Міністерство освіти та науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



**ЗВІТ**

**з лабораторної роботи № 4**

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

**на тему:** «СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ»

Виконав: ст. гр. КІ-35

Гоцій М.М.

Прийняв:

доцент кафедри 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.

**Варіант 11:**

****

**Текст програми:**

***Лістинг Monitorable***

package KI35.Hotsii.Lab4;  
  
*/\*\*  
 \* Interface for monitors  
 \*/*public interface Monitorable {  
 */\*\*  
 \* This method turns on the monitor  
 \*/* void turnOnScreen();  
 */\*\*  
 \* This method turns off the monitor  
 \*/* void turnOfScreen();  
}

***Лістинг Resolution***

package KI35.Hotsii.Lab4;  
  
*/\*\*  
 \* Enum <code>Resolution</code> contains all available resolution values.  
 \** ***@author*** *Hotsii Mykyta  
 \*/*public enum Resolution {  
 *R1920x1080*,  
 *R1680x1059*,  
 *R1600x900*,  
 *R1440x900*,  
 *R1400x1050*}

***Лістинг Sensorable***

package KI35.Hotsii.Lab4;  
  
*/\*\*  
 \* Interface for sensor tech  
 \*/*public interface Sensorable {  
 */\*\*  
 \* This method is called when user holds his finger on the screen  
 \*/* void hold();  
 */\*\*  
 \* This method is called when user swipes his finger down from top to bottom  
 \*/* void scroll();  
 */\*\*  
 \* This method is called when user touches the screen.  
 \*/* void touch();  
}

***Лістинг SensorScreen***

package KI35.Hotsii.Lab4;  
  
import java.util.function.Consumer;  
  
public class SensorScreen extends Monitor implements Sensorable{  
 */\*\*  
 \* Constructor  
 \*/* public SensorScreen() {  
 }  
  
 */\*\*  
 \* Constructor  
 \** ***@param*** *controller - controller  
 \** ***@param*** *mainDisplay - main display  
 \** ***@param*** *corps - corps  
 \** ***@param*** *manufacturer - manufacturer  
 \*/* public SensorScreen(Controller controller, Display mainDisplay, Corps corps, String manufacturer) {  
 super(controller, mainDisplay, corps, manufacturer);  
 }  
  
 @Override  
 public void hold() {  
 executeIfTurnedOn(System.*out*::println, "User is holding his finger.");  
 }  
  
 @Override  
 public void scroll() {  
 executeIfTurnedOn(System.*out*::println, "User swipe his finger down from top to bottom.");  
 }  
  
 @Override  
 public void touch() {  
 executeIfTurnedOn(System.*out*::println, "User touched the screen.");  
 }  
  
 private void executeIfTurnedOn(Consumer<String> consumer, String value) {  
 if (turnedOn) {  
 consumer.accept(value);  
 } else {  
 System.*out*.println("The screen is off.");  
 }  
 }  
}

***Лістинг Helper***

package utils;  
  
import java.io.\*;  
  
*/\*\*  
 \* Class <code>Helper</code> implements helper.  
 \** ***@author*** *Hotsii Mykyta  
 \*/*public class Helper {  
 public static PrintWriter getWriter(String path) {  
 try {  
 return new PrintWriter(new FileWriter(path), true);  
 } catch (IOException e) {  
 throw new IllegalArgumentException("There's no file with such path");  
 }  
 }  
}

***Лістинг Validator***

package utils;  
  
*/\*\*  
 \* Class <code>Validator</code> implements validator.  
 \** ***@author*** *Hotsii Mykyta  
 \*/*public class Validator {  
 public static void validateRange(int... values) {  
 for (int field: values) {  
 if (field <= 0)  
 throw new IllegalStateException("Value must be greater than zero.");  
 }  
 }  
  
 public static void validateObjectValues(Object... objects) {  
 for (Object object: objects) {  
 if (object == null)  
 throw new IllegalStateException("Object value can't be null.");  
 }  
 }  
  
 public static void validateStringValue(String value) {  
 if (value == null || value.isEmpty())  
 throw new IllegalStateException("String value can't be empty or null.");  
 }  
}

***Лістинг Main***

import KI35.Hotsii.Lab4.\*;  
  
import java.io.IOException;  
  
*/\*\*  
 \* Monitor Application class implements main method for Monitor class possibilities demonstration  
 \** ***@author*** *Mykyta Hotsii  
 \** ***@version*** *1.0  
 \*/*public class Main {  
 */\*\*  
 \** ***@param*** *args  
 \** ***@throws*** *IOException  
 \*/* public static void main(String[] args) throws IOException {  
 SensorScreen monitor = new SensorScreen();  
 Display display = new Display();  
 display.setResolution(Resolution.*R1400x1050*);  
 display.setScale(100);  
 Corps corps = new Corps();  
 corps.setDiagonalInInches(35);  
 corps.setMaterial("iron");  
 Controller controller = new Controller();  
 controller.setAvailableLVDS(true);  
 controller.setHeight(124);  
 controller.setWidth(60);  
 controller.setVoltage(16);  
 monitor.addAdditionalDisplay(display);  
 monitor.setController(controller);  
 monitor.setCorps(corps);  
 monitor.setManufacturer("Apple");  
 monitor.removeAdditionalDisplay(display);  
  
 monitor.turnOnScreen();  
 monitor.hold();  
 monitor.scroll();  
 monitor.touch();  
 monitor.turnOfScreen();  
 monitor.touch();  
  
 monitor.dispose();  
 }  
}

|  |
| --- |
| ***Лістинг Controller***  package KI35.Hotsii.Lab4;  import static utils.Validator.*validateRange*;  */\*\*  \* Class <code>Controller</code> implements controller.  \** ***@author*** *Hotsii Mykyta  \*/* public class Controller {  private int voltage;  private int width;  private int height;  private boolean availableLVDS;   */\*\*  \* Constructor  \*/* public Controller() {  this.availableLVDS = true;  this.width = 140;  this.height = 60;  this.voltage = 12;  }   */\*\*  \* Constructor  \** ***@param*** *width - width value  \** ***@param*** *height - height value  \** ***@param*** *voltage - voltage value  \*/* public Controller(int width, int height, int voltage) {  *validateRange*(width, height, voltage);  this.width = width;  this.height = height;  this.voltage = voltage;  }   */\*\*  \* Method sets voltage value  \** ***@param*** *voltage - voltage value  \*/* public void setVoltage(int voltage) {  *validateRange*(voltage);  this.voltage = voltage;  }   */\*\*  \* Method sets width value  \** ***@param*** *width - width value  \*/* public void setWidth(int width) {  *validateRange*(voltage);  this.width = width;  }   */\*\*  \* Method sets height value  \** ***@param*** *height - height value  \*/* public void setHeight(int height) {  *validateRange*(voltage);  this.height = height;  }   */\*\*  \* Method sets is availableLVDS available  \** ***@param*** *availableLVDS - is availableLVDS available  \*/* public void setAvailableLVDS(boolean availableLVDS) {  this.availableLVDS = availableLVDS;  }   */\*\*  \* Method returns controller's voltage  \** ***@return*** *Controller's voltage  \*/* public int getVoltage() {  return voltage;  }   */\*\*  \* Method returns controller's width  \** ***@return*** *Controller's width  \*/* public int getWidth() {  return width;  }   */\*\*  \* Method returns controller's height  \** ***@return*** *Controller's height  \*/* public int getHeight() {  return height;  }   */\*\*  \* Method returns boolean value that describes the ability of the LVDC to use  \** ***@return*** *Controller's height  \*/* public boolean isAvailableLVDS() {  return availableLVDS;  }   @Override  public String toString() {  return "Controller{" +  "voltage=" + voltage +  ", width=" + width +  ", height=" + height +  ", availableLVDS=" + availableLVDS +  '}';  } }  ***Лістинг Corps***  package KI35.Hotsii.Lab4;  import static utils.Validator.\*;  */\*\*  \* Class <code>Corps</code> implements corps.  \** ***@author*** *Hotsii Mykyta  \*/* public class Corps {  private String material;  private int diagonalInInches;   */\*\*  \* Constructor  \*/* public Corps() {  this.material = "plastic";  this.diagonalInInches = 20;  }   */\*\*  \* Constructor  \** ***@param*** *width - width value  \** ***@param*** *height - height value  \** ***@param*** *material - material value  \** ***@param*** *diagonalInInches - diagonalInInches value  \*/* public Corps(int width, int height, String material, int diagonalInInches) {  *validateRange*(diagonalInInches);  this.material = material;  this.diagonalInInches = diagonalInInches;  }   */\*\*  \* Method returns corps's material  \** ***@return*** *Corps's material  \*/* public String getMaterial() {  return material;  }   */\*\*  \* Method returns corps's diagonalInInches  \** ***@return*** *Corps's diagonalInInches  \*/* public int getDiagonalInInches() {  return diagonalInInches;  }   */\*\*  \* Method sets diagonalInInches value  \** ***@param*** *diagonalInInches - diagonalInInches value  \*/* public void setDiagonalInInches(int diagonalInInches) {  *validateRange*(diagonalInInches);  this.diagonalInInches = diagonalInInches;  }   */\*\*  \* Method sets material value  \** ***@param*** *material - diagonalInInches value  \*/* public void setMaterial(String material) {  this.material = material;  }   @Override  public String toString() {  return "Corps{" +  "material='" + material + '\'' +  ", diagonalInInches=" + diagonalInInches +  '}';  } }  ***Лістинг Display***  package KI35.Hotsii.Lab4;  import java.util.Objects;  import static KI35.Hotsii.Lab4.Resolution.\*; import static utils.Validator.*validateRange*;  */\*\*  \* Class <code>Display</code> implements display.  \** ***@author*** *Hotsii Mykyta  \*/* public class Display {  private Resolution resolution;  private int scaleInPercent;   */\*\*  \* Constructor  \*/* public Display() {  this.resolution = *R1400x1050*;  this.scaleInPercent = 125;  }   */\*\*  \* Constructor  \** ***@param*** *resolution - resolution value  \** ***@param*** *scaleInPercent - scale in percent value  \*/* public Display(Resolution resolution, int scaleInPercent) {  this.resolution = resolution;  this.scaleInPercent = scaleInPercent;  }   */\*\*  \* Method return resolution value  \** ***@return*** *resolution value  \*/* public Resolution getResolution() {  return resolution;  }   */\*\*  \* Method return scale value  \** ***@return*** *scale value  \*/* public int getScale() {  return scaleInPercent;  }   */\*\*  \* Method sets scale in percent value  \** ***@param*** *scaleInPercent - scale in percent value  \*/* public void setScale(int scaleInPercent) {  *validateRange*(scaleInPercent);  this.scaleInPercent = scaleInPercent;  }   */\*\*  \* Method sets resolution  \** ***@param*** *resolution - resolution value  \*/* public void setResolution(Resolution resolution) {  this.resolution = resolution;  }   @Override  public boolean equals(Object o) {  if (this == o) return true;  if (!(o instanceof Display)) return false;  Display display = (Display) o;  return scaleInPercent == display.scaleInPercent &&  resolution == display.resolution;  }   @Override  public int hashCode() {  return Objects.*hash*(resolution, scaleInPercent);  }   @Override  public String toString() {  return "Display{" +  "resolution=" + resolution +  ", scaleInPercent=" + scaleInPercent +  '}';  } }  ***Лістинг Monitor***  package KI35.Hotsii.Lab4;  import java.io.PrintWriter; import java.util.ArrayList; import java.util.Collections; import java.util.List;  import static utils.Helper.*getWriter*; import static utils.Validator.*validateObjectValues*; import static utils.Validator.*validateStringValue*;  */\*\*  \* Class <code>Monitor</code> implements monitor.  \** ***@author*** *Hotsii Mykyta  \*/* public class Monitor {  private Controller controller;  private Corps corps;  private String manufacturer;  private Display mainDisplay;  private final List<Display> additionalDisplays = new ArrayList<>();  private final PrintWriter fileWriter = *getWriter*("text.txt");  protected boolean turnedOn;   */\*\*  \* Constructor  \*/* public Monitor() {  this.controller = new Controller();  this.mainDisplay = new Display();  this.corps = new Corps();  this.manufacturer = "Sumsung";  fileWriter.printf("Monitor is created. %n{%n%s%n}%n%n", this);  }   */\*\*  \* Constructor  \** ***@param*** *controller - controller  \** ***@param*** *mainDisplay - main display  \** ***@param*** *corps - corps  \** ***@param*** *manufacturer - manufacturer  \*/* public Monitor(Controller controller, Display mainDisplay, Corps corps, String manufacturer) {  *validateStringValue*(manufacturer);  this.controller = controller;  this.mainDisplay = mainDisplay;  this.corps = corps;  this.manufacturer = manufacturer;  fileWriter.printf("Monitor is created. %n{%n%s%n}%n%n", this);  }   */\*\*  \* Method returns controller  \** ***@return*** *controller  \*/* public Controller getController() {  return controller;  }   */\*\*  \* Method returns main display  \** ***@return*** *main display  \*/* public Display getMainDisplay() {  return mainDisplay;  }   */\*\*  \* Method returns main corps  \** ***@return*** *corps  \*/* public Corps getCorps() {  return corps;  }   */\*\*  \* Method returns manufacturer  \** ***@return*** *manufacturer  \*/* public String getManufacturer() {  return manufacturer;  }   */\*\*  \* Method returns additional display  \** ***@return*** *additional display  \*/* public List<Display> getAdditionalDisplays() {  return Collections.*unmodifiableList*(additionalDisplays);  }   */\*\*  \* Method adds additional display  \** ***@param*** *display - display  \*/* public void addAdditionalDisplay(Display display) {  *validateObjectValues*(display);  if (display == mainDisplay)  throw new IllegalArgumentException("This display is already set as the main display");  additionalDisplays.add(display);  fileWriter.printf("Added additional display. %n{%n%s%n}%n%n", display);  }   */\*\*  \* Method removes additional display  \** ***@param*** *display - display  \*/* public void removeAdditionalDisplay(Display display) {  *validateObjectValues*(display);  additionalDisplays.remove(display);  fileWriter.printf("Deleted additional display. %n{%n%s%n}%n%n", display);  }   */\*\*  \* Method sets controller  \** ***@param*** *controller - controller  \*/* public void setController(Controller controller) {  *validateObjectValues*(controller);  this.controller = controller;  fileWriter.printf("The controller has been changed. %n{%n%s%n}%n%n", controller);  }   */\*\*  \* Method sets main display  \** ***@param*** *mainDisplay - main display  \*/* public void setMainDisplay(Display mainDisplay) {  *validateObjectValues*(mainDisplay);  if (additionalDisplays.contains(mainDisplay))  throw new IllegalArgumentException("This display is already set as the additional display");  this.mainDisplay = mainDisplay;  fileWriter.printf("The main display has been changed. %n{%n%s%n}%n%n", mainDisplay);  }   */\*\*  \* Method sets corps  \** ***@param*** *corps - corps  \*/* public void setCorps(Corps corps) {  *validateObjectValues*(corps);  this.corps = corps;  fileWriter.printf("The corps has been changed. %n{%n%s%n}%n%n", corps);  }   */\*\*  \* Method sets manufacturer  \** ***@param*** *manufacturer - manufacturer  \*/* public void setManufacturer(String manufacturer) {  *validateStringValue*(manufacturer);  this.manufacturer = manufacturer;  fileWriter.printf("The manufacturer has been changed. %n{%n%s%n}%n%n", manufacturer);  }   */\*\*  \* Method closes resources  \*/* public void dispose()  {  fileWriter.close();  }   public void turnOnScreen() {  this.turnedOn = true;  System.*out*.println("Turn on the screen.");  }   public void turnOfScreen() {  this.turnedOn = false;  System.*out*.println("Turn of the screen.");  }    @Override  public String toString() {  return "Monitor{" +  "controller=" + controller +  ", corps=" + corps +  ", manufacturer='" + manufacturer + '\'' +  ", mainDisplay=" + mainDisplay +  ", additionalDisplays=" + additionalDisplays +  ", fileWriter=" + fileWriter +  '}';  } } |

**Результат роботи:**

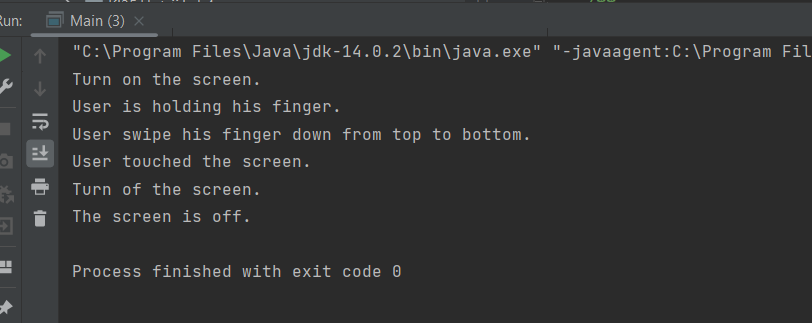
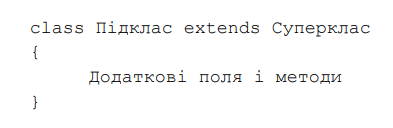


Рис.1 Результат роботи програми

**Відповіді на контрольні запитання:**

1. Синтаксис реалізації спадкування.

Спадкування в ООП призначене для розширення функціональності існуючих класів шляхом утворення нових класів на базі вже існуючих. У Java реалізована однокоренева архітектура класів згідно якої всі класи мають єдиного спільного предка (кореневий клас в ієрархії класів) – клас Object. Решта класів мови Java утворюються шляхом успадковування даного класу. Будь-яке спадкування у мові Java є відкритим, при цьому аналогів захищеному і приватному спадкуванню мови С++ не існує. На відміну від С++ у Java можливе спадкування лише одного базового класу (множинне спадкування відсутнє). Спадкування реалізується шляхом вказування ключового слова class піcля якого вказується назва підкласу, ключове слово extends та назва суперкласу, що розширюється у новому підкласі. Синтаксис реалізації спадкування:



1. Що таке суперклас та підклас?

В термінах мови Java базовий клас найчастіше називається суперкласом, а похідний клас – підкласом. Дана термінологія запозичена з теорії множин, де підмножина міститься у супермножині.

**Висновок:**

Під час виконання роботи я ознайомився з спадкуванням та інтерфейсами у мові Java.