Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп'ютерних технологій, автоматики та метрології

Кафедра ЕОМ

Звіт
До лабораторної роботи №4
З дисципліни: «Кросплатформні засоби програмування»
На тему «Спадкування та інтерфейси»
Варіант №9

Виконав: ст. гр. КІ-36
Нагребний Ю.С.

Прийняв:
Іванов Ю.С.

Львів – 2022

Мета: 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 №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, 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.
4. Дати відповідь на контрольні запитання.

9. Дерево

Виконання:

Лістинг програми:

```
//Main.java
package KI36.Nahrebnyi.Lab4;

/**
 * Main class
 *
 * @author Yura
 * @version 1.0
 */
public class Main
{
    /**
     * Main method
     * @param args
     */
    public static void main(String[] args)
    {
        Tree tree = new Tree("Tree", 1, 19.2);

        tree.GrowUp();
        System.out.println("-----------------------------------------");
        tree.PrintInfo();
        System.out.println("-----------------------------------------");
        tree.Wither();
        System.out.println("-----------------------------------------");
        tree.PrintInfo();
    }
}

//Leaf.java
package KI36.Nahrebnyi.Lab3;

import java.util.Random;
```

```java
/**
 * Class Leaf
 * @version 1.0
 * @author Yura
 */
public class Leaf
{
    private String color;
    private double length;
    private Logger logger = Logger.getLogger();

    /**
     * Constructor
     * @param color
     */
    public Leaf(String color)
    {
        logger.log(logger.infoFlag + "Leaf constructor was called");
        this.color = color;
        Random random = new Random();
        this.length = random.nextDouble(10);
    }

    /**
     * Getter for color
     * @return color
     */
    public String getColor()
    {
        logger.log(logger.infoFlag + "Leaf getColor was called");
        return color;
    }

    /**
     * Setter for color
     * @param color
     */
    public void setColor(String color)
    {
        logger.log(logger.infoFlag + "Leaf setColor was called");
        this.color = color;
    }

    /**
     * Getter for length
     * @return length
     */
    public double getLength()
    {
        logger.log(logger.infoFlag + "Leaf setLength was called");
        return length;
    }

    /**
     * Setter for length
     * @param length
     */
    public void setLength(double length)
```

```
    {
        logger.log(logger.infoFlag + "Leaf getColor was called");
        this.length = length;
    }

    /**
     * Method Wither
     */
    void Wither()
    {
        logger.log(logger.infoFlag + "Leaf Wither method was called");
        this.color = "Yellow";
        System.out.println("Autumn came and the leaves withered");
    }

    /**
     * Method GrowUp
     */
    void GrowUp()
    {
        logger.log(logger.infoFlag + "Leaf GrowUp method was called");
        Random random = new Random();
        double grow = random.nextDouble(10);
        this.length +=  grow;

        System.out.println("Leaf grow up " + grow + "s.");
    }

    /**
     * Method PrintInfo
     */
    void PrintInfo()
    {
        logger.log(logger.infoFlag + "Leaf PrintInfo was called");
        System.out.println("Leaf: { length: " + length +" s.; color: " + color +
" }");
    }
}

//Branch.java
package KI36.Nahrebnyi.Lab3;

import java.util.Random;

/**
 * Lab3. Class Branch
 *
 * @author Yura
 * @version 1.0
 */
public class Branch
{
    private double length;
    Logger logger = Logger.getLogger();

    /**
     * Constructor
     */
    public Branch()
```

```
    {
        logger.log(logger.infoFlag + "Branch constructor was called");
        Random random = new Random();
        this.length = random.nextDouble(10);
    }

    /**
     * Getter for length
     * @return
     */
    public double getLength()
    {
        logger.log(logger.infoFlag + "Branch getLength was called");
        return length;
    }

    /**
     * Setter for length
     * @param length
     */
    public void setLength(double length)
    {
        logger.log(logger.infoFlag + "Branch setLength was called");
        this.length = length;
    }

    /**
     * Method GrowUp
     */
    void GrowUp()
    {
        logger.log(logger.infoFlag + "Branch GrowUp method was called");
        Random random = new Random();
        double grow = random.nextDouble(10);
        this.length +=  grow;

        System.out.println("Branch grow up " + grow + "s.");
    }

    /**
     * Method to print info
     */
    void PrintInfo()
    {
        logger.log(logger.infoFlag + "Branch PrintInfo method was called");
        System.out.println("Branch { length: " + length + "s. }" );
    }
}

//Logger.java
package KI36.Nahrebnyi.Lab3;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

/**
```

```java
 * Class Logger. Was created to log information, errors and warnings. Also there
was implemented Singelton
 * @author Yura
 * @version 1.0
 */
public class Logger
{
    private static Logger logger;
    private final String fileName;

    protected final String infoFlag = new String("[INFO] ");

    /**
     * Constructor
     * @param fileName
     */
    private Logger(String fileName)
    {
        this.fileName = fileName;
        File loggerFile = null;
        FileWriter fout = null;
        try
        {
            loggerFile = new File(fileName);
            fout = new FileWriter(loggerFile, true);
            SimpleDateFormat formatter= new SimpleDateFormat("yyyy-MM-dd 'at'
HH:mm:ss z");
            Date date = new Date(System.currentTimeMillis());
            fout.write("[" + formatter.format(date) + "] " + "Logger start to
work\n");
        }
        catch (IOException e)
        {
            System.err.println("Something wrong with log file" +
e.getMessage());
            System.exit(1);
        }
        finally
        {
            try
            {
                fout.flush();
                fout.close();
            }
            catch (IOException e)
            {
                System.out.println(e.getMessage());
            }
        }
    }

    /**
     * Method to do logging
     * @param massege
     */
    public void log(String massege)
    {
        File loggerFile = null;
        FileWriter fout = null;
```

```
        try
        {
            loggerFile = new File(this.fileName);
            fout = new FileWriter(loggerFile, true);
            SimpleDateFormat formatter= new SimpleDateFormat("yyyy-MM-dd 'at'
HH:mm:ss z");
            Date date = new Date(System.currentTimeMillis());
            fout.write("[" + formatter.format(date) + "] " + massege +  "\n");
        }
        catch (IOException e)
        {
            System.err.println("Something wrong with log file" +
e.getMessage());
            System.exit(1);
        }
        finally
        {
            try
            {
                fout.flush();
                fout.close();
            }
            catch (IOException | NullPointerException e)
            {
                System.out.println(e.getMessage());
            }
        }
    }

    /**
     * Singleton implementation
     * @param fileName
     * @return
     */
    public static Logger getLogger(String fileName)
    {
        if (logger == null)
        {
            logger = new Logger(fileName);
        }
        return logger;
    }

    /**
     * Getter for logger
     * @return logger
     */
    public static Logger getLogger()
    {
        return logger;
    }

}

//Plant.java
package KI36.Nahrebnyi.Lab4;

/**
 * Lab 3. Class Plant
```

```java
 */
public abstract class Plant
{
    protected String name;
    protected int age;
    protected double length;
    protected Leaf[] leaves = null;
    protected Branch[] branches = null;
    protected Logger logger = Logger.getLogger("logs.txt");

    /**
     * Constructor
     * @param name
     * @param age
     * @param length
     */
    public Plant(String name, int age, double length)
    {
        logger.log(logger.infoFlag + "Plant constructor was called");
        this.name = name;
        this.age = age;
        this.length = length;

        leaves = new Leaf[age * 6];
        for (int i = 0; i < age * 6; i++)
        {
            leaves[i] = new Leaf("green");
        }

        branches = new Branch[age * 3];
        for (int i = 0 ; i< age * 3; i++)
        {
            branches[i] = new Branch();
        }
    }

    /**
     * Getter for name
     * @return name
     */
    public String getName()
    {
        logger.log(logger.infoFlag + "Plant getName was called");
        return name;
    }

    /**
     * Setter for name
     * @param name
     */
    public void setName(String name)
    {
        logger.log(logger.infoFlag + "Plant setName was called");
        this.name = name;
    }

    /**
     * Getter for Age
     * @return age
```

```
     */
    public int getAge()
    {
        logger.log(logger.infoFlag + "Plant getAge was called");
        return age;
    }

    /**
     * Setter for Age
     * @param age
     */
    public void setAge(int age)
    {
        logger.log(logger.infoFlag + "Plant setAge was called");
        this.age = age;
    }

    /**
     * Getter for leaves
     * @return leaves
     */
    public Leaf[] getLeaves()
    {
        logger.log(logger.infoFlag + "Plant getLeaves was called");
        return leaves;
    }

    /**
     * Setter for leaves
     * @param leaves
     */
    public void setLeaves(Leaf[] leaves)
    {
        logger.log(logger.infoFlag + "Plant setLeaves was called");
        this.leaves = leaves;
    }

    /**
     * Getter for branches
     * @return branches
     */
    public Branch[] getBranches()
    {
        logger.log(logger.infoFlag + "Plant getBranches was called");
        return branches;
    }

    /**
     * Setter for branches
     * @param branches
     */
    public void setBranches(Branch[] branches)
    {
        logger.log(logger.infoFlag + "Plant setBranches was called");
        this.branches = branches;
    }

    /**
     * Getter for length
```

```
     * @return length
     */
    public double getLength()
    {
        logger.log(logger.infoFlag + "Plant getLength was called");
        return length;
    }

    /**
     * Setter for length
     * @param length
     */
    public void setLength(double length)
    {
        logger.log(logger.infoFlag + "Plant setLength was called");
        this.length = length;
    }

    /**
     * Method to grow up
     */
    public abstract void GrowUp();

    /**
     * Method to print info
     */
    public abstract void PrintInfo();

    /**
     * Method to Wither
     */
    public abstract void Wither();

}

//DropLeaves.java
package KI36.Nahrebnyi.Lab4;

public interface DropLeaves
{
    void ToDropLeaves();
}

//Tree.java
package KI36.Nahrebnyi.Lab4;

public class Tree extends Plant implements DropLeaves
{
    /**
     * Constructor
     *
     * @param name
     * @param age
     * @param length
     */
    public Tree(String name, int age, double length)
    {
        super(name, age, length);
        logger.log(logger.infoFlag + "Tree constructor was called");
```

```
    }

    @Override
    public void ToDropLeaves()
    {
        logger.log(logger.infoFlag + "Tree ToDropLeaves method was called");
        for (int i = 0; i < getAge() * 6; i++)
        {
            leaves[i] = null;
        }
    }

    @Override
    public void GrowUp()
    {
        logger.log(logger.infoFlag + "Tree GrowUp method was called was
called");
        for (int i = 0; i < age*3; i++)
        {
            branches[i].GrowUp();
        }
        for (int i = 0; i < age*6; i++)
        {
            leaves[i].GrowUp();
        }
    }

    @Override
    public void PrintInfo()
    {
        logger.log(logger.infoFlag + "Tree PrintInfo method was called");
        System.out.println("Plant: { name: " + name + "; length: " + length +
"s.; age: " + age +" }");
        for (int i = 0; i < age*3; i++)
        {
            branches[i].PrintInfo();
        }
        for (int i = 0; i < age*6; i++)
        {
            leaves[i].PrintInfo();
        }
    }

    @Override
    public void Wither()
    {
        logger.log(logger.infoFlag + "Plant Wither was called");
        for (int i = 0; i < age * 6; i++)
        {
            leaves[i].Wither();
        }

    }
}
```

Результати:

```
Plant: { name: Tree; length: 19.2s.; age: 1 }
Branch { length: 9.61969373231711s. }
Branch { length: 6.4585200865544766s. }
Branch { length: 10.006204069663966s. }
Leaf: { length: 10.809410290598166 s.; color: green }
Leaf: { length: 4.184390028011466 s.; color: green }
Leaf: { length: 16.768245851948414 s.; color: green }
Leaf: { length: 1.903345698116372 s.; color: green }
Leaf: { length: 8.946492618338183 s.; color: green }
Leaf: { length: 14.12357427297116 s.; color: green }
-------------------------------------------
Autumn came and the leaves withered
Autumn came and the leaves withered
Autumn came and the leaves withered
Autumn came and the leaves withered
Autumn came and the leaves withered
Autumn came and the leaves withered
-------------------------------------------
Plant: { name: Tree; length: 19.2s.; age: 1 }
Branch { length: 9.61969373231711s. }
Branch { length: 6.4585200865544766s. }
Branch { length: 10.006204069663966s. }
Leaf: { length: 10.809410290598166 s.; color: Yellow }
Leaf: { length: 4.184390028011466 s.; color: Yellow }
Leaf: { length: 16.768245851948414 s.; color: Yellow }
Leaf: { length: 1.903345698116372 s.; color: Yellow }
Leaf: { length: 8.946492618338183 s.; color: Yellow }
Leaf: { length: 14.12357427297116 s.; color: Yellow }
```

Висновок: ознайомився з спадкуванням та інтерфейсами у мові Java.