Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

З лабораторної роботи №2

Варіант – 24

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Класи та пакети»

Виконав: ст. гр. КІ-301

Чоланюк О.Ю

Прийняв:

Майдан М.В.

Львів 2023

**Мета роботи:** ознайомитися з процесом розробки класів та пакетів мовою Java.

**Теоретичні відомості**

**Класи**

Мова Java є повністю об’єктно-орієнтованою мовою програмування, тому вона дозволяє писати програми лише з використанням об’єктно-орієнтованих парадигм програмування, що базуються на понятті класів.

Синтаксис оголошення простого класу в мові Java має наступний вигляд:

[public] class НазваКласу

{

[конструктори] [методи] [поля]

}

Необов’язковий специфікатор доступу public робить клас загальнодоступним. У кожному файлі з кодом програми може бути лише один загальнодоступний клас, ім’я якого співпадає з назвою файлу, та безліч класів без специфікатора public.

## Методи

*Метод* – функція-член класу, яка призначена маніпулювати станом об’єкту класу. Методи можуть бути перевантаженими. Перевантаження методів відбувається шляхом вказування різної кількості параметрів та їх типів методам з однаковими назвами. Синтаксис оголошення методу наступний:

[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]

{

[Тіло методу] [return [значення]];

}

Конструктори, методи, та поля класу можуть бути відкритими (public),

закритими (private) та захищеними (protected), що визначається специфікатором доступу.

## Конструктори

*Конструктор* – спеціальний метод класу, який не повертає значення, має ім’я класу та призначений для початкової ініціалізації об’єктів класу. Синтаксис оголошення конструктора:

[СпецифікаторДоступу] НазваКласу([параметри])

{

Тіло конструктора

}

## Поля

*Поле* (властивість) – це дані-члени класу, що призначені для зберігання стану об’єкту. Поле може бути статичним (в цьому випадку воно називається *полем класу*), незмінним (*константне поле*), простим типом чи об’єктом та мати різні рівні доступу, що визначаються специфікатором доступу. Допускається ініціалізація поля в місці оголошення. Синтаксис оголошення поля наступний:

[СпецифікаторДоступу] [static] [final] Тип НазваПоля [= ПочатковеЗначення];

**Завдання**

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту. Програма має задовольняти наступним вимогам:
   * програма має розміщуватися в пакеті Група.Прізвище.Lab2;
   * клас має містити мінімум 3 поля, що є об’єктами класів, які описують складові частини предметної області;
   * клас має містити кілька конструкторів та мінімум 10 методів;
   * для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;
   * методи класу мають вести протокол своєї діяльності, що записується у файл;
   * розробити механізм коректного завершення роботи з файлом (не надіятися на метод finalize());
   * програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленої програми.
3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

Варіант 24: спорядження альпініста

package lab2;  
  
import java.io.\*;  
import java.io.IOException;  
import java.io.FileWriter;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
public class Main {  
 private ClimbingEquipment climbingEquipment;  
 private Position currentPosition;  
 private Carabiner rightCarabiner;  
 private Carabiner leftCarabiner;  
 private PrintWriter fout;  
 private PrintWriter actionLogWriter;  
  
  
 private static final String *LOG\_FILE* = "Log.txt";  
  
  
 private static final String *ACTION\_LOG\_FILE* = "ActionLog.txt";  
  
 public Main() throws IOException {  
 climbingEquipment = new ClimbingEquipment();  
 currentPosition = new Position();  
 rightCarabiner = new Carabiner();  
 leftCarabiner = new Carabiner();  
 fout = new PrintWriter(new File(*LOG\_FILE*));  
 actionLogWriter = new PrintWriter(new FileWriter(*ACTION\_LOG\_FILE*, true));  
 }  
  
 private void logToFile(String message) {  
 SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
 String timestamp = dateFormat.format(new Date());  
  
 String logMessage = "[" + timestamp + "] " + message;  
  
 try (FileWriter writer = new FileWriter(*ACTION\_LOG\_FILE*, true)) {  
 writer.write(logMessage + "\n");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void setPosition(int xPosition, int yPosition) {  
 currentPosition.setXPosition(xPosition);  
 currentPosition.setYPosition(yPosition);  
 logToFile("Set climber position. X: " + xPosition + ", Y: " + yPosition);  
 fout.print("Set climber position. X: " + xPosition + ", Y: " + yPosition + "\n");  
 fout.flush();  
 }  
  
 public void clickLeftButton() {  
 leftCarabiner.clickButton();  
 logToFile("Clicked left carabiner button. New resource: " + leftCarabiner.getButtonResource());  
 fout.print("Clicked left carabiner button. New resource: " + leftCarabiner.getButtonResource() + "\n");  
 fout.flush();  
 }  
  
 public void clickRightButton() {  
 rightCarabiner.clickButton();  
 logToFile("Clicked right carabiner button. New resource: " + rightCarabiner.getButtonResource());  
 fout.print("Clicked right carabiner button. New resource: " + rightCarabiner.getButtonResource() + "\n");  
 fout.flush();  
 }  
  
 public int getRightButtonResource() {  
 logToFile("Requested right carabiner button resource: " + rightCarabiner.getButtonResource());  
 return rightCarabiner.getButtonResource();  
 }  
  
 public int getLeftButtonResource() {  
 logToFile("Requested left carabiner button resource: " + leftCarabiner.getButtonResource());  
 return leftCarabiner.getButtonResource();  
 }  
  
 public void scrollUp() {  
 climbingEquipment.setUpDirection();  
 logToFile("Scrolled up. Climbing equipment direction: " + climbingEquipment.getDirection());  
 fout.print("Scrolled up. Climbing equipment direction: " + climbingEquipment.getDirection() + "\n");  
 fout.flush();  
 }  
  
 public void scrollDown() {  
 climbingEquipment.setDownDirection();  
 logToFile("Scrolled down. Climbing equipment direction: " + climbingEquipment.getDirection());  
 fout.print("Scrolled down. Climbing equipment direction: " + climbingEquipment.getDirection() + "\n");  
 fout.flush();  
 }  
  
 public void resetScroller() {  
 climbingEquipment.resetScroller();  
 logToFile("Reset climber scroller. Climbing equipment direction: " + climbingEquipment.getDirection());  
 fout.print("Reset climber scroller. Climbing equipment direction: " + climbingEquipment.getDirection() + "\n");  
 fout.flush();  
 }  
  
 public ClimbingEquipment.Directions getScrollingDirection() {  
 logToFile("Requested climbing equipment scrolling direction: " + climbingEquipment.getDirection());  
 return climbingEquipment.getDirection();  
 }  
  
 public void dispose() {  
 fout.close();  
 actionLogWriter.close();  
 }  
}  
  
class ClimbingEquipment {  
 enum Directions {*NEUTRAL*, *UP*, *DOWN*};  
 private Directions direction;  
  
 public ClimbingEquipment() {  
 direction = Directions.*NEUTRAL*;  
 }  
  
 public void setUpDirection() {  
 direction = Directions.*UP*;  
 }  
  
 public void setNeutralDirection() {  
 direction = Directions.*NEUTRAL*;  
 }  
  
 public void setDownDirection() {  
 direction = Directions.*DOWN*;  
 }  
  
 public void resetScroller() {  
 setNeutralDirection();  
 }  
  
 public Directions getDirection() {  
 return direction;  
 }  
}  
  
class Position {  
 private int x, y;  
  
 public Position() {  
 x = 0;  
 y = 0;  
 }  
  
  
 public void setXPosition(int xPos) {  
 x = xPos;  
 }  
  
 public void setYPosition(int yPos) {  
 y = yPos;  
 }  
}  
  
class Carabiner {  
 private int btnResource;  
  
 public Carabiner() {  
 btnResource = 10;  
 }  
  
 public void clickButton() {  
 btnResource = btnResource - 1;  
 }  
  
 public int getButtonResource() {  
 return btnResource;  
 }  
  
}

package lab2;  
  
import java.io.IOException;  
  
class App {  
 public static void main(String[] args) throws IOException {  
 ClimbingEquipment.Directions dir = null;  
 Main climber = new Main();  
 climber.clickLeftButton();  
 System.*out*.print(climber.getLeftButtonResource() + "\n");  
 climber.clickRightButton();  
 System.*out*.print(climber.getRightButtonResource() + "\n");  
 climber.setPosition(5, 3);  
 climber.scrollDown();  
 climber.scrollUp();  
 dir = climber.getScrollingDirection();  
 if (dir == ClimbingEquipment.Directions.*DOWN*)  
 System.*out*.print("Down" + "\n");  
 else if (dir == ClimbingEquipment.Directions.*UP*)  
 System.*out*.print("Up" + "\n");  
 else  
 System.*out*.print("Neutral" + "\n");  
 climber.dispose();  
 }  
}