Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Звіт з лабораторної роботи №4

з дисципліни «Кросплатформні засоби програмування»

на тему: «СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ»

Виконала: ст.групи КІ-36

Гульчевська Є. Л.

Прийняв, та перевірив :

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.

Варіант №28 Енергозберігаюча лампочка.

Хід роботи:

Код програми:

Lab04.java

```
import java.io.*;
//Lab03 class implements main method for Bulb class possibilities demonstration
public class Lab04 {
 public static void main(String[] args) throws FileNotFoundException {
     Bulb bulb = new Bulb();
     bulb.setBulbPosition(5, -5);
     //bulb.moveBulb(5, -5);
     bulb.brightness_Dark();
     bulb.Colour_Warm();
     bulb.dispose();
     ESBulb esbulb = new ESBulb();
     esbulb.Colour_Green();
     esbulb.SetTablePosition();
     esbulb.BulbPosition();
     esbulb.dispose();
 }
}
```

Bulb.java

```
import java.io.*;
/**
 * Class </code>Bulb</code> implements bulb
 * */
public class Bulb extends Lamp{
        private Colour colour;
        private Brightness_Types brightness;
        private Position pos;
        private PrintWriter fout;
        /**
         * Constructor
         * @throws FileNotFoundException
         */
        public Bulb() throws FileNotFoundException
        {
            colour = new Colour();
            brightness = new Brightness_Types();
            pos = new Position();
            fout = new PrintWriter(new File("Log.txt"));
        }
        /**
             * Method implements bulb position offset on (xPos, yPos)
             * @param </code>xPos</code> The X coordinate of the bulb position
             * @param </code>yPos</code> The Y coordinate of the bulb position
             */
        public void moveBulb(int xPos, int yPos)
        {
            pos.setXPosition(pos.getXPosition() + xPos);
            pos.setYPosition(pos.getYPosition() + yPos);
            fout.print("Bulb's X position: " + pos.getXPosition() + "\n");
            fout.print("Bulb's Y position: " + pos.getYPosition() + "\n");
        }
        /**
         * Method sets the new bulb position
         * @param </code>xPos</code> The X coordinate of the bulb position
         * @param </code>yPos</code> The Y coordinate of the bulb position
         */
        public void setBulbPosition(int xPos, int yPos)
        {
            pos.setXPosition(xPos);
            pos.setYPosition(yPos);
        }
        /**
         * Method returns bulb's current X position
         * @return Bulb's current X position
         */
        public int getBulbXPosition()
        {
            return pos.getXPosition();
        }
        /**
         * Method returns bulb's current Y position
         * @return Bulb's current Y position
         */
        public int getBulbYPosition()
        {
            return pos.getYPosition();
        }
        /**
         * Method returns bulb's brightness type
         * @return Bulb's brightness type of
</code>Brightness_Types.Types</code> type
         */
        public void BulbPosition()
```

```
        {
                fout.print("Bulb position is: x = "+ pos.getXPosition()+"; y =
"+pos.getYPosition()+"\n");
                fout.flush();
        }
        /**
         * Method simulates Bright brightness of the bulb
         */
        public void brightness_Bright()
        {
            brightness.setBrightType();
            fout.print("Bulb brightness's is bright\n");
            fout.flush();
        }
        /**
         * Method simulates Dark brightness of the bulb
         */
        public void brightness_Dark()
        {
            brightness.setDarkType();
            fout.print("Bulb brightness's is dark\n");
            fout.flush();
        }
        /**
         * Method simulates bulb's brightness setting in a standart (Medium)
type
         */
        public void reset_brightness()
        {
            brightness.reset_Type_of_light();
            fout.print("Bulb brightness's is medium");
            fout.flush();
        }
        /**
         * Method returns bulb's brightness type
         * @return Bulb's brightness type of
</code>Brightness_Types.Types</code> type
         */
        public Brightness_Types.Types get_brightness_type()
        {
            return brightness.getType();
        }

        /**
         * Method simulates White colour of the bulb
         */
        public void Colour_White()
        {
            colour.setWhite();
            fout.print("Bulb colour is white\n");
            fout.flush();
        }
        /**
         * Method simulates Warm colour of the bulb
         */
        public void Colour_Warm()
        {
            colour.setWarm();
            fout.print("Bulb colour is warm\n");
            fout.flush();
        }
        /**
         * Method simulates Green colour of the bulb
         */
        public void Colour_Green()
        {
```

```
            colour.setGreen();
            fout.print("Bulb colour is green\n");
            fout.flush();
        }
        /**
         * Method simulates Red colour of the bulb
         */
        public void Colour_Red()
        {
            colour.setRed();
            fout.print("Bulb colour is red\n");
            fout.flush();
        }
        /**
         * Method simulates Blue colour of the bulb
         */
        public void Colour_Blue()
        {
            colour.setBlue();
            fout.print("Bulb colour is blue\n");
            fout.flush();
        }
        /**
         * Method simulates Pink colour of the bulb
         */
        public void Colour_Pink()
        {
            colour.setPink();
            fout.print("Bulb colour is pink\n");
            fout.flush();
        }
        /**
         * Method simulates Violet colour of the bulb
         */
        public void Colour_Violet()
        {
            colour.setViolet();
            fout.print("Bulb colour is violet\n");
            fout.flush();
        }
        /**
         * Method simulates bulb's colour setting a standart (Mixed_White_Warm)
type
         */
        public void reset_colour()
        {
            colour.resetColour();
            fout.print("Bulb brightness's is mixed (white and warm)");
            fout.flush();
        }
        /**
         * Method returns bulb's colour type
         * @return Bulb's colour type of </code>Colour.Types</code> type
         */
        public Colour.Types get_colour_type()
        {
            return colour.getType();
        }
        /**
         * Method releases used recourses
         */
        public void dispose()
        {
            fout.close();
        }
}
```

Lamp.java

```
import lab3_bulb.Bulb;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;

public abstract class Lamp extends Object {
        public abstract void moveBulb(int xPos, int yPos);
        public abstract void setBulbPosition(int xPos, int yPos);
        public abstract int getBulbXPosition();
        public abstract int getBulbYPosition();
        public abstract void brightness_Bright();
        public abstract void brightness_Dark();
        public abstract void reset_brightness();
        public abstract Brightness_Types.Types get_brightness_type();
        public abstract void Colour_White();
        public abstract void Colour_Warm();
        public abstract void Colour_Green();
        public abstract void Colour_Red();
        public abstract void Colour_Blue();
        public abstract void Colour_Pink();
        public abstract void Colour_Violet();
        public abstract void reset_colour();
        public abstract Colour.Types get_colour_type();
        public abstract void dispose();
}
```

Position.java

```
package KI_36_Hulchevska_Lab4;

// Class <code>Position</code> implements relative positioning
 //       coordinate system
public class Position {
    // coordinates of the mouse position
    private int x, y;
    /**
     * Constructor
     */
    public Position()
    {
        x = 0;
        y = 0;
    }
    /**
     * Constructor
     * @param </code>xPos</code> The X coordinate value
     * @param </code>yPos</code> The Y coordinate value
     */
    public Position(int xPos, int yPos)
    {
        x = xPos;
        y = yPos;
    }
    /**
     * Method returns the X coordinate value
     * @return The X coordinate value
     */
    public int getXPosition()
    {
        return x;
```

```
    }
    /**
     * Method returns the Y coordinate value
     * @return The Y coordinate value
     */
    public int getYPosition()
    {
        return y;
    }

    /**
     * Method returns coordinates of the position in the <code>obj</code>,
     that is passed into method through method parameter
     * @param </code>obj</code> The object, where coordinates of the current
    position are set
     */
    public void getPosition(Position obj)
    {
        obj.x = x;
        obj.y = y;
    }
    /**
     * Method sets the X coordinate value
     * @param </code>xPos</code> The X coordinate value
     */
    public void setXPosition(int xPos)
    {
        x = xPos;
    }
    /**
     * Method sets the Y coordinate value
     * @param </code>yPos</code> The Y coordinate value
     */
    public void setYPosition(int yPos)
    {
        y = yPos;
    }
}
```

Brightness_Types.java

```
public class Brightness_Types {
    // Type for brigthness
    enum Types {Medium, Bright, Dark};
    // current type of brightness
    private Types type;
    /**
     * Constructor
     */
    public Brightness_Types()
    {
        type = Types.Medium;
    }
    /**
     * Method sets bright type
     */
    public void setBrightType()
    {
        type = Types.Bright;
    }
    /**
     * Method sets medium type
     */
    public void setMediumType()
    {
```

```
        type = Types.Medium;
    }
    /**
     * Method sets dark type
     */
    public void setDarkType()
    {
        type = Types.Dark;
    }
    /**
     * Method resets type of brightness to medium type
     */
    public void reset_Type_of_light()
    {
        setMediumType();
    }
    /**
     * Method returns type of brightness
     * @return Type of brightness of <code>Brightness_Types.Types</code> type
     */
    public Types getType()
    {
        return type;
    }
}
```

Colour.java

```
import java.io.*;
public class Colour {
    // Type for colour
    enum Types {White, Warm, Mixed_White_Warm, Green, Red, Blue, Pink, Violet};
    // current colour type
    private Types type;
    /**
     * Constructor
     */
    public Colour()
    {
        type = Types.Mixed_White_Warm;
    }
    /**
     * Method sets White colour
     */
    public void setWhite()
    {
        type = Types.White;
    }
    /**
     * Method sets Warm colour
     */
    public void setWarm()
    {
        type = Types.Warm;
    }
    /**
     * Method sets Mixed_White_Warm colour
     */
    public void setMixed_White_Warm()
    {
        type = Types.Mixed_White_Warm;
    }
    /**
     * Method sets Green colour
     */
```

```
    public void setGreen()
    {
        type = Types.Green;
    }
    /**
     * Method sets Red colour
     */
    public void setRed()
    {
        type = Types.Red;
    }
    /**
     * Method sets Blue colour
     */
    public void setBlue()
    {
        type = Types.Blue;
    }
    /**
     * Method sets Pink colour
     */
    public void setPink()
    {
        type = Types.Pink;
    }
    /**
     * Method sets Violet colour
     */
    public void setViolet()
    {
        type = Types.Violet;
    }
    /**
     * Method resets colour type to Mixed_White_Warm state
     */
    public void resetColour()
    {
        setMixed_White_Warm();
    }
    /**
     * Method returns scrolling direction
     * @return Scrolling direction of <code>Scroller.Directions</code> type
     */
    public Types getType()
    {
        return type;
    }
}
```

ESBulb.java

```
import java.io.*;

public class ESBulb extends Bulb implements ESBulbInterface{
    public ESBulb() throws FileNotFoundException {
        new Bulb();
    }
    public void SetTablePosition()
    {
        setBulbPosition(26,-7);
    }
}
```

ESBulbInterface.java

```
import java.io.*;
public interface ESBulbInterface {
    void SetTablePosition();
}
```

Результат виконання:

```
1    Bulb colour is green
2    Bulb position is: x = 26; y = -7
3
```

Контрольні питання

1. Синтаксис реалізації спадкування.

class Підклас extends Суперклас {

Додаткові поля і методи }

2. Що таке суперклас та підклас?

Суперклас – це базовий клас, а підклас – це похідний клас від суперкласу.

3. Як звернутися до членів суперкласу з підкласу?

Для звернення до методів чи полів суперкласу з підкласу потрібно використати ключове слово super.

super.назваМетоду([параметри]); // виклик методу суперкласу

super.назваПоля // звертання до поля суперкласу

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

Механізм статичного зв’язування передбачає визначення методу, який необхідно викликати, на етапі компіляції.

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

Поліморфізм реалізується за допомогою механізму динамічного зв’язування, який полягає у тому, що вибір методу, який необхідно викликати, відбувається не на етапі компіляції, а під час виконання програми.

6. Що таке абстрактний клас та як його реалізувати?

Абстрактні класи призначені бути основою для розробки ієрархій класів та не дозволяють створювати об’єкти свого класу. Вони реалізуються за допомогою ключового слова abstract. На відміну від звичайних класів абстрактні класи можуть містити абстрактні методи (а можуть і не містити).

7. Для чого використовується ключове слово instanceof?

Оператор instanceof дозволяє перевірити приналежність об'єкта до зазначеного класу з урахуванням успадкування.

8. Як перевірити чи клас є підкласом іншого класу?

1) Можна перевірити за допомогою ключового слова instanceof.

2) Перевірити за допомогою метода isAssignableFrom(), наприклад:

```
public class Test {
public class A {}
public class B extends A {}
public class C {}
public static void main(String[] args) {
System.out.println("B extends A : " + A.class.isAssignableFrom(B.class));
System.out.println("C extends A : " + A.class.isAssignableFrom(C.class));
}
```

```
}
```

Результат:

```
B extends A : true

C extends A : false
```

9. Що таке інтерфейс?

Інтерфейси вказують що повинен робити клас не вказуючи як саме він це повинен робити. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

10. Як оголосити та застосувати інтерфейс?

Синтаксис оголошення інтерфейсів:

[public] interface НазваІнтерфейсу {

Прототипи методів та оголошення констант інтерфейсу

}

Для застосування інтерфейсу потрібно оголосити за допомогою ключового слова implements, що клас реалізує інтерфейс. Якщо клас реалізує кілька інтерфейсів, то вони перелічуються через кому після ключового слова implements.

Висновок: на даній лабораторній роботі я ознайомилася з спадкуванням та інтерфейсами у мові Java.