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

Хван Д.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 23:**

****

**Текст програми:**

|  |
| --- |
| import java.io.FileNotFoundException;  public class Burner {      public Location location;      public Burner(Location location) throws FileNotFoundException {          this.location = location;      }      public Burner(){}      /\*\*       \* Method implements burning of one plate       \*       \*/      public void burn() {          System.out.println("Burner" + this.location.toString() + "is burning \n");      }      /\*\*       \* Method implements stopBurning       \*       \*/      public void stopBurn() {          System.out.println("Burner" + this.location.toString() + "stops burning");      }  }  import java.io.FileNotFoundException;  interface ChoiceTemp{      void choiceTemp(Temperature temperature);  }  public class GasStove extends KitchenStove implements ChoiceTemp{      private StovesButton OvenButton;      private Temperature temperature;      public GasStove() throws FileNotFoundException {          OvenButton = new StovesButton(Location.Central);          temperature=Temperature.t100;      }      public void turnOnOvenButton() {          OvenButton.turnOnButton(3);      }      public void turnOffOvenButton() {          OvenButton.turnOffButton();      }      public void choiceTemp(Temperature \_temperature){          temperature=\_temperature;      }  }  import java.io.\*;  /\*\*   \* Class <code>ComputerMouse</code> implements computer mouse   \*   \* @author EOM Stuff   \* @version 1.0   \*/  public abstract class KitchenStove {      private StovesButton leftUpButtonForBurner;      private StovesButton leftDownButtonForBurner;      private StovesButton rightUpButtonForBurner;      private StovesButton rightDownButtonForBurner;      public static int countOfStoves;      /\*\*       \* Constructor       \*       \* @throws FileNotFoundException       \*/      public KitchenStove() throws FileNotFoundException {          leftUpButtonForBurner = new StovesButton(Location.leftUP);          leftDownButtonForBurner = new StovesButton(Location.leftDown);          rightUpButtonForBurner = new StovesButton(Location.RightUP);          rightDownButtonForBurner = new StovesButton(Location.RightDown);      }      /\*\*       \* Constructor       \*       \* @param resource Button clicks resource       \* @throws FileNotFoundException       \*/      public KitchenStove(int resource) throws FileNotFoundException {          countOfStoves++;      }      /\*\*       \* Method implement turn on Left Up plate       \*       \*/      public void turnOnLeftUpButton() {          leftUpButtonForBurner.turnOnButton(5);      }      /\*\*       \* Method implement turn on Left Down plate       \*       \*/      public void turnOnLeftDownButton() {          leftDownButtonForBurner.turnOnButton(4);      }      /\*\*       \* Method implement turn on Right Up plate       \*       \*/      public void turnOnRightUpButton() {          rightUpButtonForBurner.turnOnButton(7);      }      /\*\*       \* Method implement turn on Right Down plate       \*       \*/      public void turnOnRightDownButton() {          rightDownButtonForBurner.turnOnButton(2);      }      /\*\*       \* Method implement turn on Left Up plate       \*       \*/      public void turnOffLeftUpButton() {          leftUpButtonForBurner.turnOffButton();      }      /\*\*       \* Method implement turn on Left Down plate       \*       \*/      public void turnOffLeftDownButton() {          leftDownButtonForBurner.turnOffButton();      }      /\*\*       \* Method implement turn on Right Up plate       \*       \*/      public void turnOffRightUpButton() {          rightUpButtonForBurner.turnOffButton();      }      /\*\*       \* Method implement turn on Right Down plate       \*       \*/      public void turnOffRightDownButton() {          rightDownButtonForBurner.turnOffButton();      }  }  enum Location {      leftUP,      leftDown,      RightUP,      RightDown,      Central,  }  /\*\*   \* Class <code>RelativePosition</code> implements relative positioning   \* coordinate system   \*   \* @author EOMStuff   \* @version 1.0   \*/  class RelativePosition {      // coordinates of the mouse position      private int x;      /\*\*       \* Constructor       \*/      public RelativePosition() {          x = 0;      }      /\*\*       \* Constructor       \*       \* @param xPos The X coordinate value       \*/      public RelativePosition(int xPos) {          x = xPos;      }      /\*\*       \* Method returns the X coordinate value       \*       \* @return The X coordinate value       \*/      public int getXPosition() {          return x;      }      /\*\*       \* Method returns coordinates of the position in the <code>obj</code>,       \* that is passed into method through method parameter       \*       \* @param obj The object, where coordinates of the current       \*                         position are set       \*/      public void getPosition(RelativePosition obj) {          obj.x = x;      }      /\*\*       \* Method sets the X coordinate value       \*       \* @param xPos The X coordinate value       \*/      public void setXPosition(int xPos) {          if (xPos < 0) {              xPos = 0;          }          x = xPos;      }  }  import java.io.FileNotFoundException;  public class StovesButton {      private Location location;      private Burner burner;      private RelativePosition position;      private static boolean turnOn;      public StovesButton(Location location) throws FileNotFoundException {          this.location = location;          this.burner = new Burner();          this.burner.location = this.location;          this.position = new RelativePosition();          turnOn = false;      }      /\*\*       \* Method implements pressing on a button og kitchen stove       \*       \* @param xPos       \*/      public void turnOnButton(int xPos) {          position.setXPosition(xPos);          System.out.println("Button" + this.location.toString() + "is pressed\n");          if (xPos > 0) {              burner.burn();              turnOn = true;          }      }      boolean isTurnOn() {          return turnOn;      }      public void turnOffButton() {          position.setXPosition(0);          System.out.println("Button" + this.location.toString() + "is pressed\n");          if(!isTurnOn())          {              System.out.println("Plate was turned off");              return;          }          turnOn = false;          burner.stopBurn();      }  }  enum Temperature{      t100,      t150,      t200,      t250,  } |

**Результат роботи:**

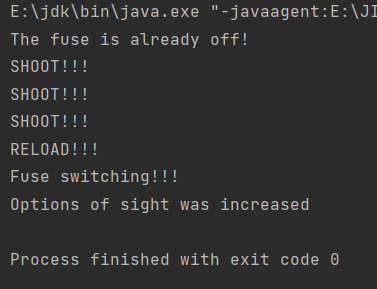
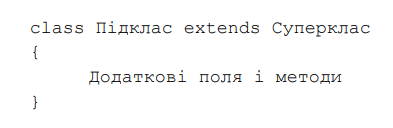
****

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.