Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



**Звіт**

з лабораторної роботи № 3

**з дисципліни:** «Кросплатформенні засоби програмування»

**на тему:** «Спадкування та інтерфейси»

**Виконала:**

студентка групи *КІ-306*

*Максимчук Катерина*

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

**Львів – 2023**

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

**Завдання ( Варіант 9 «Дерево» ):**

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі No3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі No3, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її

виконання та фрагменту згенерованої документації.

1. Дати відповідь на контрольні запитання.

**Вихідний код програми**

import java.io.FileNotFoundException;  
  
*/\*\*  
 \* Class <code>Flower</code> implements flower  
 \* @author Kateryna Maksymchuk  
 \* @version 1.0  
 \*/*public class Flower {  
 private int size;  
  
 */\*\*  
 \* @param size  
 \*/* public void setSize(int size) {  
 this.size = size;  
 }  
  
 */\*\*  
 \* @return size  
 \*/* public int getSize() {  
 return size;  
 }  
  
 */\*\*  
 \* Constructor  
 \* @param size size of flower  
 \*/* public Flower(int size)  
 {  
 this.size = size;  
 }  
 */\*\*  
 \* Constructor  
 \*/* public Flower(){  
 this.size = 1;  
 }  
  
 */\*\*  
 \* Method prints info about flower  
 \*/* public void Print()  
 {  
 System.*out*.println("My flower is " + size);  
 Plant.*fout*.println("My flower is " + size);  
 Plant.*fout*.flush();  
  
 }  
 */\*\*  
 \* Method makes flower grow using waterpower and sunpower  
 \* @param waterpower amount of waterpower  
 \* @param sunpower amount of sunpower  
 \* @return Flower's size  
 \*/* public int GetWaterSunPower(int waterpower, int sunpower) {  
 setSize((waterpower + sunpower) \* size);  
 Plant.*fout*.println("Flower grows to " + getSize());  
 System.*out*.println("Flower grows to " + getSize());  
 Plant.*fout*.flush();  
  
 return getSize();  
 }  
 */\*\*  
 \* Method that say if flower enough size to reproduce  
 \* @return true or false  
 \*/* public boolean IfCanReproduce(int maxSize){  
 if(size >= maxSize)  
 return true;  
 return false;  
 }  
  
}

import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
  
*/\*\*  
 \* Class <code>Leaf</code> implements leaf of plant  
 \* @author Kateryna Maksymchuk  
 \* @version 1.0  
 \*/*public class Leaf {  
 private int size;  
  
 */\*\*  
 \* @param size  
 \*/* public void setSize(int size) {  
 this.size = size;  
 }  
  
 */\*\*  
 \* @return  
 \*/* public int getSize() {  
 return size;  
 }  
  
 */\*\*  
 \* @param size  
 \*/* //constructors  
 public Leaf(int size)  
 {  
 this.size = size;  
 }  
  
 */\*\*  
 \*  
 \*/* public Leaf(){  
 this.size = 1;  
 }  
  
 */\*\*  
 \*  
 \*/* public void Print()  
 {  
 System.*out*.println("My leaf is " + size);  
 Plant.*fout*.println("My leaf is " + size);  
 Plant.*fout*.flush();  
  
 }  
  
 */\*\*  
 \* @param sunpower  
 \* @return  
 \*/* public int GetSunPower(int sunpower) {  
 setSize(getSize() + sunpower/2);  
 System.*out*.println("Leaf send " + (sunpower/2) + " energy to flower");  
 Plant.*fout*.println("Leaf send " + (sunpower/2) + " energy to flower");  
 Plant.*fout*.flush();  
  
 return sunpower/2;  
 }  
  
 */\*\*  
 \* @return  
 \*/* public boolean IfCanDry(int maxSize){  
 if(size > maxSize)  
 return true;  
 return false;  
 }  
  
}

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.io.PrintWriter;  
  
*/\*\*  
 \*  
 \*/*public class Stem {  
 */\*\*  
 \*  
 \*/* private int size;  
  
 */\*\*  
 \* @param size  
 \*/* public void setSize(int size) {  
 this.size = size;  
 }  
  
 */\*\*  
 \* @return  
 \*/* public int getSize() {  
 return size;  
 }  
  
 */\*\*  
 \* @param size  
 \*/* //constructors  
 public Stem(int size)  
 {  
 this.size = size;  
 }  
  
 */\*\*  
 \*  
 \*/* public Stem(){  
  
 this.size = 2;  
 }  
  
 */\*\*  
 \*  
 \*/* public void Print()  
 {  
 System.*out*.println("My stem is " + size);  
 Plant.*fout*.println("My stem is " + size);  
 Plant.*fout*.flush();  
  
 }  
  
 */\*\*  
 \* @param waterpower  
 \* @return  
 \*/* public int GetWaterPower(int waterpower) {  
 setSize(getSize() + waterpower/2);  
 System.*out*.println("Stem send " + (waterpower/2) + " energy to flower");  
 Plant.*fout*.println("Stem send " + (waterpower/2) + " energy to flower");  
 Plant.*fout*.flush();  
  
 return waterpower/2;  
 }  
}

public class Romashka extends Plant implements PrintInterface{  
 private static final int *maxSizeFlowerRep* = 100;  
 private static final int *maxSizeLeafDry* = 10;  
  
 public Romashka(){  
 super();  
 }  
 public Romashka(int sizeLeaf, int sizeFlower, int sizeStem) {  
 super(sizeLeaf, sizeFlower, sizeStem);  
 }  
  
 public void Grow(int waterpower, int sunpower){  
 stem.GetWaterPower(waterpower);  
 leaf.GetSunPower(sunpower);  
  
 flower.GetWaterSunPower(stem.GetWaterPower(waterpower), leaf.GetSunPower(sunpower));  
 System.*out*.println("Romashka size is " + (flower.getSize() + leaf.getSize() + stem.getSize()));  
 *fout*.println("Romashka size is " + (flower.getSize() + leaf.getSize() + stem.getSize()));  
 *fout*.flush();  
 }  
 public void IfFlowerCanReproduce(){  
 if(flower.IfCanReproduce(*maxSizeFlowerRep*)) {  
 System.*out*.println("Romashka can reproduce");  
 *fout*.println("Romashka can reproduce");  
 }  
 else {  
 System.*out*.println("Romashka can`t reproduce");  
 *fout*.println("Romashka can't reproduce");  
 }  
 *fout*.flush();  
  
 }  
  
 public void IfLeafCanDry() {  
 if (leaf.IfCanDry(*maxSizeLeafDry*)) {  
 System.*out*.println("Romashka can dry");  
 Plant.*fout*.println("Romashka can dry");  
 } else {  
 System.*out*.println("Romashka can`t dry");  
 *fout*.println("Romashka can't dry");  
 }  
 *fout*.flush();  
  
 }  
 @Override  
 public void Print(){  
 System.*out*.println("This is Romashka");  
 Plant.*fout*.println("This is Romashka");  
 leaf.Print();  
 flower.Print();  
 stem.Print();  
 }  
}

public class Muhoboika extends Plant implements PrintInterface{  
 private static final int *maxSizeFlowerRep* = 140;  
 private static final int *maxSizeLeafDry* = 70;  
  
 public Muhoboika(){  
 super();  
 }  
 public Muhoboika(int sizeLeaf, int sizeFlower, int sizeStem) {  
 super(sizeLeaf, sizeFlower, sizeStem);  
 }  
  
 public void Grow(int waterpower, int sunpower){  
 stem.GetWaterPower(waterpower);  
 leaf.GetSunPower(sunpower);  
  
 flower.GetWaterSunPower(stem.GetWaterPower(waterpower), leaf.GetSunPower(sunpower));  
 System.*out*.println("Muhoboika size is " + (flower.getSize() + leaf.getSize() + stem.getSize()));  
 *fout*.println("Muhoboika size is " + (flower.getSize() + leaf.getSize() + stem.getSize()));  
 *fout*.flush();  
 }  
 public void IfFlowerCanReproduce(){  
 if(flower.IfCanReproduce(*maxSizeFlowerRep*)) {  
 System.*out*.println("Muhoboika can reproduce");  
 *fout*.println("Muhoboika can reproduce");  
 }  
 else {  
 System.*out*.println("Muhoboika can`t reproduce");  
 *fout*.println("Muhoboika can't reproduce");  
 }  
 *fout*.flush();  
  
 }  
  
 public void IfLeafCanDry() {  
 if (leaf.IfCanDry(*maxSizeLeafDry*)) {  
 System.*out*.println("Muhoboika can dry");  
 Plant.*fout*.println("Muhoboika can dry");  
 } else {  
 System.*out*.println("Muhoboika can`t dry");  
 *fout*.println("Muhoboika can't dry");  
 }  
 *fout*.flush();  
  
 }  
 @Override  
 public void Print(){  
 System.*out*.println("This is Muhoboika");  
 Plant.*fout*.println("This is Muhoboika");  
 leaf.Print();  
 flower.Print();  
 stem.Print();  
 }  
}

import java.io.File;  
import java.io.IOException;  
import java.io.PrintWriter;  
  
*/\*\*  
 \*  
 \*/*public abstract class Plant implements AutoCloseable {  
 protected Leaf leaf;  
 protected Flower flower;  
 protected Stem stem;  
 public static PrintWriter *fout*;  
  
  
 */\*\*  
 \*  
 \*/* public Plant() {  
 leaf = new Leaf();  
 flower = new Flower();  
 stem = new Stem();  
 try {  
 *fout* = new PrintWriter(new File("PlantTxt.txt"));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* @param sizeLeaf  
 \* @param sizeFlower  
 \* @param sizeStem  
 \*/* public Plant(int sizeLeaf, int sizeFlower, int sizeStem) {  
 try {  
 *fout* = new PrintWriter(new File("PlantTxt.txt"));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 leaf = new Leaf(sizeLeaf);  
 flower = new Flower(sizeFlower);  
 stem = new Stem(sizeStem);  
 }  
  
 */\*\*  
 \* @return  
 \*/* public Flower getFlower() {  
 return flower;  
 }  
  
 */\*\*  
 \* @param flower  
 \*/* public void setFlower(Flower flower) {  
 this.flower = flower;  
 }  
  
 */\*\*  
 \* @return  
 \*/* public Stem getStem() {  
 return stem;  
 }  
  
 */\*\*  
 \* @param stem  
 \*/* public void setStem(Stem stem) {  
 this.stem = stem;  
 }  
  
 */\*\*  
 \* @return  
 \*/* public Leaf getLeaf() {  
 return leaf;  
 }  
  
 */\*\*  
 \* @param leaf  
 \*/* public void setLeaf(Leaf leaf) {  
 this.leaf = leaf;  
 }  
  
 */\*\*  
 \*  
 \*/*// public void Print() {  
// leaf.Print();  
// flower.Print();  
// stem.Print();  
// }  
  
 */\*\*  
 \* @param waterpower  
 \* @param sunpower  
 \*/* public abstract void Grow(int waterpower, int sunpower);  
 */\*\*  
 \*  
 \*/* public abstract void IfFlowerCanReproduce();  
  
 */\*\*  
 \*  
 \*/* public abstract void IfLeafCanDry();  
  
 */\*\*  
 \* Method closes the file  
 \*  
 \* @throws Exception  
 \*/* @Override  
 public void close() throws Exception {  
 if (*fout* != null) {  
 *fout*.flush();  
 *fout*.close();  
 }  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 Muhoboika Voloda = new Muhoboika(3, 5,8);  
 Romashka Katya = new Romashka(10, 8, 9);  
 Voloda.Grow(6,8);  
 Katya.Grow(5,4);  
 System.*out*.println("Implemented interfaces");  
 Katya.Print();  
 Voloda.Print();  
 System.*out*.println("Katya & Voloda method");  
 Katya.IfFlowerCanReproduce();  
 Katya.IfLeafCanDry();  
 Voloda.IfFlowerCanReproduce();  
 Voloda.IfLeafCanDry();  
 }  
}

**Результат виконання програми**

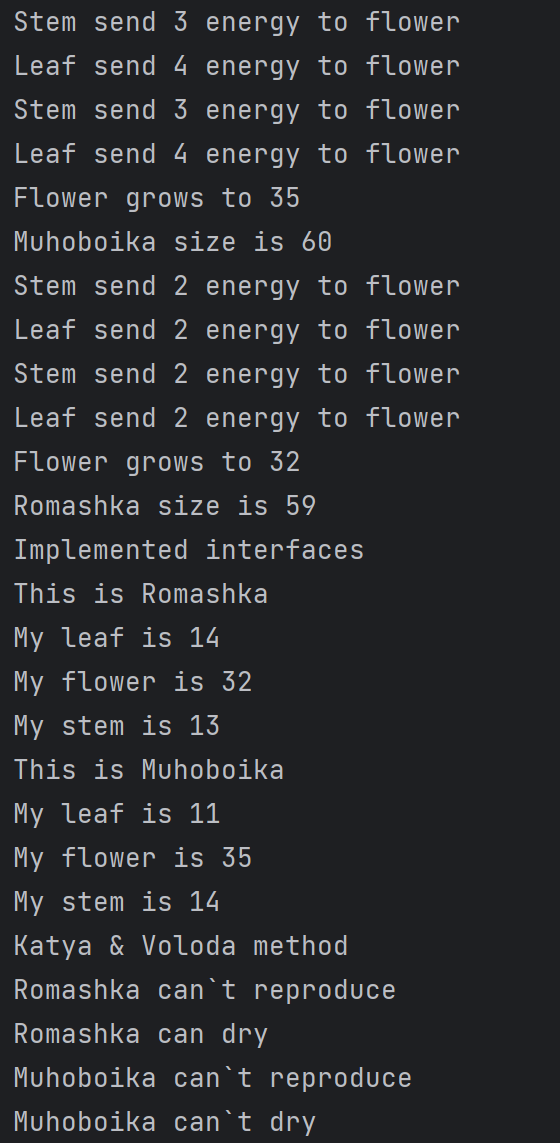
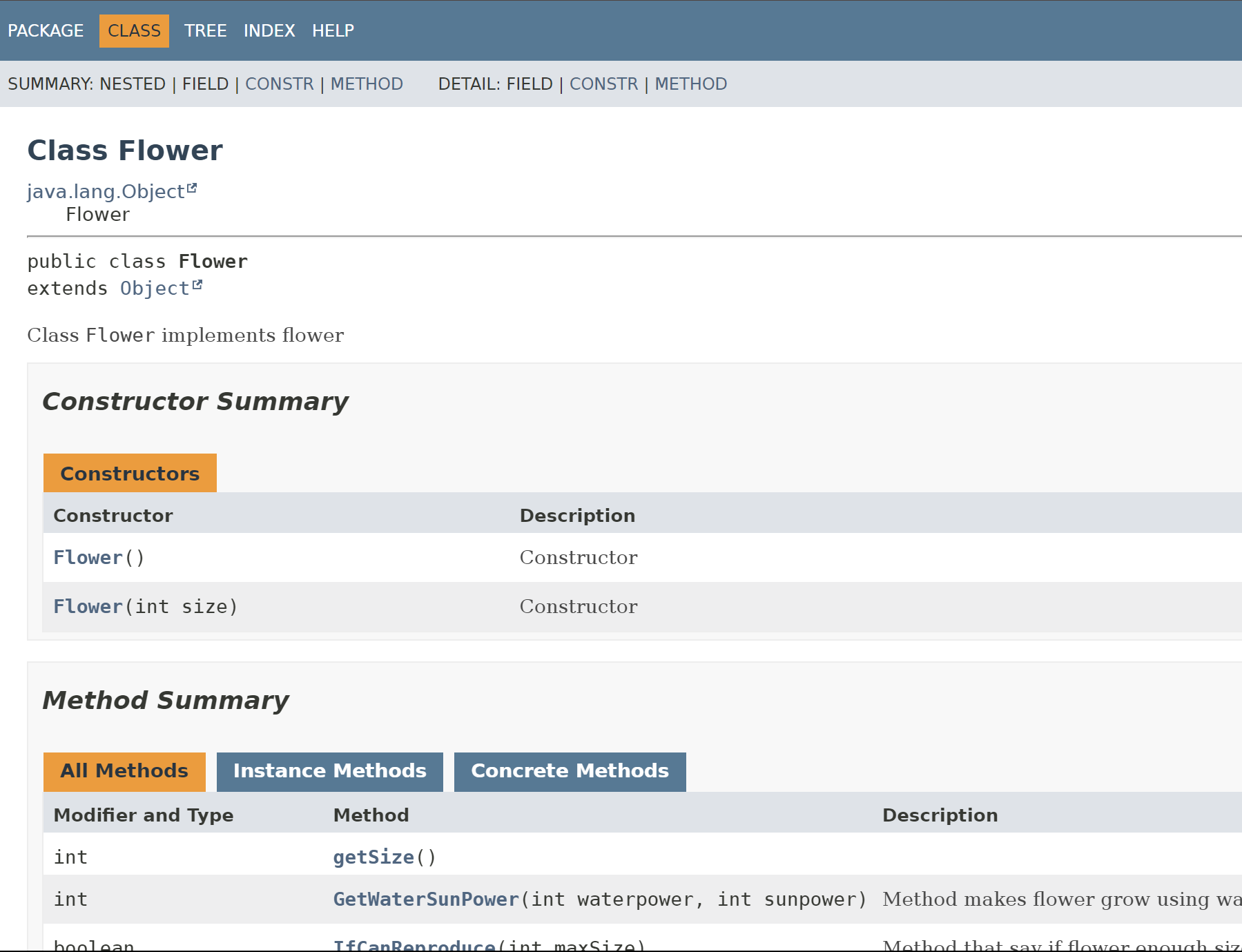
****

Рис.1 – Скріншот документу згенерованого програмою

**Фрагмент згенерованої документації**



**Відповіді на контрольні запитання**

1. Синтаксис реалізації спадкування.

class ChildClass extends ParentClass {

// Код підкласу тут

}

1. Що таке суперклас та підклас?

Суперклас (або батьківський клас) - це клас, від якого успадковуються властивості та методи.

Підклас (або дочірній клас) - це клас, який успадковує властивості та методи від суперкласу і може додавати власні.

1. Як звернутися до членів суперкласу з підкласу?

За допомогою ключового слова super. Наприклад: super.methodName(); для виклику методу суперкласу.

1. Коли використовується статичне зв’язування при виклику методу?

Статичне зв'язування відбувається під час компіляції, коли відомо, який метод викликати на основі типу посилання.

1. Як відбувається динамічне зв’язування при виклику методу?

Динамічне зв'язування відбувається під час виконання програми і визначається на основі об'єкта, який викликає метод.

1. Що таке абстрактний клас та як його реалізувати?

Абстрактний клас - це клас, який не може бути інстанційований і може містити абстрактні методи.

Для створення абстрактного класу використовується ключове слово abstract.

1. Для чого використовується ключове слово instanceof?

Використовується для перевірки, чи об'єкт належить до певного класу або інтерфейсу.

1. Як перевірити чи клас є підкласом іншого класу?

Використовується ключове слово extends. Наприклад, class ChildClass extends ParentClass.

1. Що таке інтерфейс?

Інтерфейс - це контракт, який описує методи, які клас повинен реалізувати. В інтерфейсах методи не мають реалізації.

1. Як оголосити та застосувати інтерфейс?

Оголошення інтерфейсу: interface InterfaceName { ... }

Застосування інтерфейсу в класі: class ClassName implements InterfaceName { ... }

**Висновок**

Ознайомилася з спадкуванням та інтерфейсами у мові Java. Розробила розширений клас Tree за областю з варіанту «Дерево».