Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

З лабораторної роботи №3

Варіант – 24

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Спадкування та інтерфейси»

Виконав: ст. гр. КІ-301

Чоланюк О.Ю

Прийняв:

Майдан М.В.

Львів 2023

**Мета роботи:** ознайомитися з спадкуванням та інтерфейсами у мові Java.

**Теоретичні відомості**

# 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:

class Підклас extends Суперклас{

Додаткові поля і методи

}

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

При наслідувані у Java дозволяється перевизначення (перевантаження) методів та полів. При цьому область видимості методу, що перевизначається, має бути не меншою, ніж область видимості цього методу у суперкласі, інакше компілятор видасть повідомлення, про обмеження привілеїв доступу до даних. Перевизначення методу полягає у визначені у підкласі методу з сигнатурою методу суперкласу. При виклику такого методу з-під об’єкта підкласу викличеться метод цього підкласу. Якщо ж у підкласі немає визначеного методу, що викликається, то викличеться метод суперкласу. Якщо ж у суперкласі даний метод також відсутній, то згенерується повідомлення про помилку.

# Поліморфізм

Механізм поліморфізму забезпечує можливість присвоєння об’єктним змінним суперкласу об’єктів похідних класів та звертання з-під цих змінних до перевизначених у підкласі членів суперкласу. У Java всі об’єктні змінні є поліморфними. Поліморфізм реалізується за допомогою механізму динамічного (пізнього) зв’язування, який полягає у тому, що вибір методу, який необхідно викликати, відбувається не на етапі компіляції, а під час виконання програми.

# Абстрактні класи

Абстрактні класи призначені бути основою для розробки ієрархій класів та не дозволяють створювати об’єкти свого класу. Вони реалізуються за допомогою ключового слова abstract. На відміну від звичайних класів абстрактні класи можуть містити абстрактні методи (а можуть і не містити). *Абстрактні методи* – це методи, що оголошені з використанням ключового слова abstract і не місять тіла. Розширюючи абстрактний клас можна залишити деякі або всі методи невизначеними. При цьому підклас автоматично стане абстрактним. Перевизначення у підкласі усіх абстрактних методів призведе до того, що підклас не буде абстрактним, що дозволить створювати на його основі об’єкти класу. Синтаксис оголошення абстрактного класу наведено в пункті

«Класи та об’єкти». Синтаксис оголошення абстрактного методу:

[СпецифікаторДоступу] abstract Тип назваМетоду([параметри]);

# Інтерфейси

Інтерфейси вказують що повинен робити клас не вказуючи як саме він це повинен робити. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів. Синтаксис оголошення інтерфейсів:

[public] interface НазваІнтерфейсу

{

Прототипи методів та оголошення констант інтерфейсу

}

**Завдання**

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
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 військового альпініста

Файл:Main

package KZP.Lab3.Cholaniuk;

import java.io.FileWriter;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.Date;

public abstract class Main {

private String item;

private double weight;

private String description;

private double price;

private static final String LOG\_FILE\_NAME = "gear\_log.txt";

public Main(String itemName, double weight, String description, double price) {

this.item = itemName;

this.weight = weight;

this.description = description;

this.price = price;

}

public Main(String itemName, double weight, String description) {

this.item = itemName;

this.weight = weight;

this.description = description;

}

public Main(String itemName, double weight) {

this.item = itemName;

this.weight = weight;

}

public void setItemName(String itemName) {this.item = itemName;}

public String getItemName() {

return item;

}

public void setWeight(double weight) {

this.weight = weight;

}

public double getWeight() {

return weight;

}

public void setDescription(String description) {

this.description = description;

}

public String getDescription() {

return description;

}

public void setPrice(double price) {

this.price = price;

}

public double getPrice() {

return price;

}

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(LOG\_FILE\_NAME, true)) {

writer.write(logMessage + "\n");

} catch (IOException e) {

e.printStackTrace();

}

}

public void logItemToFile(String filename) {

FileWriter writer = null;

try {

writer = new FileWriter(filename, true);

writer.write("Назва: " + item + "\n");

writer.write("Вага: " + weight + " g\n");

writer.write("Ціна: $" + price + "\n");

writer.write("Опис: " + description + "\n\n");

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (writer != null) {

writer.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

logToFile("Зареєстрований елемент обладнання: " + item);

}

public void use() {

System.out.println("\nВи використовуєте предмет обладнання: " + item);

System.out.println(item + "Вага " + weight + " gram(s)");

System.out.println(item + " Ціна $" + price);

System.out.println(item + " Має такий опис: " + description);

logToFile("Елемент обладнання, що використовується: " + item);

}

public void inspect() {

System.out.println("\nОгляд обладнання:");

System.out.println("Назва: " + item);

System.out.println("Вага: " + weight + " g");

System.out.println("Ціна: $" + price);

System.out.println("Опис: " + description);

if (weight >= 2000) {

System.out.println("Це спорядження досить важке.");

} else if (weight < 2000 && weight > 500) {

System.out.println("Це спорядження не досить важке.");

} else {

System.out.println("Це легке спорядження, ідеальне для тривалих походів.");

}

if(price == 0){

System.out.println("Це обладнання ви отримали безкоштовно.");

}

else if (price > 600) {

System.out.println("Це обладнання досить дороге.");

} else if (price <= 600 && price > 200) {

System.out.println("Коштує таке обладнання не надто дорого.");

} else {

System.out.println("Це обладнання доступне за нижчою ціною.");

}

if (description.contains("захист від дощу")) {

System.out.println("Це обладнання чудово захищає від дощу.");

}

if (description.contains("для безпеки")) {

System.out.println("Це обладнання ідеально підходить для безпеки.");

}

if (description.contains("легко переносити")) {

System.out.println("Це компактне обладнання, яке зручно носити з собою.");

}

if (description.contains("високоякісні матеріали")) {

System.out.println("Дане обладнання відрізняється високою якістю і надійністю.");

}

logToFile("Оглянуте обладнання: " + item);

}

}

Файл:Military

package KZP.Lab3.Cholaniuk;

public interface Military {

int getCamoPattern();

void setCamoPattern(int camoPattern);

void specialMilitaryInspect();

}

Файл: MilitaryAlpinistsEquipment

package KZP.Lab3.Cholaniuk;

public class MilitaryAlpinistsEquipment extends Main implements Military {

private int camoPattern;

private boolean nightVision;

public MilitaryAlpinistsEquipment(String itemName, double weight, String description, double price, int camoPattern, boolean nightVision) {

super(itemName, weight, description, price);

this.camoPattern = camoPattern;

this.nightVision = nightVision;

}

@Override

public int getCamoPattern() {

return camoPattern;

}

@Override

public void setCamoPattern(int camoPattern) {

this.camoPattern = camoPattern;

}

public boolean hasNightVision() {

return nightVision;

}

public void setNightVision(boolean nightVision) {

this.nightVision = nightVision;

}

@Override

public void specialMilitaryInspect() {

super.inspect();

System.out.println("Виконання особливої військової функції з: " + getItemName());

if (hasNightVision()) {

System.out.println("Використання нічного бачення для спецоперації...");

} else {

System.out.println("Це обладнання не має можливості нічного бачення.");

}

if (getDescription().toLowerCase().contains("special")) {

System.out.println("Використовуючи особливі властивості цього обладнання...");

} else {

System.out.println("Це обладнання не має особливої властивості.");

}

}

}

Файл: MilitaryAlpinistsEquipmentApp

package KZP.Lab3.Cholaniuk;

public class MilitaryAlpinistsEquipmentApp {

public static void main(String[] args) {

MilitaryAlpinistsEquipment gear1 = new MilitaryAlpinistsEquipment("Тактичний ремінь", 1200,

"Якісна тактична упряж для військових альпіністів", 250, 3, true);

MilitaryAlpinistsEquipment gear2 = new MilitaryAlpinistsEquipment("Шолом для скелелазіння", 800,

"Міцний альпіністський шолом з кріпленням нічного бачення", 150, 2, true);

MilitaryAlpinistsEquipment gear3 = new MilitaryAlpinistsEquipment("Мотузкова драбина", 1500,

"Спеціальна мотузкова драбина для таємних операцій", 300, 1, false);

gear1.specialMilitaryInspect();

gear2.specialMilitaryInspect();

gear3.specialMilitaryInspect();

gear1.logItemToFile("military\_equipment\_log.txt");

gear2.logItemToFile("military\_equipment\_log.txt");

gear3.logItemToFile("military\_equipment\_log.txt");

}

}

