

Міністерство освіти і науки України
Національний університет “Львівська політехніка”

Кафедра ЕОМ



Звіт

з лабораторної роботи №4

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Спадкування та інтерфейси»

Виконав: ст.гр. КІ-34

Лис Б. Л.

Прийняв:

викл. каф. ЕОМ

Іванов Ю. С.

Львів 2022

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 №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.

Лістинг програми:

Файл MonitorApp.java

```
/**
 * lab 4 package
 */

package KI34.Lys.Lab4;

import java.io.*;

/**
 * Monitor Application class implements main method for Monitor
 * class possibilities demonstration
 * @author Lys Bohdan
 * @version 1.0
 */

public class MonitorApp {
    /**
     * @param args param
     * @throws FileNotFoundException
     */
    public static void main(String[] args) throws FileNotFoundException {

        TouchScreen object = new TouchScreen(1000000, 6.0, "MSI", 24, "LED",
"1920 x 1080", 144, 100, "16:9");

        object.off_onMonitor(Status.ON);
        object.getSettings();
        object.brightDisplayPlus(25);
        object.brightDisplayMinus(63);

        object.changeMatrix("IPS");
        object.changeHz(75);
        object.changeResolution("1080x720");
        object.changeMonitorFormat("4:3");
        object.getSettings();

        object.screenResource(578);
        object.getResource();

        object.off_onMonitor(Status.OFF);
        object.dispose();
    }
}
```

```
}  
}
```

Файл Monitor.java

```
/**  
 * lab 4 package  
 */  
package KI34.Lys.Lab4;  
  
import java.io.*;  
  
/**  
 * Class <code>Monitor</code> implements monitor  
 * * @author Lys Bohdan  
 * * @version 1.0  
 */  
  
public abstract class Monitor {  
    private double hz;  
    private String name;  
    private String resolution;  
    private double diagonal;  
    private String matrix;  
    private double BrightDisplay;  
    public Status status;  
    public PrintWriter myWrite;  
    private String monitorFormat;  
  
    /**  
     * Constructor  
     * @throws FileNotFoundException param  
     */  
  
    public Monitor() throws FileNotFoundException {  
        matrix = "None";  
        diagonal = 0;  
        name = "None";  
        resolution = "None";  
        hz = 0;  
        BrightDisplay = 0;  
        monitorFormat = "None";  
        myWrite = new PrintWriter("Log.txt");  
    }  
  
    /**  
     * Constructor  
     * @param name <code>name</code> Name of monitor  
     * @param diagonal <code>diagonal</code> Monitor diagonal  
     * @param matrix <code>matrix</code> Monitor matrix  
     * @param resolution <code>resolution</code> Monitor resolution  
     * @param hz <code>hz</code> Monitor refresh rate  
     * @param BrightDisplay <code>BrightDisplay</code> The brightness of the  
monitor display  
     * @param monitorFormat <code>monitorFormat</code> Monitor screen format  
     * @throws FileNotFoundException  
     */  
  
    public Monitor(String name, double diagonal, String matrix, String  
resolution, double hz, double BrightDisplay, String monitorFormat) throws  
FileNotFoundException {  
        this.matrix = matrix;  
        this.diagonal = diagonal;  
        this.name = name;  
        this.resolution = resolution;  
        this.hz = hz;  
        this.BrightDisplay = BrightDisplay;  
        this.monitorFormat = monitorFormat;  
    }  
}
```

```

        myWrite = new PrintWriter("Log.txt");
    }

    /**
     * Method returns monitor's refresh rate
     * @return monitor's refresh rate
     */
    public double getHz() {
        return hz;
    }

    /**
     * Method returns monitor's name
     * @return monitor's name
     */
    public String getName() {
        return name;
    }

    /**
     * Method returns monitor's resolution
     * @return monitor's resolution
     */
    public String getResolution() {
        return resolution;
    }

    /**
     * Method returns monitor's diagonal
     * @return monitor's diagonal
     */
    public double getDiagonal() {
        return diagonal;
    }

    /**
     * Method returns monitor's matrix type
     * @return monitor's matrix type
     */
    public String getMatrix() {
        return matrix;
    }

    /**
     * Method returns monitor's display brightness
     * @return monitor's display brightness
     */
    public double getBrightDisplay() {
        return BrightDisplay;
    }

    /**
     * Method sets the new monitor's refresh rate
     * @param hz <code>hz</code> monitor's refresh rate
     */
    public void setHz(double hz) {
        this.hz = hz;
    }

    /**
     * Method sets the new monitor's name
     * @param name <code>name</code> monitor's name
     */
    public void setName(String name) {
        this.name = name;
    }
}

```

```

/**
 * Method sets the new monitor's resolution and format
 * @param resolution <code>resolution</code> monitor's resolution
 * @param monitorFormat <code>monitorFormat</code> monitor's screen format
 */
public void setResolution(String resolution, String monitorFormat) {
    this.resolution = resolution;
    this.monitorFormat = monitorFormat;
}

/**
 * Method sets the new monitor's diagonal
 * @param diagonal <code>diagonal</code> monitor's diagonal
 */
public void setDiagonal(double diagonal) {
    this.diagonal = diagonal;
}

/**
 * Method sets the new monitor's matrix type
 * @param matrix <code>matrix</code> monitor's matrix type
 */
public void setMatrix(String matrix) {
    this.matrix = matrix;
}

/**
 * Method sets the new monitor's display brightness
 * @param brightDisplay <code>brightDisplay</code> monitor's display
brightness
 */
public void setBrightDisplay(double brightDisplay) {
    BrightDisplay = brightDisplay;
}

/**
 * Method simulates monitor's off/on
 */
public void off_onMonitor(Status comand) {
    if (comand == Status.ON) {
        myWrite.println("Монітор включено.");
        status = Status.ON;
    } else {
        status = Status.OFF;
        myWrite.println("Монітор виключено.");
    }
}

/**
 * Method return monitor's settings which we set
 */
public void getSettings() {
    if (status == Status.ON) {
        System.out.println("Name - " + name);
        System.out.println("Diagonal - " + diagonal);
        System.out.println("Resolution - " + resolution);
        System.out.println("Hz - " + hz);
        System.out.println("Matrix - " + matrix);
        System.out.println("Bright - " + BrightDisplay);
        myWrite.println("Налаштування монітора виведенно.");
    } else {
        System.out.println("Your monitor is off!!! ");
        myWrite.println("Налаштування монітора не вивієденно, та як монітор
виключенно.");
    }
}

```

```

    }

    /**
     * Method simulates increment monitor`s display brightness
     */
    public void brightDisplayPlus(int change) {
        BrightDisplay = BrightDisplay + change;
        myWrite.println("Яскравість монітора збільшена.");
    }

    /**
     * Method simulates decrement monitor`s display brightness
     */
    public void brightDisplayMinus(int change) {
        BrightDisplay = BrightDisplay - change;
        myWrite.println("Яскравість монітора зменшена.");
    }

    /**
     * Method simulates changing monitor`s matrix type
     */
    public void changeMatrix(String mat) {
        System.out.println("Selected matrix - " + matrix);
        myWrite.println("Матриця монітора змінена з " + matrix + " на " + mat);
        matrix = mat;
        System.out.println("Matrix changed - " + matrix);
    }

    /**
     * Method simulates changing monitor`s refresh rate
     */
    public void changeHz(int hz) {
        if (status == Status.ON) {
            System.out.println("Selected Hz - " + hz);
            myWrite.println("Частота оновлення монітора змінена з " + this.hz +
" на " + hz);
            this.hz = hz;
            System.out.println("Hz changed - " + hz);

        } else {
            System.out.println("Your monitor is off!!! ");
            myWrite.println("Частота оновлення монітора не змінена, та як
монітор виключено.");
        }
    }

    /**
     * Method simulates changing monitor`s name
     */
    public void changeName(String name) {
        System.out.println("Selected name - " + name);
        myWrite.println("Назва монітора змінена з " + this.name + " на " +
name);
        this.name = name;
        System.out.println("Name changed - " + name);
    }

    /**
     * Method simulates changing monitor`s resolution
     */
    public void changeResolution(String resolution) {
        if (status == Status.ON) {
            System.out.println("Selected resolution - " + resolution);
            myWrite.println("Розширення монітора змінено з " + this.resolution

```

```

+ " на " + resolution);
    this.resolution = resolution;
    System.out.println("Resolution changed - " + resolution);

    } else {
        System.out.println("Your monitor is off!!! ");
        myWrite.println("Розширення монітора не змінено, та як монітор
включено.");
    }
}

/**
 * Method simulates changing monitor's screen format
 */
public void changeMonitorFormat(String monitorFormat) {
    if (monitorFormat == "16:9") {
        this.monitorFormat = monitorFormat;
        System.out.println("Monitor format changed - " + monitorFormat);
        System.out.println("Max resolution for this format is 2560 x
1440.");
        myWrite.println("Формат монітора змінено на - " + monitorFormat);
    } else if (monitorFormat == "4:3") {
        this.monitorFormat = monitorFormat;
        System.out.println("Monitor format changed - " + monitorFormat);
        System.out.println("Max resolution for this format is 1440x1080.");
        myWrite.println("Формат монітора змінено на - " + monitorFormat);
    } else if (monitorFormat == "16:10") {
        this.monitorFormat = monitorFormat;
        System.out.println("Monitor format changed - " + monitorFormat);
        System.out.println("Max resolution for this format is 1920x1200.");
        myWrite.println("Формат монітора змінено на - " + monitorFormat);
    } else {
        System.out.println("Input incorrect monitor format!");
        myWrite.println("Формат монітора не змінено, некоректне
введення.");
    }
}

/**
 * Method releases used resources
 */
public void dispose()
{
    myWrite.flush();
    myWrite.close();
}
}

```

Файл TouchScreen.java

```

/**
 * lab 4 package
 */

package KI34.Lys.Lab4;

import java.io.FileNotFoundException;

/**
 * Class <code>TouchScreen</code> implements PresForce
 * * @author Lys Bohdan
 * * @version 1.0
 */
public class TouchScreen extends Monitor implements PresForce {
    private int resource;
    private final double presForceConst = 2.5;
    private double presForce;
}

```

```

/**
 * Constructor
 * @throws FileNotFoundException param
 */

public TouchScreen() throws FileNotFoundException {
    resource = 0;
    presForce = 0.0;
}

/**
 * Constructor
 * @param name <code>name</code> Name of monitor
 * @param diagonal <code>diagonal</code> Monitor diagonal
 * @param matrix <code>matrix</code> Monitor matrix
 * @param resolution <code>resolution</code> Monitor resolution
 * @param hz <code>hz</code> Monitor refresh rate
 * @param BrightDisplay <code>BrightDisplay</code> The brightness of the
monitor display
 * @param monitorFormat <code>monitorFormat</code> Monitor screen format
 * @param resource <code>resource</code> Touch monitor resource
 * @param presForce <code>presForce</code> Optimal pressing force
 * @throws FileNotFoundException
 */

public TouchScreen(int resource, double presForce, String name, double
diagonal, String matrix, String resolution, double hz, double BrightDisplay,
String monitorFormat) throws FileNotFoundException{
    super(name, diagonal, matrix, resolution, hz, BrightDisplay,
monitorFormat);
    this.resource = resource;
    this.presForce = presForce;
}

/**
 * Method simulates changing touch monitor's resource
 */
public void screenResource(int touches) {
    if (status == Status.ON)
    {
        if (presForce < presForceConst) {
            resource = resource - touches;
            myWrite.println("Ресурс сенсорного монітора зменшено на " +
touches);
        } else {
            aboveTheNorm(touches);
            myWrite.println("Ресурс сенсорного монітора зменшено на " +
error * touches);
        }
    }
    else{
        myWrite.println("Екран не реагує на нажаття тому, що монітор
виключено.");
    }

}

/**
 * Method return touch monitor's resource
 */
public void getResource()
{
    System.out.println("Touch monitor resource - " + resource);
    myWrite.println("Ресурс сенсорного монітора виведенно.");
}

```



```

        @Override
        public void aboveTheNorm(int touches) {
            resource = resource - touches * error;
        }
    }
}

```

Файл PresForce.java

```

/**
 * lab 4 package
 */

package KI34.Lys.Lab4;

/**
 * Interface <code>PresForce</code> extends Resource
 * * @author Lys Bohdan
 * * @version 1.0
 */
// оголошуємо інтерфейс PresForce, що успадковує інтерфейс Resource
public interface PresForce extends Resource{
    void aboveTheNorm (int touches); // прототип методу
    int error = 3; // константа
}

```

Файл Resource.java

```

/**
 * lab 4 package
 */

package KI34.Lys.Lab4;

/**
 * Interface <code>Resource</code>
 * * @author Lys Bohdan
 * * @version 1.0
 */
// оголошуємо інтерфейс Resource
public interface Resource {
    void screenResource (int touches); // прототип методу
}

```

Файл Status.java

```

/**
 * lab 4 package
 */

package KI34.Lys.Lab4;


/**
 * Method simulates off/on
 */
public enum Status {
    ON, OFF
}

```

Результат виконання програми:

```
C:\Users\mrpet\.jdk\openjdk-18.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\I
Name - MSI
Diagonal - 24.0
Resolution - 1920 x 1080
Hz - 144.0
Matrix - LED
Bright - 100.0
Selected matrix - LED
Matrix changed - IPS
Selected Hz - 75
Hz changed - 75
Selected resolution - 1080x720
Resolution changed - 1080x720
Monitor format changed - 4:3
Max resolution for this format is 1440x1080.
Name - MSI
Diagonal - 24.0
Resolution - 1080x720
Hz - 75.0
Matrix - IPS
Bright - 62.0
Touch monitor resource - 998266

Process finished with exit code 0
```

 Log.txt: Блокнот

Файл Редагування Формат Вигляд Довідка

Монітор включено.

Налаштування монітора виведенно.

Яскравість монітора збільшена.

Яскравість монітора зменшена.

Матриця монітора змінена з LED на IPS

Частота оновлення монітора змінена з 144.0 на 75

Розширення монітора змінено з 1920 x 1080 на 1080x720

Формат монітора змінено на - 4:3

Налаштування монітора виведенно.

Ресурс сенсорного монітора зменшено на 1734

Ресурс сенсорного монітора виведенно.

Монітор виключено.

Протокол діяльності

Package KI34.Lys.Lab4

package KI34.Lys.Lab4

All Classes and Interfaces	Interfaces	Classes	Enum Classes
Class		Description	
Monitor		Class Monitor implements monitor * @author Lys Bohdan * @version 1.0	
MonitorApp		Monitor Application class implements main method for Monitor class possibilities demonstration	
PresForce		Interface PresForce extends Resource * @author Lys Bohdan * @version 1.0	
Resource		Interface Resource * @author Lys Bohdan * @version 1.0	
Status		Method simulates off/on	
TouchScreen		Class TouchScreen implements PresForce * @author Lys Bohdan * @version 1.0	

Згенерована документація

SUMMARY: NESTED | FIELD | CONSTR | METHODDETAIL: FIELD | CONSTR | METHOD

Package KI34.Lys.Lab4

Class Monitor

java.lang.Object[Ⓢ]
KI34.Lys.Lab4.Monitor

Direct Known Subclasses:
TouchScreen

public abstract class Monitor
extends Object[Ⓢ]

Class Monitor implements monitor * @author Lys Bohdan * @version 1.0

Field Summary

Fields

Modifier and Type	Field	Description
PrintWriter [Ⓢ]	myWrite	
Status	status	

Constructor Summary

Constructors

Constructor	Description
Monitor()	Constructor
Monitor(String [Ⓢ] name, double diagonal, String [Ⓢ] matrix, String [Ⓢ] resolution, double hz, double BrightDisplay, String [Ⓢ] monitorFormat)	Constructor

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
void	brightDisplayMinus(int change)	Method simulates decrement monitor`s display brightness
void	brightDisplayPlus(int change)	Method simulates increment monitor`s display brightness
void	changeHz(int hz)	Method simulates changing monitor`s refresh rate
void	changeMatrix(String ^{id} mat)	Method simulates changing monitor`s matrix type
void	changeMonitorFormat(String ^{id} monitorFormat)	Method simulates changing monitor`s screen format
void	changeName(String ^{id} name)	Method simulates changing monitor`s name
void	changeResolution(String ^{id} resolution)	Method simulates changing monitor`s resolution
void	dispose()	Method releases used recourses
double	getBrightDisplay()	Method returns monitor`s display brightness
double	getDiagonal()	Method returns monitor`s diagonal
double	getHz()	Method returns monitor`s refresh rate
String ^{id}	getMatrix()	Method returns monitor`s matrix type
String ^{id}	getName()	Method returns monitor`s name
String ^{id}	getResolution()	Method returns monitor`s resolution
void	getSettings()	Method return monitor`s settings which we set
void	off_onMonitor(Status comand)	Method simulates monitor`s off/on
void	setBrightDisplay(double brightDisplay)	Method sets the new monitor`s display brightness
void	setDiagonal(double diagonal)	Method sets the new monitor`s diagonal
void	setHz(double hz)	Method sets the new monitor`s refresh rate
void	setMatrix(String ^{id} matrix)	Method sets the new monitor`s matrix type
void	setName(String ^{id} name)	Method sets the new monitor`s name
void	setResolution(String ^{id} resolution, String ^{id} monitorFormat)	Method sets the new monitor`s resolution and format

Інформація про клас Monitor

Package `KI34.Lys.Lab4`

Class MonitorApp

`java.lang.Object`
`KI34.Lys.Lab4.MonitorApp`

`public class MonitorApp`
`extends Object`

Monitor Application class implements main method for Monitor class possibilities demonstration

Version:

1.0

Author:

Lys Bohdan

Constructor Summary

Constructors

Constructor	Description
<code>MonitorApp()</code>	

Method Summary

All MethodsStatic MethodsConcrete Methods

Modifier and Type	Method	Description
<code>static void</code>	<code>main(String[] args)</code>	

Methods inherited from class `java.lang.Object`
`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Details

MonitorApp

`public MonitorApp()`

Інформація про клас `MonitorApp`

Package [KI34.Lys.Lab4](#)

Enum Class Status

[java.lang.Object](#)¹²
[java.lang.Enum](#)¹²<[Status](#)>
[KI34.Lys.Lab4.Status](#)

All Implemented Interfaces:

[Serializable](#)¹², [Comparable](#)¹²<[Status](#)>, [Constable](#)¹²

```
public enum Status
extends Enum<Status>
```

Method simulates off/on

Nested Class Summary

Nested classes/interfaces inherited from class [java.lang.Enum](#)¹²

[Enum.EnumDesc](#)¹²<[E](#)¹² extends [Enum](#)¹²<[E](#)¹²>>

Enum Constant Summary

Enum Constants

Enum Constant	Description
---------------	-------------

OFF

ON

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method	Description
static Status	valueOf (String ¹² name)	Returns the enum constant of this class with the specified name.
static Status []	values ()	Returns an array containing the constants of this enum class, in the order they are declared.

Methods inherited from class [java.lang.Enum](#)¹²

Інформація про клас EnumStatus

Package [KI34.Lys.Lab4](#)

Interface PresForce

All Superinterfaces:

[Resource](#)

All Known Implementing Classes:

[TouchScreen](#)

```
public interface PresForce
extends Resource
```

Interface PresForce extends [Resource](#) * @author Lys Bohdan * @version 1.0

Field Summary

Fields

Modifier and Type	Field	Description
static final int	error	

Method Summary

All Methods Instance Methods Abstract Methods

Modifier and Type	Method	Description
void	aboveTheNorm (int touches)	

Methods inherited from interface [KI34.Lys.Lab4.Resource](#)

[screenResource](#)

Інформація про інтерфейс PresForce

Package [KI34.Lys.Lab4](#)

Interface Resource

All Known Subinterfaces:

[PresForce](#)

All Known Implementing Classes:

[TouchScreen](#)

```
public interface Resource
```

Interface Resource * @author Lys Bohdan * @version 1.0

Method Summary

All Methods	Instance Methods	Abstract Methods
Modifier and Type	Method	Description
void	<code>screenResource(int touches)</code>	

Method Details

screenResource

```
void screenResource(int touches)
```

Інформація про інтерфейс Resource

Package [KI34.Lys.Lab4](#)

Class TouchScreen

`java.lang.Object`
`KI34.Lys.Lab4.Monitor`
`KI34.Lys.Lab4.TouchScreen`

All Implemented Interfaces:

`PresForce`, `Resource`

```
public class TouchScreen
    extends Monitor
    implements PresForce
```

Class TouchScreen implements PresForce * @author Lys Bohdan * @version 1.0

Field Summary

Fields inherited from class `KI34.Lys.Lab4.Monitor`

`myWrite`, `status`

Fields inherited from interface `KI34.Lys.Lab4.PresForce`

`error`

Constructor Summary

Constructors	Description
<code>TouchScreen()</code>	Constructor
<code>TouchScreen(int resource, double presForce, String[Ⓜ] name, double diagonal, String[Ⓜ] matrix, String[Ⓜ] resolution, double hz, double BrightDisplay, String[Ⓜ] monitorFormat)</code>	Constructor

Інформація про клас TouchScreen

Відповіді на контрольні запитання:

1. Синтаксис реалізації спадкування?

```
class Підклас extends Суперклас
{
    Додаткові поля і методи
}
```

2. Що таке суперклас та підклас?

Базовий клас найчастіше називається суперкласом, а похідний клас – підкласом.

3. Як звернутися до членів суперкласу з підкласу?

Виклик методу суперкласу:
super.назваМетоду([параметри]);

Звертання до поля суперкласу:
super.назваПоля

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

Статичне зв'язування використовується коли метод є приватним, статичним, фінальним або конструктором.

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

Віртуальна машина повинна викликати версію методу, що відповідає фактичному типу об'єкту на який посилається об'єктна змінна.

Оскільки на пошук необхідного методу потрібно багато часу, то віртуальна машина заздалегідь створює для кожного класу таблицю методів, в якій перелічуються сигнатури всіх методів і фактичні методи, що підлягають виклику. При виклику методу віртуальна машина просто переглядає таблицю методів.

6. Що таке абстрактний клас та як його реалізувати?

Абстрактний клас – це клас, для якого не можна створити об'єкти, призначений бути основою для розробки ієрархії класів.

*Реалізується за допомогою ключового слова **abstract**.*

7. Для чого використовується ключове слово instanceof?

*Оператор **instanceof** дозволяє визначити, чи вказаний об'єкт належить до заданого типу.*

8. Як перевірити чи клас є підкласом іншого класу?

*Щоб перевірити чи клас є підкласом іншого класу, потрібно за допомогою **instanceof** порівняти, чи дійсно посилання на об'єкт супертипу посилається на об'єкт підтипу.*

9. Що таке інтерфейс?

Це абстрактний тип, який використовується для визначення поведінки, яку класи повинні реалізовувати

10. Як оголосити та застосувати інтерфейс?

```
[public] interface НазваІнтерфейсу  
{  
    Прототипи методів та оголошення констант інтерфейсу  
}
```

Висновок:

На цій лабораторній роботі, я ознайомитися з спадкуванням та інтерфейсами у мові Java.