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

Рибець О.В.

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 7:**

****

**Текст програми:**

|  |
| --- |
| ***Лістинг ComputerAbstract***  package lab4;  import lab3.Keyboard; import lab3.Monitor; import lab3.Mouse;  import java.io.PrintWriter;  public abstract class ComputerAbstract extends Object {   protected Mouse m\_mouse = new Mouse();  protected Keyboard k\_keybord = new Keyboard();  protected Monitor m\_monitor = new Monitor();  protected PrintWriter fout;  public void disclose(){  fout.close();  }  public void printConnectionOfDevices(){  System.*out*.println("Current state of monitor:" + m\_monitor.isM\_state());  System.*out*.println("Current state of keyboard:" + k\_keybord.isK\_state());  System.*out*.println("Current state of mouse:" + m\_mouse.isMo\_state());  }  public void printConnectionOfMouse(){  if(m\_mouse.isMo\_state()){  System.*out*.println("Mouse is connected");  }else {  System.*out*.println("Mouse is disconnected");  }  }  public void printConnectionOfKeyboard(){  if(k\_keybord.isK\_state()){  System.*out*.println("Keyboard is connected");  }else {  System.*out*.println("Keyboard is disconnected");  }  }  public void printConnectionOfMonitor(){  if(m\_monitor.isM\_state()){  System.*out*.println("Monitor is connected");  }else {  System.*out*.println("Monitor is disconnected");  }  }  public void AttachMonitor(){  if(!m\_monitor.isM\_state()){  m\_monitor.setM\_state(true);   fout.println("New state of monitor: " + m\_monitor.isM\_state());  System.*out*.println("Monitor connected!!!");  }  else{  System.*out*.println("The monitor is already attached!");  }  }  public void RemoveMonitor(){  if(m\_monitor.isM\_state()){  m\_monitor.setM\_state(false);   fout.println("New state of monitor: " + m\_monitor.isM\_state());  System.*out*.println("Monitor removed!!!");  }  else{  System.*out*.println("The monitor is already removed!");  }  }   public void AttachMouse(){  if(!m\_mouse.isMo\_state()){  m\_mouse.setMo\_state(true);   fout.println("New state of mouse: " + m\_mouse.isMo\_state());  System.*out*.println("Mouse connected!!!");  }  else{  System.*out*.println("The mouse is already attached!");  }  }   public void RemoveMouse(){  if(m\_mouse.isMo\_state()){  m\_mouse.setMo\_state(false);   fout.println("New state of mouse: " + m\_mouse.isMo\_state());  System.*out*.println("Mouse removed!!!");  }  else{  System.*out*.println("The mouse is already removed!");  }  }  public void AttachKeyboard(){  if(!k\_keybord.isK\_state()){  k\_keybord.setK\_state(true);   fout.println("New state of keyboard: " + k\_keybord.isK\_state());  System.*out*.println("Keyboard connected!!!");  }  else{  System.*out*.println("The keyboard is already attached!");  }  }  public void RemoveKeyboard(){  if(k\_keybord.isK\_state()){  k\_keybord.setK\_state(false);   fout.println("New state of keyboard: " + k\_keybord.isK\_state());  System.*out*.println("Keyboard removed!!!");  }  else{  System.*out*.println("The keyboard is already removed!");  }  }  class Keyboard {   private boolean K\_state;   public Keyboard(){  this.K\_state = false;  }   public boolean isK\_state() {  return K\_state;  }   public void setK\_state(boolean k\_state) {  K\_state = k\_state;  }  }  class Monitor {  private boolean M\_state;  public Monitor(){  this.M\_state = false;  }   public boolean isM\_state() {  return M\_state;  }   public void setM\_state(boolean m\_state) {  M\_state = m\_state;  }  }  class Mouse {  private boolean Mo\_state;   public Mouse() {  this.Mo\_state = false;  }    public boolean isMo\_state() {  return Mo\_state;  }   public void setMo\_state(boolean mo\_state) {  Mo\_state = mo\_state;  }  } }  }  ***Лістинг Laptop***  package lab4;  import lab3.Keyboard; import lab3.Monitor; import lab3.Mouse;  import java.io.File; import java.io.FileNotFoundException; import java.io.PrintWriter;  public class Laptop extends ComputerAbstract implements BatteryPower {  boolean power;   public Laptop(){  super();  }   public boolean isPower() {  return power;  }   public void setPower(boolean power) {  this.power = power;  }   public void AttachMouse(){  if(!m\_mouse.isMo\_state()){  m\_mouse.setMo\_state(true);   fout.println("New state of mouse: " + m\_mouse.isMo\_state());  System.*out*.println("Mouse connected!!!");  }  else{  System.*out*.println("The mouse is already attached!");  }  }   public void RemoveMouse(){  if(m\_mouse.isMo\_state()){  m\_mouse.setMo\_state(false);   fout.println("New state of mouse: " + m\_mouse.isMo\_state());  System.*out*.println("Mouse removed!!!");  }  else{  System.*out*.println("The mouse is already removed!");  }  }   public void stateOfPower() {  if(power){  System.*out*.println("Laptop is connected to chain");  }  else if(!power) {  System.*out*.println("Laptop is disconnected from chain");  }  }   @Override  public void turnOnLaptop() {  System.*out*.println("The laptop is turned on");  } }  }  ***Лістинг BatteryPower***  package lab4;  public abstract interface BatteryPower {  void turnOnLaptop();  }  ***Лістинг LaptopApp***  package lab4;  public class LaptopApp {  public static void main(String[] args) {  Laptop laptop = new Laptop();  laptop.turnOnLaptop();    } } |

**Результат роботи:**

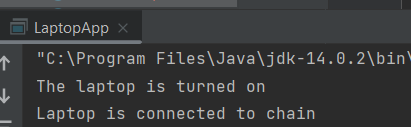
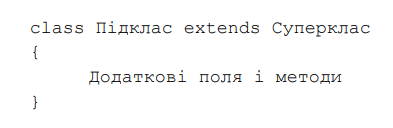
****

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.