МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ПОЛТАВСЬКИЙ ПОЛІТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Циклова комісія дисциплін програмної інженерії

**ЗВІТ**

з навчальної практики

«Практика з об’єктно-орієнтованого програмування»

Виконав: здобувач освіти 3 курсу, групи 35 напрям підготовки: 121 «Інженерія програмного забезпечення» спеціалізації «Розробка програмного забезпечення»

Любченко О. В.

(прізвище та ініціали)

Керівник

Зелінська А. О.

(прізвище та ініціали)

(підпис)

Полтава – 2024

**ЗМІСТ**

[ВСТУП 4](#_Toc161816418)

[1. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №1 5](#_Toc161816419)

[1.1. Постановка завдання 5](#_Toc161816420)

[1.2. Код програми 5](#_Toc161816421)

[1.3. Результат роботи програми 7](#_Toc161816422)

[2. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №2 8](#_Toc161816423)

[2.1. Постановка завдання 8](#_Toc161816424)

[2.2. Код програми 8](#_Toc161816425)

[2.3. Результат роботи програми 10](#_Toc161816426)

[3. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №3 11](#_Toc161816427)

[3.1. Постановка завдання 11](#_Toc161816428)

[3.2. Код програми 11](#_Toc161816429)

[3.3. Результат роботи програми 13](#_Toc161816430)

[4. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №4 14](#_Toc161816431)

[4.1. Постановка завдання 14](#_Toc161816432)

[4.2. Код програми 14](#_Toc161816433)

[4.3. Результат роботи програми 15](#_Toc161816434)

[5. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №5 16](#_Toc161816435)

[5.1. Постановка завдання 16](#_Toc161816436)

[5.2. Код програми 16](#_Toc161816437)

[5.3. Результат роботи програми 17](#_Toc161816438)

[6. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №6 18](#_Toc161816439)

[6.1. Постановка завдання 18](#_Toc161816440)

[6.2. Код програми 18](#_Toc161816441)

[6.3. Результат роботи програми 20](#_Toc161816442)

[ВИСНОВКИ 22](#_Toc161816443)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 23](#_Toc161816444)

[ДОДАТОК А. РЕПОЗИТОРІЙ ПРОГРАМИ 24](#_Toc161816445)

# **ВСТУП**

Практика з об'єктно-орієнтованого програмування є частиною навчального процесу і сприяє саморозвитку студентів з метою розвитку професійних навичок та набуття студентами першого досвіду у сфері професійної діяльності. Метою даної практики є не тільки розвиток професійних навичок, а й підвищення майбутньої професійної кваліфікації. Практична підготовка дає можливість ознайомитися з обраною спеціалізацією та набути практичних навичок для виконання основних завдань за цією спеціалізацією, а також закріпити теоретичні знання та сформувати потребу в систематичному оновленні знань і творчому застосуванні їх на практиці.

Завданнями практики об'єктно-орієнтованого програмування є наступні:

* поглиблення та закріплення теоретичних знань з об'єктно-орієнтованого програмування, теорії алгоритмів та структур даних, технологій розробки програмного забезпечення та інших спеціалізованих областей;
* набуття практичних навичок об'єктно-орієнтованого програмування;
* вивчення основних інструментів об'єктно-орієнтованого програмування та їх застосування для розробки реальних програмних продуктів;
* аналіз об'єктно-орієнтованого програмного коду;
* практичні навички написання об'єктно-орієнтованого програмного коду;
* ознайомлення з життєвим циклом розробки програмного продукту;
* написання звіту – набору проектних документів, що відображають основні результати реалізації;
* розробка програмного продукту.

# **1. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №1**

## **1.1. Постановка завдання**

* розробити клас, що серіалізується, для зберігання параметрів і результатів обчислень;
* використовуючи агрегування, розробити клас для знаходження рішення задачі;
* розробити клас для демонстрації в діалоговому режимі збереження та відновлення стану об'єкта, використовуючи серіалізацію. Показати особливості використання transient полів;
* розробити клас для тестування коректності результатів обчислень та серіалізації/десеріалізації;
* виконати індивідуальне завдання (Визначити найбільше значення послідовно розташованих 1 у двійковому поданні цілісного значення потенційної енергії фізичного тіла при заданих значеннях маси та висоти).

## **1.2. Код програми**

|  |
| --- |
| **package ex01;**  **import java.io.BufferedReader;**  **import java.io.IOException;**  **import java.io.InputStreamReader;**  **public class Main {**  **private Calc calc = new Calc();**  **private void menu() {**  **String s = null;**  **BufferedReader in = new BufferedReader(new**  **InputStreamReader(System.in));**  **do {**  **do {**  **System.out.println("Enter command...");**  **System.out.print("'q'uit, 'v'iew, 'g'enerate, 's'ave,**  **'r'estore: ");**  **try {**  **s = in.readLine();**  **} catch (IOException e) {**  **System.out.println("Error: " + e);**  **System.exit(0);**  **}**  **} while (s.length() != 1);**  **switch (s.charAt(0)) {**  **case 'q':**  **System.out.println("Exit.");**  **break;**  **case 'v':**  **System.out.println("View current.");**  **calc.show();**  **break;**  **case 'g':**  **System.out.println("Random generation.");**  **calc.init(Math.random() \* 360.0, Math.random() \***  **100.0);**  **calc.show();**  **break;**  **case 's':**  **System.out.println("Save current.");**  **try {**  **calc.save();**  **} catch (IOException e) {**  **System.out.println("Serialization error: " +**  **e);**  **}**  **calc.show();**  **break;**  **case 'r':**  **System.out.println("Restore last saved.");**  **try {**  **calc.restore();**  **} catch (Exception e) {**  **System.out.println("Serialization error: " +**  **e);**  **}**  **calc.show();**  **break;**  **default:**  **System.out.print("Wrong command. ");**  **}**  **} while (s.charAt(0) != 'q');**  **}**  **public static void main(String[] args) {**  **Main main = new Main();**  **main.menu();**  **}**  **}** |

## **1.3. Результат роботи програми**

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рисунок 1.1 – Результат роботи програми

# **2. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №2**

## **2.1. Постановка завдання**

* як основа використовувати вихідний текст проекту попередньої лабораторної роботи. Забезпечити розміщення результатів обчислень у колекції з можливістю збереження/відновлення;
* використовуючи шаблон проектування Factory Method (Virtual Constructor), розробити ієрархію, що передбачає розширення рахунок додавання нових відображуваних класів;
* розширити ієрархію інтерфейсом "фабрикованих" об'єктів, що представляє набір методів для відображення результатів обчислень;
* реалізувати ці методи виведення результатів у текстовому виде;
* розробити та реалізувати інтерфейс для "фабрикуючого" методу.

## **2.2. Код програми**

|  |
| --- |
| **package ex03;**  **import java.io.BufferedReader;**  **import java.io.IOException;**  **import java.io.InputStreamReader;**  **public class Main {**  **private Calc calc = new Calc();**  **private void menu() {**  **String s = null;**  **BufferedReader in = new BufferedReader(new**  **InputStreamReader(System.in));**  **do {**  **do {**  **System.out.println("Enter command...");**  **System.out.print("'q'uit, 'v'iew, 'g'enerate, 's'ave,**  **'r'estore: ");**  **try {**  **s = in.readLine();**  **} catch (IOException e) {**  **System.out.println("Error: " + e);**  **System.exit(0);**  **}**  **} while (s.length() != 1);**  **switch (s.charAt(0)) {**  **case 'q':**  **System.out.println("Exit.");**  **break;**  **case 'v':**  **System.out.println("View current.");**  **Displayable displayable =**  **calc.createDisplayable();**  **displayable.display();**  **break;**  **case 'g':**  **System.out.println("Random generation.");**  **calc.init(Math.random() \* 360.0, Math.random() \***  **100.0);**  **calc.show();**  **break;**  **case 's':**  **System.out.println("Save current.");**  **try {**  **calc.save();**  **} catch (IOException e) {**  **System.out.println("Serialization error: " +**  **e);**  **}**  **calc.show();**  **break;**  **case 'r':**  **System.out.println("Restore last saved.");**  **try {**  **calc.restore();**  **} catch (Exception e) {**  **System.out.println("Serialization error: " +**  **e);**  **}**  **calc.show();**  **break;**  **default:**  **System.out.print("Wrong command. ");**  **}**  **} while (s.charAt(0) != 'q');**  **}**  **public static void main(String[] args) {**  **Main main = new Main();**  **main.menu();**  **}**  **}** |

## **2.3. Результат роботи програми**

Зображення, що містить текст, знімок екрана, Шрифт, документ

Автоматично згенерований опис

Рисунок 2.1 – Результат роботи програми

# **3. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №3**

## **3.1. Постановка завдання**

* за основу використовувати вихідний текст проекту попередньої лабораторної роботи Використовуючи шаблон проектування Factory Method (Virtual Constructor), розширити ієрархію похідними класами, реалізують методи для подання результатів у вигляді текстової таблиці. Параметри відображення таблиці мають визначатися користувачем;
* продемонструвати заміщення (перевизначення, overriding), поєднання (перевантаження, overloading), динамічне призначення методів (Пізнє зв'язування, поліморфізм, dynamic method dispatch);
* забезпечити діалоговий інтерфейс із користувачем;
* розробити клас для тестування основної функціональності.

## **3.2. Код програми**

|  |
| --- |
| **package ex04;**  **import java.io.BufferedReader;**  **import java.io.IOException;**  **import java.io.InputStreamReader;**  **public class Main {**  **public static void main(String[] args) {**  **Calc calc = new Calc();**  **BufferedReader in = new BufferedReader(new**  **InputStreamReader(System.in));**  **String input;**  **try {**  **do {**  **System.out.println("Choose an option:");**  **System.out.println("1. Initialize and view");**  **System.out.println("2. Save and restore");**  **System.out.println("3. Exit");**  **input = in.readLine();**  **switch (input) {**  **case "1":**  **System.out.println("Enter mass:");**  **double mass =**  **Double.parseDouble(in.readLine());**  **System.out.println("Enter height:");**  **double height =**  **Double.parseDouble(in.readLine());**  **calc.init(mass, height);**  **System.out.println("Enter display format**  **(default or table):");**  **String format = in.readLine();**  **calc.show(format);**  **break;**  **case "2":**  **System.out.println("Saving...");**  **calc.save();**  **System.out.println("Restoring...");**  **try {**  **calc.restore();**  **} catch (Exception e) {**  **System.out.println("Error restoring: " +**  **e.getMessage());**  **}**  **calc.show("default");**  **break;**  **case "3":**  **System.out.println("Exiting...");**  **break;**  **default:**  **System.out.println("Invalid option");**  **break;**  **}**  **} while (!input.equals("3"));**  **} catch (IOException e) {**  **e.printStackTrace();**  **}**  **}**  **}** |

## **3.3. Результат роботи програми**

Зображення, що містить текст, знімок екрана, Шрифт, документ

Автоматично згенерований опис

Рисунок 3.1 – Результат роботи програми

# **4. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №4**

## **4.1. Постановка завдання**

* реалізувати можливість скасування (undo) операцій (команд);
* продемонструвати поняття "макрокоманда";
* при розробці програми використовувати шаблон Singletone;
* забезпечити діалоговий інтерфейс із користувачем;
* розробити клас для тестування функціональності програми.

## **4.2. Код програми**

|  |
| --- |
| **package ex05;**  **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner scanner = new Scanner(System.in);**  **CommandManager commandManager = new CommandManager();**  **MacroCommand macroCommand = new MacroCommand();**  **ColorCommand colorCommand = new ColorCommand();**  **boolean running = true;**  **while (running) {**  **System.out.println("Enter 'generate' to generate a color,**  **'undo' to undo last operation, 'exit' to exit:");**  **String input = scanner.nextLine();**  **switch (input) {**  **case "generate":**  **commandManager.executeCommand(colorCommand);**  **macroCommand.addCommand(colorCommand);**  **break;**  **case "undo":**  **commandManager.undoLastCommand();**  **break;**  **case "exit":**  **running = false;**  **break;**  **default:**  **System.out.println("Invalid command.");**  **break;**  **}**  **}**  **scanner.close();**  **}** |

## **4.3. Результат роботи програми**

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рисунок 4.1 – Результат роботи програми

# **5. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №5**

## **5.1. Постановка завдання**

* продемонструвати можливість паралельної обробки елементів колекції (пошук мінімуму, максимуму, обчислення середнього значення, відбір за критерієм, статистична обробка тощо);
* управління чергою завдань (команд) реалізувати за допомогою шаблону Worker Thread.

## **5.2. Код програми**

|  |
| --- |
| **package ex06;**  **import java.util.ArrayList;**  **import java.util.Collections;**  **import java.util.List;**  **import java.util.concurrent.\*;**  **public class Main {**  **public static void main(String[] args) {**  **List<Integer> data = new ArrayList<>();**  **for (int i = 0; i <= 100; i++) {**  **data.add(i);**  **}**  **ExecutorService executor = Executors.newFixedThreadPool(4);**  **Future<Integer> minFuture = executor.submit(() ->**  **Collections.min(data));**  **Future<Integer> maxFuture = executor.submit(() ->**  **Collections.max(data));**  **Future<Double> avgFuture = executor.submit(() ->**  **data.stream().mapToInt(Integer::intValue).average().orElse(0)**  **);**  **Future<Integer> sumFuture = executor.submit(() ->**  **data.stream().filter(x -> x >**  **50).mapToInt(Integer::intValue).sum());**  **try {**  **int min = minFuture.get();**  **int max = maxFuture.get();**  **double avg = avgFuture.get();**  **int sum = sumFuture.get();**  **System.out.println("Мінімум: " + min);**  **System.out.println("Максимум: " + max);**  **System.out.println("Середнє: " + avg);**  **System.out.println("Сума елементів більших за 50: " + sum);**  **} catch (InterruptedException | ExecutionException e) {**  **e.printStackTrace();**  **}**  **executor.shutdown();**  **}**  **}** |

## **5.3. Результат роботи програми**

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рисунок 5.1 – Результат роботи програми

# **6. ІНДИВІДУАЛЬНЕ ЗАВДАННЯ №6**

## **6.1. Постановка завдання**

* розробити ієрархію класів відповідно до шаблону Observer (java) та продемонструвати можливість обслуговування розробленої раніше колекції (об'єкт, що спостерігається, Observable) різними (не менше двох) спостерігачами (Observers) – відстеження змін, упорядкування, висновок, відображення і т.д;
* при реалізації ієрархії класів використати інструкції (Annotation) відзначити особливості різних політик утримання анотацій (annotation retention policies). Продемонструвати підтримку класів концепції рефлексії (Reflection);
* використовуючи раніше створені класи, розробити додаток, що відображає результати обробки колекції об'єктів у графічному вигляді;
* забезпечити діалоговий інтерфейс з користувачем та перемальовування графіка під час зміни значень елементів колекції.

## **6.2. Код програми**

|  |
| --- |
| **package ex07;**  **import org.knowm.xchart.\*;**  **import org.knowm.xchart.style.Styler;**  **import javax.swing.\*;**  **import java.awt.\*;**  **import java.util.ArrayList;**  **import java.util.List;**  **import java.util.Observable;**  **import java.util.Observer;**  **import java.util.Scanner;**  **class ChartObserver implements Observer {**  **private XYChart chart;**  **private JFrame chartPanel;**  **Collection data = new Collection();**  **public ChartObserver() {**  **chart = new**  **XYChartBuilder().width(500).height(300).title("Обрахунок**  **енергії").xAxisTitle("Обрахунок").yAxisTitle("Енергія").buil**  **d();**  **chart.getStyler().setLegendPosition(Styler.LegendPosition.In**  **sideNW);**  **chart.addSeries("Результат", new double[] {0}, new double[]**  **{0});**  **data.addObserver(this);**  **chartPanel = new SwingWrapper(chart).displayChart();**  **}**    **public void addPoint(double point){**  **data.addData(point);**  **}**    **@Override**  **public void update(java.util.Observable o, Object arg) {**  **System.out.println("UPDATE");**  **List<Double> xData = new ArrayList<Double>();**  **for (double i = 0; i < data.getData().size(); i++) {**  **xData.add(i);**  **}**  **chart.updateXYSeries("Результат", xData, data.getData(),**  **null);**  **chartPanel.repaint();**    **}**  **}**  **public class Test {**  **public static void main(String[] args) {**  **ChartObserver chartObserver = new ChartObserver();**    **while(true){**  **EnergyCalculator calc = new EnergyCalculator();**  **Scanner in = new Scanner(System.in);**  **System.out.println("enter mass:");**  **double mass = Double.parseDouble(in.nextLine());**  **System.out.println("enter height:");**  **double height = Double.parseDouble(in.nextLine());**  **var res = calc.calculateEnergy(mass, height);**  **System.out.println("result: " + res.getEnergy());**  **chartObserver.addPoint(res.getEnergy());**  **}**  **}**  **}**  **class Collection extends Observable{**  **private List<Double> data = new ArrayList<>();**    **Collection(){**    **}**    **public void addData(double data){**  **this.data.add(data);**  **super.setChanged();**  **super.notifyObservers();**  **}**    **public List<Double> getData(){**  **return data;**  **}**  **}** |

## **6.3. Результат роботи програми**

Зображення, що містить текст, знімок екрана

Автоматично згенерований опис

Рисунок 6.1 – Результат роботи програми

Зображення, що містить текст, знімок екрана, Графік, ряд

Автоматично згенерований опис

Рисунок 6.2 – Результат виведення графіка

# **ВИСНОВКИ**

В ході практики з об'єктно-орієнтованого програмування було досягнуто наступних цілей:

* поглиблення та закріплення теоретичних знань з об'єктно-орієнтованого програмування, теорії алгоритмів та структур даних, а також технологій розробки програмного забезпечення;
* набуття практичних навичок у сфері об'єктно-орієнтованого програмування шляхом розробки реальних програмних продуктів;
* оволодіння основними інструментами об'єктно-орієнтованого програмування та їх застосування для ефективної розробки програм;
* вміння аналізувати об'єктно-орієнтований програмний код з метою виявлення його сильних та слабких сторін;
* практичні навички написання об'єктно-орієнтованого програмного коду відповідно до вимог проекту;
* ознайомлення з життєвим циклом розробки програмного продукту, включаючи етапи від аналізу вимог до підтримки та післярелізного обслуговування;
* навички написання звіту, що включає в себе проектні документи, що відображають основні результати реалізації проекту;
* реалізація програмного продукту з використанням отриманих знань і навичок з об'єктно-орієнтованого програмування.

Ця практика сприяла вдосконаленню професійних компетенцій та підготовці до подальшої кар'єрної діяльності в сфері розробки програмного забезпечення.

# **СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. МЕТОДИЧНІ ВКАЗІВКИ ДО ПРОХОДЖЕННЯ ПРАКТИКИ З «ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ» ДЛЯ ЗДОБУВАЧІВ ОСВІТИ НАПРЯМУ ПІДГОТОВКИ 121 «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ» СПЕЦІАЛІЗАЦІЇ «РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ» [Текст]: МЕТОДИЧНІ ВКАЗІВКИ / Відокремлений структурний підрозділ «Полтавський політехнічний фаховий коледж Національного технічного університету «Харківський політехнічний інститут»; [уклад.: В.В.Олійник, А.О.Зелінська]. – Полтава: ВСП «ППФК НТУ «ХПІ», 2022. – 27 с.
2. Oracle Docs: Java Documentation – Get Started. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/ (дата звернення: 07.03.2024).
3. Stack Overflow: Empowering the world to develop technology through collective knowledge – Stack Overflow. [Електронний ресурс] – Режим доступу до ресурсу: https://try.stackoverflow.co/ (дата звернення: 05.03.2024).
4. GitHub: Let’s build from here. GitHub. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ (дата звернення: 07.03.2024).
5. Grunitzky G. Java Programming: Java. – Нью-Йорк, 2017. – 46 с.
6. Lassoff M. Java Programming for Beginners: Learn the fundamentals of programming with Java. – Бірмінгем, 2017. – 318 с.
7. McBride P. K. Java Made Simple (Made Simple Programming). 2-ге вид. – Лондон, 2002. – 208 с.
8. Norton W. Java Programming for Beginners: A Piratical Beginners Guide to Learn Programming, Fundamentals and Code. – Нью-Йорк, 2020. – 145 с.

# **ДОДАТОК А. РЕПОЗИТОРІЙ ПРОГРАМИ**

Репозиторій з кодом програми доступний на веб-сервісі GitHub за посиланням, що зображене на рисунку А.1.



Рисунок А.1 – Посилання на репозиторій