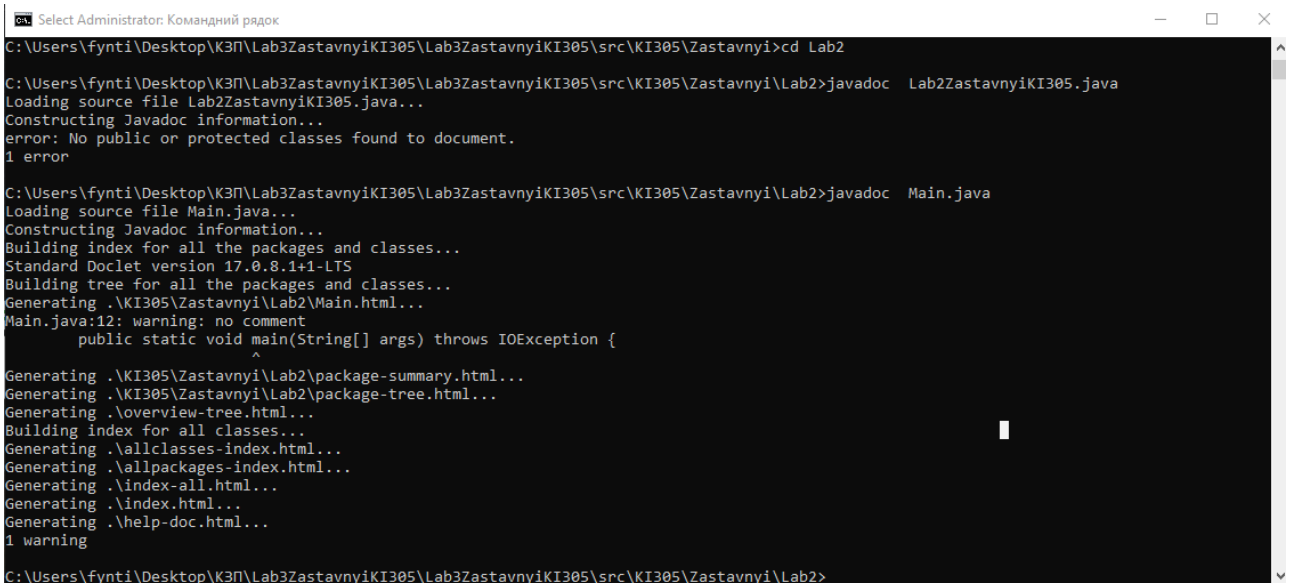
```
package KI305.Zastavnyi.Lab3;

import java.io.IOException;

import static java.lang.System.out;

/**
 * Клас, що представляє об'єкт "Дерево", яке розширює клас "Рослина" та реалізує інтерфейс "GrowUp".
 * Дерево має додаткове поле "тип".
 */
public class Tree extends Plant implements GrowUp {
    private String type;  // Тип дерева

    /**
     * Конструктор за замовчуванням. Викликає конструктор базового класу та ініціалізує тип дерева на "Немає".
     *
     * @throws IOException
     */
    public Tree() throws IOException {
        super();
        this.type = "Немає";
    }

    /**
     * Конструктор, який дозволяє встановити ім'я, колір, кількість листків та тип дерева при створенні об'єкта.
     *
     * @param _name         Ім'я дерева
     * @param _color        Колір дерева
     * @param countOfLeaves  Кількість листків
     * @param type          Тип дерева
     * @throws IOException
     */
```

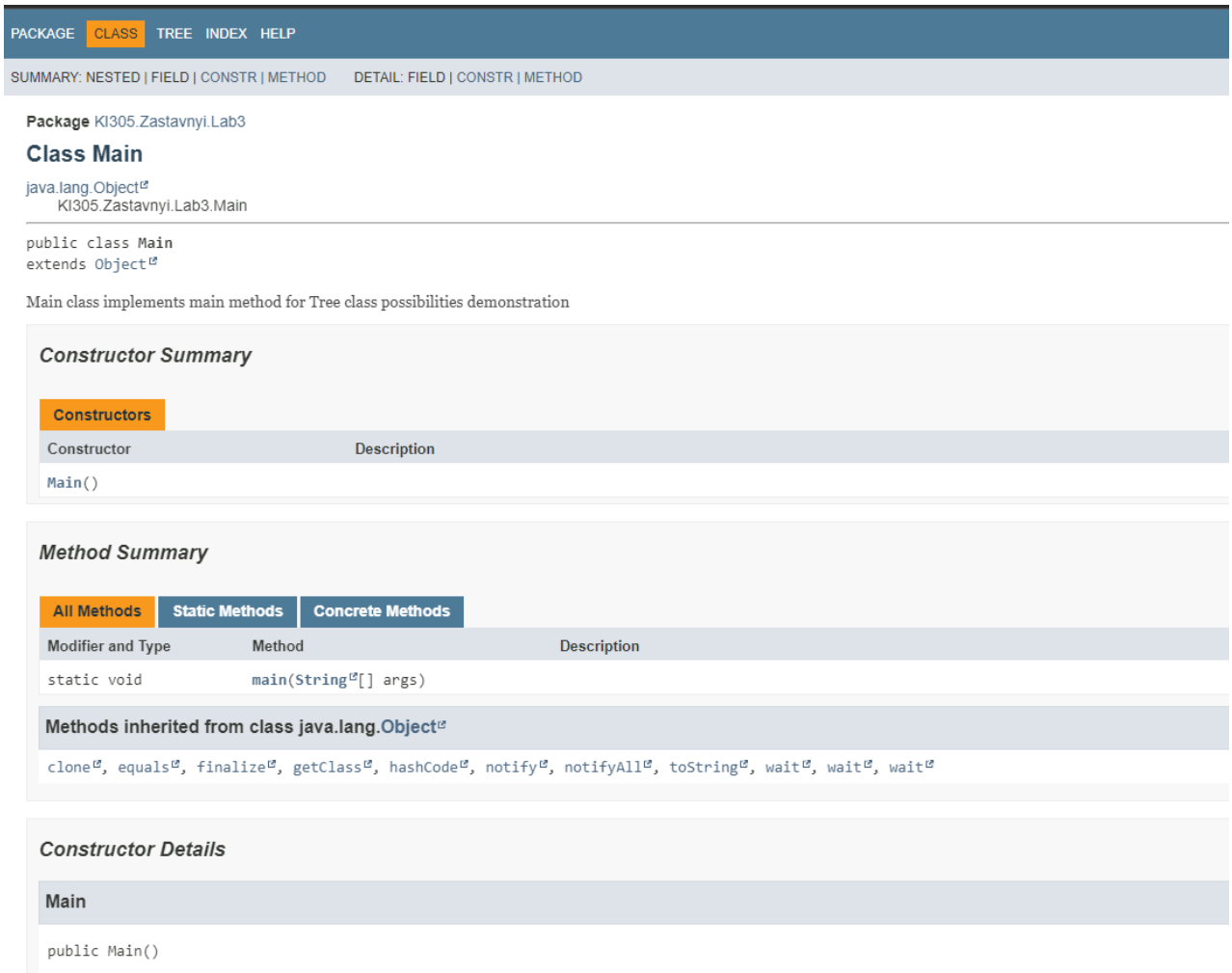
```java
    public Tree(String _name, String _color, int countOfLeaves, String type) throws IOException {
        super(_name, _color, countOfLeaves);
        this.type = type;
    }

    /**
     * Конструктор, який дозволяє встановити всі властивості дерева при створенні об'єкта.
     *
     * @param name          Ім'я дерева
     * @param size          Розмір дерева
     * @param color         Колір дерева
     * @param fruit         Плід дерева
     * @param leaf          Листок дерева
     * @param humidity      Вологість дерева
     * @param countOfLeaves  Кількість листків
     * @param type          Тип дерева
     * @throws IOException
     */
    public Tree(String name, Size size, String color, Fruit fruit, Leaf leaf, double humidity, int countOfLeaves, String type) throws IOException {
        super(name, size, color, fruit, leaf, humidity, countOfLeaves);
        this.type = type;
    }

    /**
     * Метод, який визначає зростання дерева. Змінює розмір та кількість листків за певних умов.
     */
    @Override
    public void Grow() {
        if (this.HumidityToGrow()) {
            this.getSize().changeSizeToGrow();
            setCountOfLeaves(this.getCountOfLeaves() + 1);
        } else {
            out.println("Дерево не може рости: недостатньо вологості.");
        }
    }

    /**
     * Метод, який визначає, чи є достатньо вологості для зростання дерева.
     *
     * @return true, якщо вологість менше 10, інакше false.
     */
    @Override
    public boolean HumidityToGrow() {
        return this.getHumidity() < 10;
    }

    /**
     * Метод, який виводить статус дерева на консоль та у файл за допомогою PrintWriter.
     */
    public void getStatus() {
        super.getStatus();
        out.println("Тип: " + this.type);
        getFout().println("Тип: " + this.type);
    }
}
```

Результата роботи програми:



Фрагмент згенерованої документації





Висновок: Я ознайомився з базовими конструкціями мови Java та оволодів навиками написання й автоматичного документування простих консольних програм мовою Java.