

Міністерство освіти і науки України
Державний університет „Житомирська політехніка”

Кафедра ІПЗ та ВТ
Група: ВТ-22-1

Звіти з лабораторних робіт з Java

Виконав: Лупашина А. А.
Прийняв: Піонтківський В. І.

					ІРТР.420001.123-ЗЛ			
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з Лабораторних робіт Java	Літ.	Арк.	Аркушів
Розроб.		Лупашина А.А.					1	5
Перевір.		Піонтківський В.І.						
Керівник						ФІКТ, гр. ВТ-22-1		
Н. контр.								
Затверд.								

Лабораторна работа №1

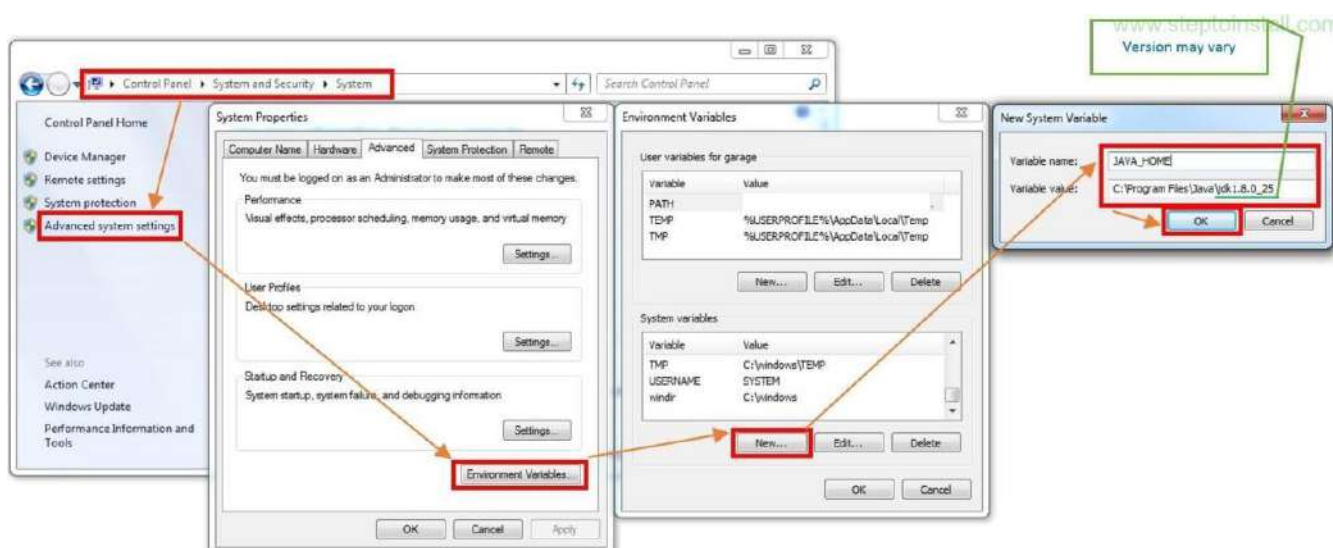
Тема: Знайомство з мовою програмування Java.

Написання простих програм на мові програмування Java

Мета роботи: встановити IDE IntelliJ IDEA; створити репозиторій на GitLab; вивчити реалізацію базових алгоритмічних конструкцій у мові програмування Java; знайомство з правилами оформлення програмного коду

Завдання 1. Встановлення і налаштування JDK:

<https://www.oracle.com/java/technologies/javase/javase8u211-later-archiveddownloads.html> створити змінну JAVA_HOME та додати її в PATH (%JAVA_HOME%\bin)



Завдання 2. Встановлення та налаштування програмного середовища для веб-розробки за даним посиланням:

<https://www.jetbrains.com/idea/download/#section=windows>

Завдання 4. Написання простих програм:

Програма 1

Ім'я класу: com.education.ztu.Task1

Напишіть клас, який реалізує функціональність відображення рядка «Hello, World!!!» у консолі.

Скрін всього проєкту:


```

// Запит користувача на введення першого цілого числа
System.out.println("Введіть перше ціле число:");
int number1 = scanner.nextInt(); // Зчитування першого цілого числа з консолі

// Запит користувача на введення другого цілого числа
System.out.println("Введіть друге ціле число:");
int number2 = scanner.nextInt(); // Зчитування другого цілого числа з консолі

// Обчислення суми двох чисел
int sum = number1 + number2; // Додавання обох чисел і збереження результату в
змінній sum

// Виведення результату на екран
System.out.println("Сума: " + sum); // Виведення суми двох чисел
}
}

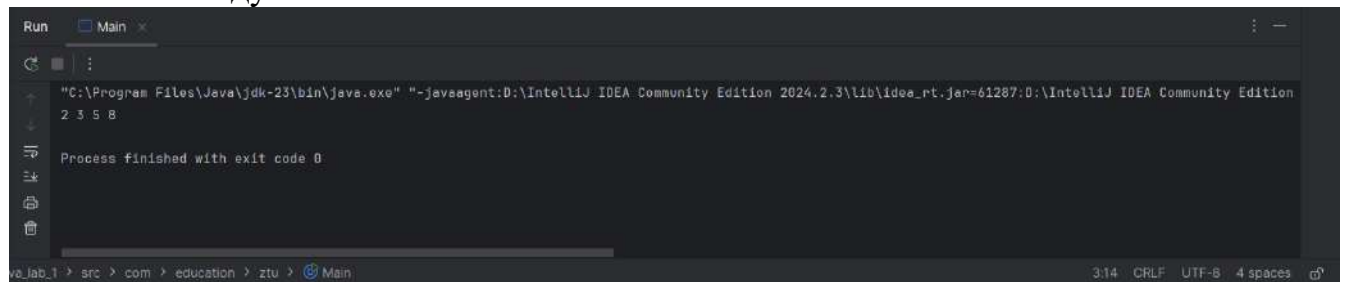
```

Програма 3

Ім'я класу: com.education.ztu.Task3

Напишіть клас, який реалізує функціональність відображення параметрів командного рядка в консолі (відображення через пробіл між ними), результат не повинен закінчуватися пробілом. Аргументи передавати таким чином Task3.main(new String[]{"2", "3", "5", "8"}); в класі Main.

Виконання коду:



Лістинг коду:

```

package com.education.ztu;

public class Main {
    public static void main(String[] args) {
        // Виклик методу main з класу Task3, передаючи йому масив рядків як аргументи
        Task3.main(new String[]{"2", "3", "5", "8"}); // Передача масиву строк з числовими
значеннями
    }
}

```

```

package com.education.ztu;

public class Task3 {
    public static void main(String[] args) {
        // Перевірка, чи є аргументи командного рядка
        if (args.length > 0) {
            StringBuilder result = new StringBuilder();
            // Цикл для проходження через всі аргументи
            for (int i = 0; i < args.length; i++) {
                result.append(args[i]); // Додавання аргументу до результату
                // Додавання пробілу між аргументами, якщо це не останній аргумент
                if (i < args.length - 1) {

```

					ІПТР.420001.123-ЗЛ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        result.append(" ");
    }
}
// Виведення зібраного рядка на екран
System.out.println(result.toString()); // Виведення результату
}
}
}

```

Програма 4

Ім'я класу: com.education.ztu.Task4

Напишіть клас, який реалізує функціональні можливості визначення найбільшого спільного дільника двох цілих додатних чисел. Для зчитування даних використовувати методи класу Scanner.

Виконання коду:

```

Run Task4 x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=61358:D:\IntelliJ IDEA Community Edition
Введіть перше число:
1
Введіть друге число:
3
Найбільший спільний дільник: 1
Process finished with exit code 0

```

Лістинг коду:

```

package com.education.ztu;

import java.util.Scanner;

public class Task4 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Введіть перше число:");
        int num1 = scanner.nextInt();

        System.out.println("Введіть друге число:");
        int num2 = scanner.nextInt();

        // Обчислення і виведення найбільшого спільного дільника
        System.out.println("Найбільший спільний дільник: " + gcd(num1, num2)); // Виклик
методу gcd і виведення результату
    }

    // Метод для обчислення найбільшого спільного дільника (НСД) двох чисел
    public static int gcd(int a, int b) {
        // Використання алгоритму Евкліда для обчислення НСД
        while (b != 0) { // Цикл, поки b не дорівнює 0
            int temp = b; // Зберігаємо значення b в тимчасову змінну
            b = a % b; // Обчислюємо нове значення b
            a = temp; // Присвоюємо a значення тимчасової змінної
        }
        return a; // Повертаємо найбільший спільний дільник
    }
}

```

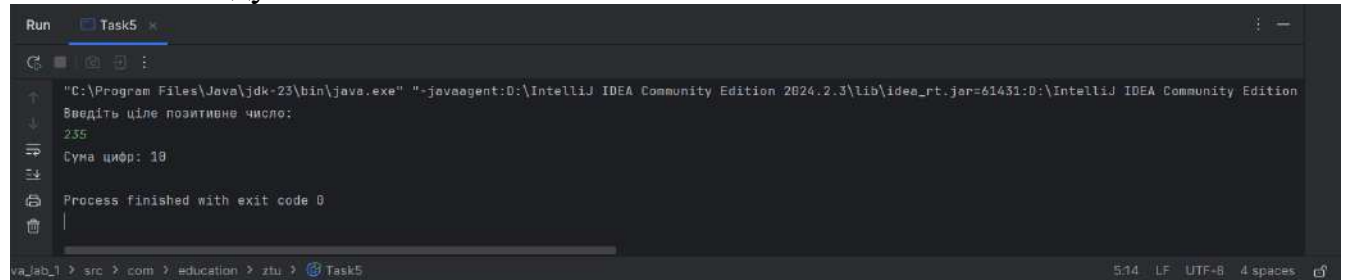
Програма 5

					ІПТР.420001.123-ЗЛ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

Ім'я класу: com.education.ztu.Task5

Напишіть клас, який реалізує функціональні можливості визначення суми цифр цілого позитивного числа. Для зчитування даних використовувати методи класу Scanner.

Виконання коду:



Лістинг коду:

```
package com.education.ztu;

import java.util.Scanner;

public class Task5 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Введіть ціле позитивне число:");
        int num = scanner.nextInt();

        // Обчислення і виведення суми цифр введенного числа
        System.out.println("Сума цифр: " + sumOfDigits(num)); // Виклик методу sumOfDigits і виведення результату
    }

    // Метод для обчислення суми цифр числа
    public static int sumOfDigits(int num) {
        int sum = 0; // Ініціалізація змінної для зберігання суми цифр
        while (num != 0) { // Цикл, поки num не дорівнює 0
            sum += num % 10; // Додавання останньої цифри до суми
            num /= 10; // Вилучення останньої цифри з числа
        }
        return sum; // Повернення суми цифр
    }
}
```

Програма 6

Ім'я класу: com.education.ztu.Task6

Напишіть клас, який створює масив із n елементів і заповнює його зростаючою послідовністю чисел Фібоначчі (1,1,2,3,5,8...). Створити новий масив та заповнити його зворотньою послідовністю Фібоначчі. Вивести в консоль обидва масиви. Для зчитування даних використовувати методи класу Scanner

Виконання коду:

					ІПТР.420001.123-ЗЛ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Run Task6 x
"С:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=61631:D:\IntelliJ IDEA Community Edition
Введіть кількість елементів в масиві:
7
Масив Фібоначчі:
1 1 2 3 5 8 13
Зворотній масив Фібоначчі:
13 8 5 3 2 1 1
Process finished with exit code 0
va.lah.1 > src > com > education > ztu > Task6
5:14 LF UTF-8 4 spaces

```

Лістинг коду:

```

package com.education.ztu;

import java.util.Scanner;

public class Task6 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Введіть кількість елементів в масиві:");
        int n = scanner.nextInt(); // Зчитування кількості елементів

        int[] fibonacci = new int[n]; // Ініціалізація масиву для чисел Фібоначчі
        int[] reverseFibonacci = new int[n]; // Ініціалізація масиву для зворотних чисел
        Фібоначчі

        // Заповнення масиву чисел Фібоначчі
        fibonacci[0] = 1; // Перший елемент Фібоначчі
        if (n > 1) { // Перевірка, чи масив має більше одного елемента
            fibonacci[1] = 1; // Другий елемент Фібоначчі
        }
        // Обчислення наступних елементів масиву Фібоначчі
        for (int i = 2; i < n; i++) {
            fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2]; // Сума двох попередніх
            елементів
        }

        // Заповнення зворотнього масиву Фібоначчі
        for (int i = 0; i < n; i++) {
            reverseFibonacci[i] = fibonacci[n - 1 - i]; // Копіювання елементів у
            зворотному порядку
        }

        // Виведення масиву Фібоначчі
        System.out.println("Масив Фібоначчі:");
        for (int i : fibonacci) { // Цикл для перебору елементів масиву Фібоначчі
            System.out.print(i + " "); // Виведення елемента
        }
        System.out.println(); // Перехід на новий рядок

        // Виведення зворотнього масиву Фібоначчі
        System.out.println("Зворотній масив Фібоначчі:");
        for (int i : reverseFibonacci) { // Цикл для перебору елементів зворотнього масиву
            System.out.print(i + " "); // Виведення елемента
        }
    }
}

```

Програма 7

Ім'я класу: com.education.ztu.Task7

					ІПТР.420001.123-ЗЛ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Створити масив символів латинського алфавіту та вести їх числові коди в такому форматі:

A ==> 65

B ==> 66

C ==> 67

Виконання коду:



```
Run Task7 x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=61789:D:\IntelliJ IDEA Community Edition
A ==> 65
B ==> 66
C ==> 67
D ==> 68
E ==> 69
F ==> 70
G ==> 71
H ==> 72
I ==> 73
J ==> 74
K ==> 75
L ==> 76
M ==> 77
N ==> 78
O ==> 79
P ==> 80
Q ==> 81
R ==> 82
S ==> 83
T ==> 84
U ==> 85
V ==> 86
W ==> 87
X ==> 88
Y ==> 89
Z ==> 90
```

Лістинг коду:

```
package com.education.ztu;

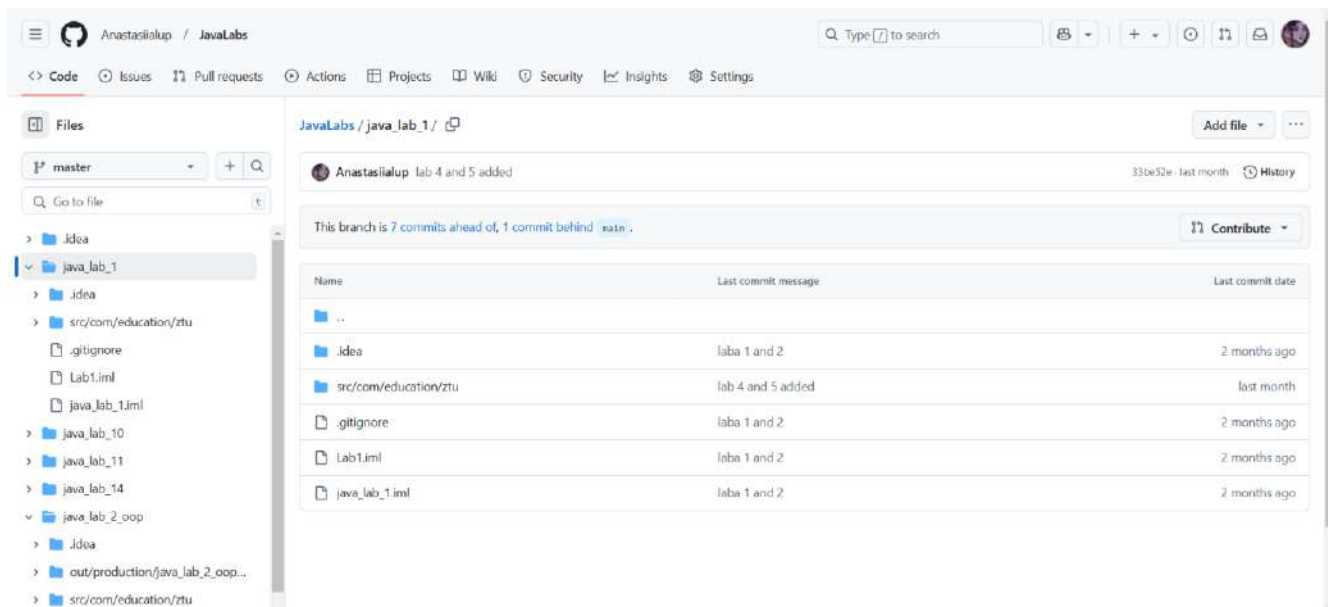
public class Task7 {
    public static void main(String[] args) {
        char[] alphabet = new char[26]; // Ініціалізація масиву символів для латинського алфавіту (26 букв)

        // Заповнення масиву символів латинського алфавіту
        for (int i = 0; i < 26; i++) { // Цикл для перебору всіх букв алфавіту
            alphabet[i] = (char) ('A' + i); // Присвоєння символу, починаючи з 'A'
        }

        // Виведення символів та їх числових кодів
        for (char letter : alphabet) { // Цикл для перебору кожної букви в масиві
            System.out.println(letter + " ==> " + (int) letter); // Виведення букви та її ASCII коду
        }
    }
}
```

Завдання 5. Створити в GitLab проект Java_labs_ztu, створити директорію Lab_1 та запустити в Lab_1 виконану лабораторну роботу. Надати доступ для перевірки викладачу.

					ІПТР.420001.123-ЗЛ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		



Висновок:

Під час лабораторної роботи №1 було освоєно основи програмування на Java, налаштування JDK та IntelliJ IDEA. Розроблено сім простих програм, які демонструють базові алгоритмічні конструкції, роботу з класом `Scanner`, аргументами командного рядка, числами Фібоначчі та ASCII-кодами. Проект завантажено на GitLab із належним оформленням. Лабораторна робота сприяла розумінню синтаксису Java та правил оформлення коду.

					ІПТР.420001.123-ЗЛ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №2

Тема: Створення структури класу заданої предметної області.

Мета роботи: створити ієрархію класів заданої предметної області, робота з статичними методами.

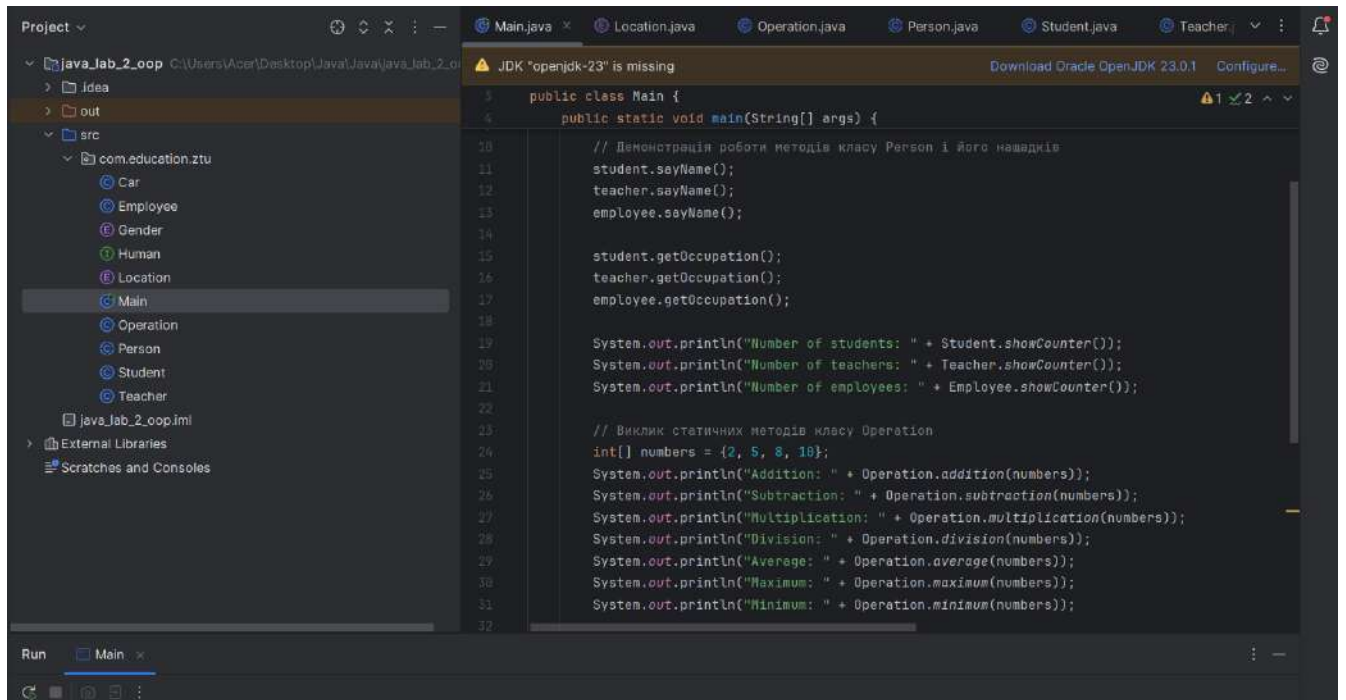
Завдання 1. Створити консольний Java проект java_lab_2_oop з пакетом com.education.ztu

Завдання 2. Створити ієрархію класів відповідно до UML діаграми:

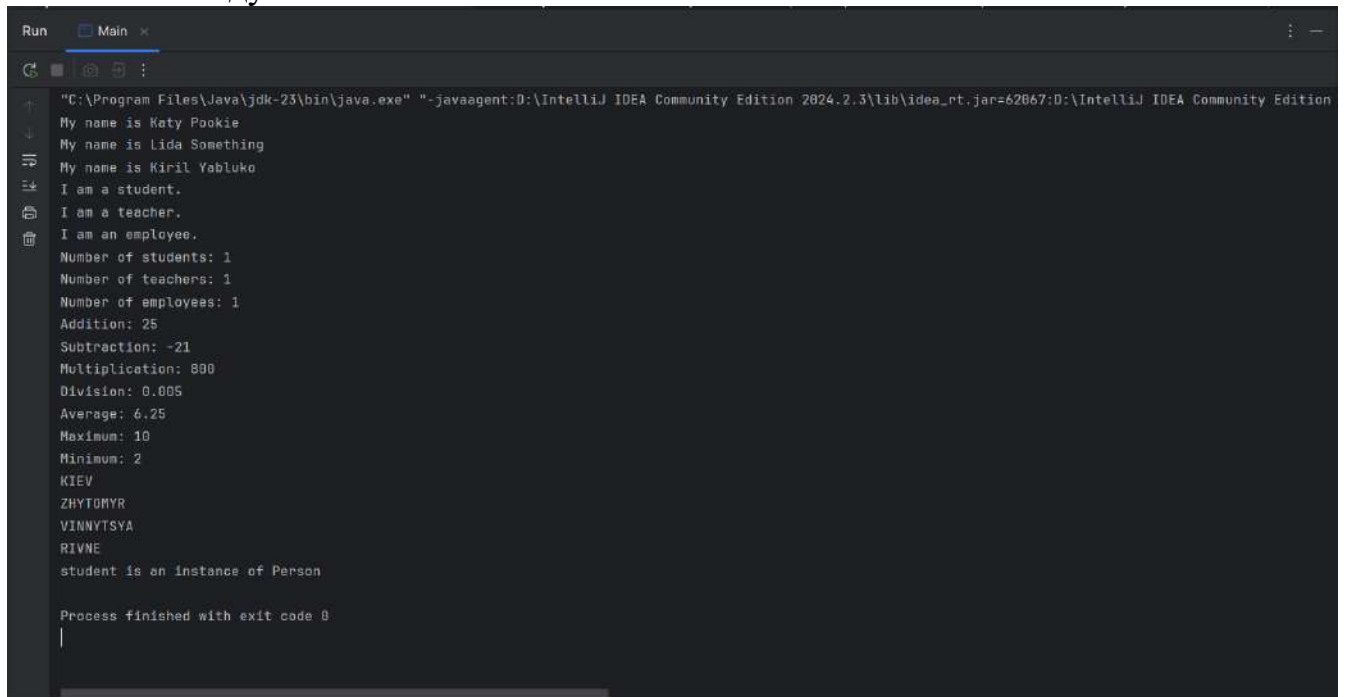
- поля класів повинні бути приховані модифікаторами доступу private, protected;
- створити конструктор без аргументів та з аргументами;
- створити блок ініціалізації, в якому ініціалізуються значення полів за замовчуванням у разі, якщо викликається конструктор без аргументів;
- створити геттери та сеттери для полів;
- створити статичну змінну counter для підрахунку створених екземплярів даного класу та статичний метод showCounter для відображення значення змінної counter.
- створити enum Location та Gender і використати їх в полях класів.
- створити інтерфейс Human з методами sayFullName, sayAge, sayLocation, sayGender та whoIAm (default)
- створити абстрактний клас Person з абстрактним методом getOccupation та звичайним методом getFullInfo, що імплементує Human;
- створити класу Student, Teacher, Employee, що наслідують Person та перевизначити необхідні методи та створити свої.
- для Teacher, Employee додати поле Car, що є об'єктом відповідного класу.
- створити в Car внутрішній клас Engine з методами startEngine, stopEngine, isEngineWorks та реалізувати їх логіку.
- додати до описаної функціональності свою (нові поля та методи).
- в методі main класу Main створити об'єкти відповідних класів та продемонструвати роботу їх методів.
- продемонструвати роботу оператора instanceof.

Проект:

					ІПТР.420001.123-ЗЛ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		



Виконання коду:



Лістинг коду:

- створити в Car внутрішній клас Engine з методами startEngine, stopEngine, isEngineWorks та реалізувати їх логіку:

```
package com.education.ztu;

public class Car {
    private String brand;
    private Engine engine;

    public Car(String brand) {
        this.brand = brand;
        this.engine = new Engine();
    }
}
```

									Арк.
									11
Змн.	Арк.	№ докум.	Підпис	Дата	ІПТР.420001.123-3Л				

```

    public boolean engineIsRunning() {
        return engine.engineWorks;
    }

    class Engine {
        private boolean engineWorks = false;

        public void startEngine() {
            engineWorks = true;
            System.out.println("Engine started.");
        }

        public void stopEngine() {
            engineWorks = false;
            System.out.println("Engine stopped.");
        }
    }
}

```

- створити enum Location та Gender і використати їх в полях класів:

```

package com.education.ztu;

public enum Gender {
    MALE, FEMALE
}

```

```

package com.education.ztu;

public enum Location {
    KIEV, ZHYTOMYR, VINNYTSYA, RIVNE
}

```

- створити інтерфейс Human з методами sayFullName, sayAge, sayLocation, sayGender та whoIAm (default):

```

package com.education.ztu;

public interface Human {
    void sayAge();
    void sayGender();
    void sayLocation();
    void sayName();

    default void whoAmI() {
        System.out.println("I am a human.");
    }
}

```

- створити абстрактний клас Person з абстрактним методом getOccupation та звичайним методом getFullInfo, що імплементує Human:

```

package com.education.ztu;

public abstract class Person implements Human {
    private int age;
    private String firstname;
    private String lastname;
    private Gender gender;
    private Location location;
    protected String fullInfo;

    private static int counter = 0; // статичний лічильник створених екземплярів
}

```

					ІПТР.420001.123-ЗЛ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

```

// Конструктор за замовчуванням
public Person() {
    this.firstname = "Default";
    this.lastname = "Name";
    this.age = 0;
    this.gender = Gender.MALE;
    this.location = Location.KIEV;
    counter++;
}

// Конструктор з аргументами
public Person(String firstname, String lastname, int age, Gender gender, Location
location) {
    this.firstname = firstname;
    this.lastname = lastname;
    this.age = age;
    this.gender = gender;
    this.location = location;
    counter++;
}

// Геттери та сеттери
public String getFirstname() {
    return firstname;
}

public void setFirstname(String firstname) {
    this.firstname = firstname;
}

public String getLastName() {
    return lastname;
}

public void setLastName(String lastname) {
    this.lastname = lastname;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public Gender getGender() {
    return gender;
}

public void setGender(Gender gender) {
    this.gender = gender;
}

public Location getLocation() {
    return location;
}

public void setLocation(Location location) {

```

```

        this.location = location;
    }

    public static int showCounter() {
        return counter;
    }

    // Методи з інтерфейсу Human
    @Override
    public void sayAge() {
        System.out.println("I am " + age + " years old.");
    }

    @Override
    public void sayGender() {
        System.out.println("I am " + gender);
    }

    @Override
    public void sayLocation() {
        System.out.println("I live in " + location);
    }

    @Override
    public void sayName() {
        System.out.println("My name is " + firstname + " " + lastname);
    }

    // Абстрактний метод, який треба буде перевизначити в похідних класах
    public abstract void getOccupation();
}

```

- створити класи Student, Teacher, Employee, що наслідують Person та перевизначити необхідні методи та створити свої:

```

package com.education.ztu;

public class Student extends Person {
    private int course;
    private String speciality;
    private String university;

    private static int counter = 0;

    public Student(String firstname, String lastname, int age, Gender gender, Location
location, String speciality, String university, int course) {
        super(firstname, lastname, age, gender, location);
        this.speciality = speciality;
        this.university = university;
        this.course = course;
        counter++;
    }

    @Override
    public void getOccupation() {
        System.out.println("I am a student.");
    }

    public static int showCounter() {
        return counter;
    }
}

```

```
}  
}
```

```
package com.education.ztu;  
  
public class Teacher extends Person {  
    private String subject;  
    private String university;  
    private Car car;  
  
    private static int counter = 0;  
  
    public Teacher(String firstname, String lastname, int age, Gender gender, Location  
location, String subject, String university, Car car) {  
        super(firstname, lastname, age, gender, location);  
        this.subject = subject;  
        this.university = university;  
        this.car = car;  
        counter++;  
    }  
  
    @Override  
    public void getOccupation() {  
        System.out.println("I am a teacher.");  
    }  
  
    public static int showCounter() {  
        return counter;  
    }  
}
```

```
package com.education.ztu;  
  
public class Employee extends Person {  
    private String company;  
    private String position;  
    private Car car;  
  
    private static int counter = 0;  
  
    public Employee(String firstname, String lastname, int age, Gender gender, Location  
location, String company, String position, Car car) {  
        super(firstname, lastname, age, gender, location);  
        this.company = company;  
        this.position = position;  
        this.car = car;  
        counter++;  
    }  
  
    @Override  
    public void getOccupation() {  
        System.out.println("I am an employee.");  
    }  
  
    public static int showCounter() {  
        return counter;  
    }  
}
```

					IPTP.420001.123-3Л	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

- створити в Car внутрішній клас Engine з методами startEngine, stopEngine, isEngineWorks та реалізувати їх логіку:

```
package com.education.ztu;

public class Car {
    private String brand;
    private Engine engine;

    public Car(String brand) {
        this.brand = brand;
        this.engine = new Engine();
    }

    public boolean engineIsRunning() {
        return engine.engineWorks;
    }

    class Engine {
        private boolean engineWorks = false;

        public void startEngine() {
            engineWorks = true;
            System.out.println("Engine started.");
        }

        public void stopEngine() {
            engineWorks = false;
            System.out.println("Engine stopped.");
        }
    }
}
```

- в методі main класу Main створити об'єкти відповідних класів та продемонструвати роботу їх методів:

```
package com.education.ztu;

public class Main {
    public static void main(String[] args) {
        // Створюємо об'єкти класів Person, Student, Teacher, Employee
        Student student = new Student("Katy", "Pookie", 20, Gender.MALE, Location.KIEV,
"Computer Science", "ZTU", 2);
        Teacher teacher = new Teacher("Lida", "Something", 40, Gender.FEMALE,
Location.VINNYTSYA, "Mathematics", "ZTU", new Car("Toyota"));
        Employee employee = new Employee("Kiril", "Yabluko", 35, Gender.MALE,
Location.ZHYTOMYR, "Tech Corp", "Developer", new Car("BMW"));

        // Демонстрація роботи методів класу Person і його нащадків
        student.sayName();
        teacher.sayName();
        employee.sayName();

        student.getOccupation();
        teacher.getOccupation();
        employee.getOccupation();

        System.out.println("Number of students: " + Student.showCounter());
        System.out.println("Number of teachers: " + Teacher.showCounter());
        System.out.println("Number of employees: " + Employee.showCounter());
    }
}
```



```

// Виклик статичних методів класу Operation
int[] numbers = {2, 5, 8, 10};
System.out.println("Addition: " + Operation.addition(numbers));
System.out.println("Subtraction: " + Operation.subtraction(numbers));
System.out.println("Multiplication: " + Operation.multiplication(numbers));
System.out.println("Division: " + Operation.division(numbers));
System.out.println("Average: " + Operation.average(numbers));
System.out.println("Maximum: " + Operation.maximum(numbers));
System.out.println("Minimum: " + Operation.minimum(numbers));

// Демонстрація використання enum Location
for (Location location : Location.values()) {
    System.out.println(location);
}

// Перевірка оператора instanceof
if (student instanceof Person) {
    System.out.println("student is an instance of Person");
}
}

```

Завдання 3. Створити клас Operation з статичними методами addition, subtraction, multiplication, division, average, maximum, minimum, що приймають необмежену кількість аргументів через varargs.

в методі main класу Main2 продемонструвати роботу методів класу Operation

- вивести всі значення енам Location.

Лістинг коду:

```

package com.education.ztu;

public class Operation {
    // Статичний метод для додавання
    public static int addition(int... numbers) {
        int sum = 0;
        for (int number : numbers) {
            sum += number;
        }
        return sum;
    }

    // Статичний метод для віднімання
    public static int subtraction(int... numbers) {
        if (numbers.length == 0) return 0;
        int result = numbers[0];
        for (int i = 1; i < numbers.length; i++) {
            result -= numbers[i];
        }
        return result;
    }

    // Статичний метод для множення
    public static int multiplication(int... numbers) {
        int result = 1;
        for (int number : numbers) {
            result *= number;
        }
    }
}

```

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

```

        return result;
    }

    // Статичний метод для ділення
    public static float division(int... numbers) {
        if (numbers.length == 0 || numbers[0] == 0) return 0;
        float result = (float) numbers[0];
        for (int i = 1; i < numbers.length; i++) {
            if (numbers[i] != 0) {
                result /= numbers[i];
            } else {
                System.out.println("Division by zero is not allowed!");
                return 0;
            }
        }
        return result;
    }

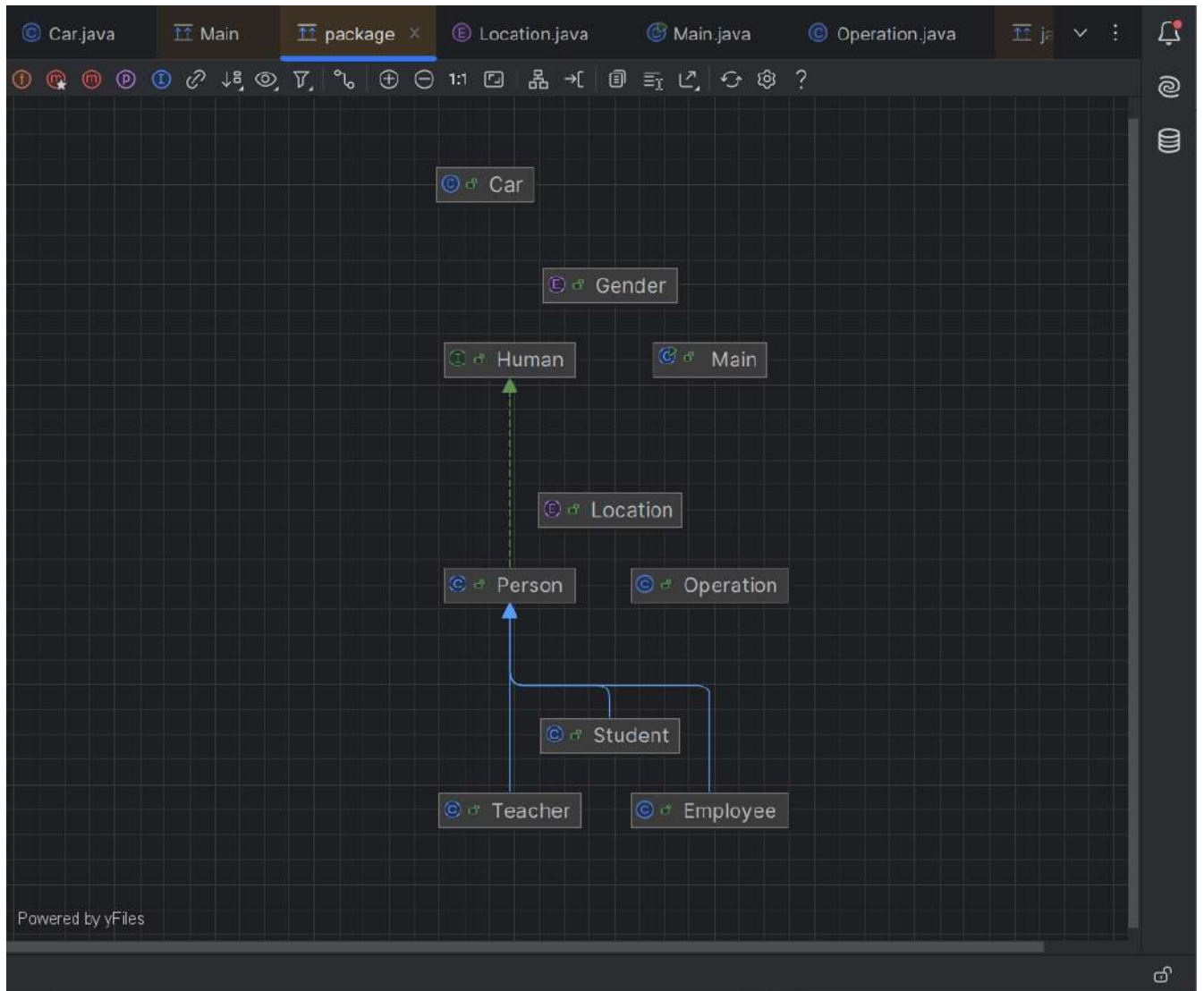
    // Статичний метод для обчислення середнього значення
    public static float average(int... numbers) {
        if (numbers.length == 0) return 0;
        return (float) addition(numbers) / numbers.length;
    }

    // Статичний метод для знаходження максимального числа
    public static int maximum(int... numbers) {
        if (numbers.length == 0) return Integer.MIN_VALUE;
        int max = numbers[0];
        for (int number : numbers) {
            if (number > max) {
                max = number;
            }
        }
        return max;
    }

    // Статичний метод для знаходження мінімального числа
    public static int minimum(int... numbers) {
        if (numbers.length == 0) return Integer.MAX_VALUE;
        int min = numbers[0];
        for (int number : numbers) {
            if (number < min) {
                min = number;
            }
        }
        return min;
    }
}

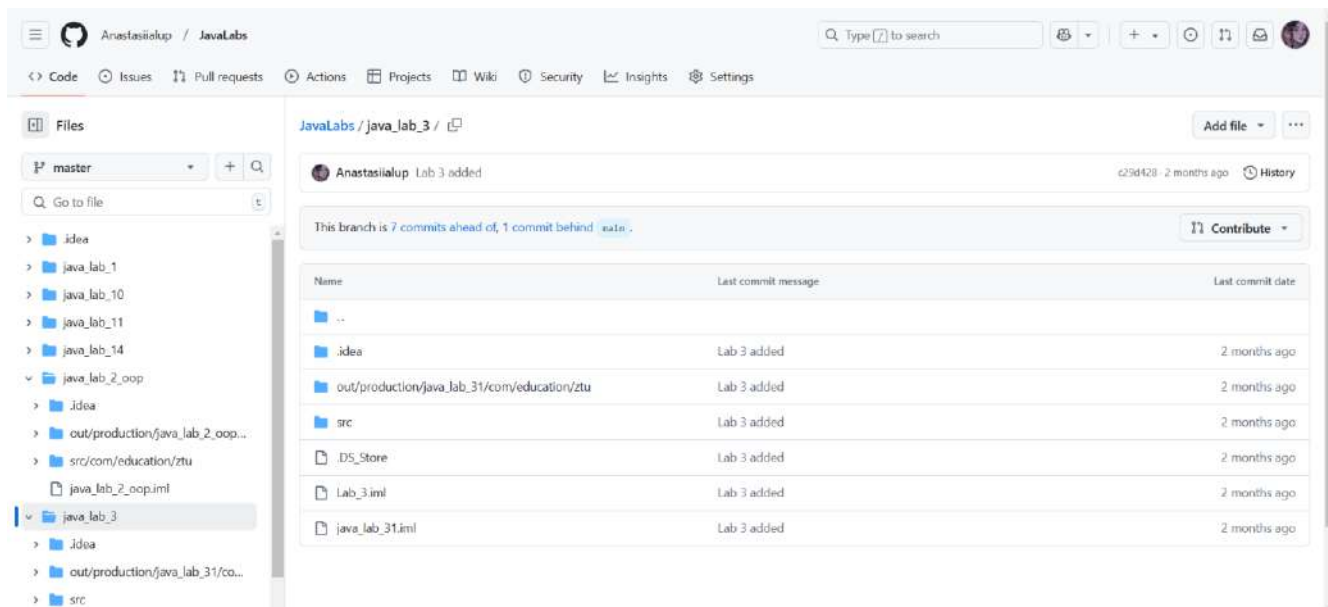
```

Завдання 4. Створити UML діаграму створеної структури ієрархії класів та зберегти як картинку.



Завдання 5. В GitLab проекті Java_labs_ztu, створити директорію Lab_2 та запустити в Lab_2 виконану лабораторну роботу. Надати доступ для перевірки викладачу.

					ІПТР.420001.123-ЗЛ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		



Висновок:

Під час виконання лабораторної роботи №2 було створено структуру класів відповідно до предметної області та UML-діаграми. Реалізовано принципи об'єктно-орієнтованого програмування: інкапсуляцію, наслідування, поліморфізм та використання інтерфейсів.

Сформовано ієрархію класів із конструкторами, полями, методами, геттерами, сеттерами та статичними змінними.

Реалізовано внутрішній клас Engine у класі Car з відповідною логікою.

Створено та продемонстровано роботу статичних методів у класі Operation для математичних операцій.

Використано модифікатори доступу, еnumератори Location та Gender, а також ключове слово instanceof.

Згенеровано UML-діаграму для відображення створеної структури.

Результати підтвердили практичну користь і глибше розуміння принципів ООП.

					ІПТР.420001.123-ЗЛ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №3

Тема: Використання узагальнень (generics). Клонування та порівняння об'єктів.

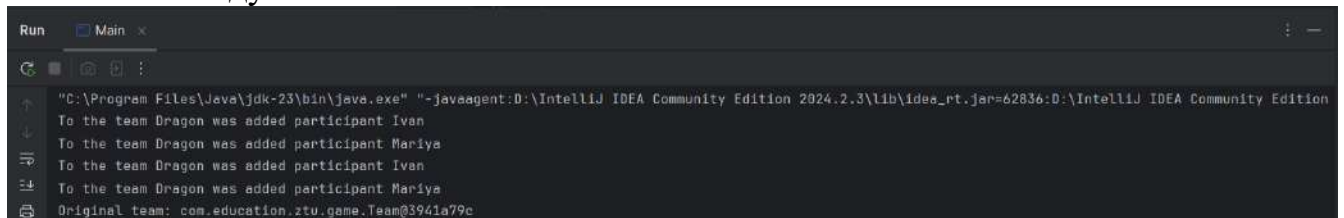
Мета роботи: створити міні проект Game з використанням узагальнень, клонування та порівняння об'єктів.

Завдання 1. Відкрити заготовлений проект з реалізованою базовою функціональністю.

Завдання 2. За допомогою узагальнень (generics) встановити такі обмеження:

- до команди можна додавати тільки учасників, що відносяться до одної ліги (Scholar, Student або Employee).
- грати між собою можуть тільки команди з учасниками одної ліги (тобто команда студентів може грати тільки іншою командою студентів).
- продемонструвати створення команд, гравців, додавання гравців до команд, гри між ними.

Виконання коду:



```
Run Main x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=62836:D:\IntelliJ IDEA Community Edition
To the team Dragon was added participant Ivan
To the team Dragon was added participant Mariya
To the team Dragon was added participant Ivan
To the team Dragon was added participant Mariya
Original team: com.education.ztu.game.Team@3941a79c
```

Лістинг коду:

```
package com.education.ztu;

import com.education.ztu.game.*;

public class Main {
    public static void main(String[] args) {
        try {
            // Створення учасників
            Scholar scholar1 = new Scholar("Ivan", 13);
            Scholar scholar2 = new Scholar("Mariya", 15);
            Team<Scholar> scollarTeam = new Team<>("Dragon");
            scollarTeam.addNewParticipant(scholar1);
            scollarTeam.addNewParticipant(scholar2);

            // Клонування
            Team<Scholar> clonedTeam = Team.deepClone(scollarTeam);
            System.out.println("Original team: " + scollarTeam);
            System.out.println("Cloned team: " + clonedTeam);

            // Зміна в оригінальній команді
            scholar1.setName("IvanUpdated");
            System.out.println("After updating the original team:");
            System.out.println("Original team: " + scollarTeam);
            System.out.println("Cloned team remains unchanged: " + clonedTeam);

        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
        }
    }
}
```

					IPTP.420001.123-3Л	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 3. Клонування:

- для класу Participant імплементувати інтерфейс Cloneable та перевизначити метод clone.
- для класу Participant перевизначити методи hashCode та equals.
- для класу Participant та його підкласів перевизначити метод toString.
- для класу Team Реалізувати глибоке клонування через статичний метод або конструктор копіювання.
- продемонструвати клонування та використання методів hashCode, equals та toString.

Виконання коду:

```
Original team: com.education.ztu.game.Team@3941a79c
Cloned team: com.education.ztu.game.Team@506e1b77
After updating the original team:
Original team: com.education.ztu.game.Team@3941a79c
Cloned team remains unchanged: com.education.ztu.game.Team@506e1b77

Process finished with exit code 0
```

Лістинг коду:

```
package com.education.ztu;

import com.education.ztu.game.*;

public class Main {
    public static void main(String[] args) {
        try {
            // Створення учасників
            Schoolar scholar1 = new Schoolar("Ivan", 13);
            Schoolar scholar2 = new Schoolar("Mariya", 15);
            Team<Schoolar> scollarTeam = new Team<>("Dragon");
            scollarTeam.addNewParticipant(scholar1);
            scollarTeam.addNewParticipant(scholar2);

            // Клонування
            Team<Schoolar> clonedTeam = Team.deepClone(scollarTeam);
            System.out.println("Original team: " + scollarTeam);
            System.out.println("Cloned team: " + clonedTeam);

            // Зміна в оригінальній команді
            scholar1.setName("IvanUpdated");
            System.out.println("After updating the original team:");
            System.out.println("Original team: " + scollarTeam);
            System.out.println("Cloned team remains unchanged: " + clonedTeam);

        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
        }
    }
}
```

Завдання 4. Порівняння:

- для класу Participant імплементувати інтерфейс Comparable та перевизначити метод compareTo для сортування учасників по імені.
- створити Comparator для порівняння учасників по віку.
- *створити компаратор з пріоритетом використовуючи можливості

					IPTP.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		22

Java 8 (спочатку порівняння по імені, а потім по віку).

- продемонструвати роботу порівнянь на прикладі сортування учасників команд.

Виконання коду:

```
Run Main2 x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=62967:0:\IntelliJ IDEA Community Edition
Before sorting:
SchoolarParticipant{name='Ivan', age=13}
SchoolarParticipant{name='Mariya', age=15}
StudentParticipant{name='Mykola', age=20}
StudentParticipant{name='Viktoria', age=21}
EmployeeParticipant{name='Andriy', age=28}
EmployeeParticipant{name='Oksana', age=25}

After sorting by age:
SchoolarParticipant{name='Ivan', age=13}
SchoolarParticipant{name='Mariya', age=15}
StudentParticipant{name='Mykola', age=20}
StudentParticipant{name='Viktoria', age=21}
EmployeeParticipant{name='Oksana', age=25}
EmployeeParticipant{name='Andriy', age=28}

Process finished with exit code 0
```

Лістинг коду:

```
package com.education.ztu;

import com.education.ztu.game.*;
import com.education.ztu.game.AgeComparator; // Імпорт класу AgeComparator

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Main2 {
    public static void main(String[] args) {
        // Створення учасників
        Schoolar schoolar1 = new Schoolar("Ivan", 13);
        Schoolar schoolar2 = new Schoolar("Mariya", 15);
        Student student1 = new Student("Mykola", 20);
        Student student2 = new Student("Viktoria", 21);
        Employee employee1 = new Employee("Andriy", 28);
        Employee employee2 = new Employee("Oksana", 25);

        // Додавання учасників до списку
        List<Participant> participants = new ArrayList<>();
        participants.add(schoolar1);
        participants.add(schoolar2);
        participants.add(student1);
        participants.add(student2);
        participants.add(employee1);
        participants.add(employee2);

        // Вивід учасників до сортування
        System.out.println("Before sorting:");
        for (Participant participant : participants) {
            System.out.println(participant);
        }

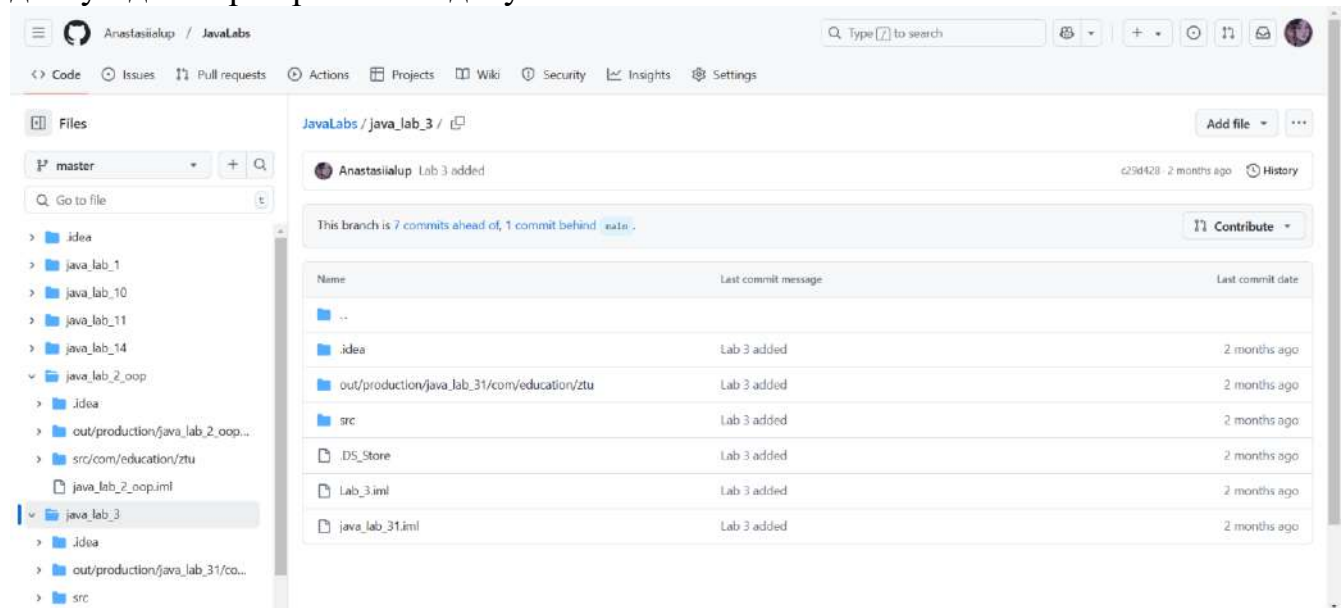
        // Сортування за віком
        Collections.sort(participants, new AgeComparator());
    }
}
```

```

// Вивід учасників після сортування
System.out.println("\nAfter sorting by age:");
for (Participant participant : participants) {
    System.out.println(participant);
}
}
}

```

Завдання 5. В GitLab проекті Java_labs_ztu, створити директорію Lab_3 та запустити в Lab_3 виконану лабораторну роботу. Надати доступ для перевірки викладачу.



Висновок:

У лабораторній роботі №3 реалізовано міні-проект Game з використанням узагальнень, клонування та порівняння об'єктів.

Generics: Додано обмеження для команд і учасників, матчі між командами однієї ліги.

Клонування: Реалізовано Cloneable, глибоке копіювання, перевизначено hashCode, equals, toString.

Порівняння: Додано Comparable та Comparator для сортування за іменем, віком і кількома критеріями.

Робота допомогла засвоїти generics, клонування й сортування об'єктів у Java.

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

Лабораторна робота №4

Тема: Класи String, StringBuffer та StringBuilder. Локалізація та інтернаціоналізація. Робота з датами.

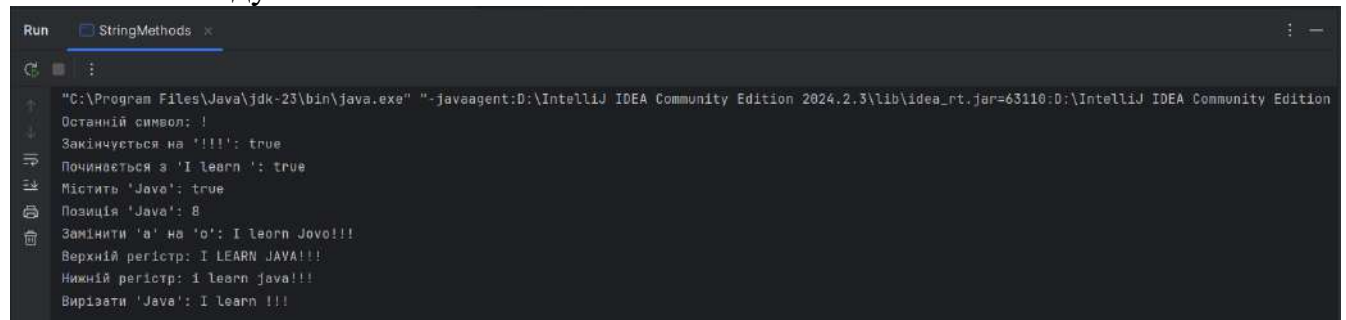
Мета роботи: робота з класами String, StringBuffer, StringBuilder та їх методами; практика використання локалізації та інтернаціоналізації; робота з датами.

Завдання 1. Створити консольний Java проект java_lab_4 з пакетом com.education.ztu

Завдання 2. Практика методів класу String:

- Напишіть метод, який приймає як параметр будь-який рядок, наприклад "I learn Java!!!".
- Роздрукувати останній символ рядка.
- Перевірити, чи закінчується ваш рядок підрядком "!!!".
- Перевірити, чи починається ваш рядок підрядком "I learn ".
- Перевірити, чи містить ваш рядок підрядком "Java".
- Знайти позицію підрядка "Java" у рядку "I learn Java!!!".
- Замінити всі символи "a" на "o".
- Перетворіть рядок на верхній регістр.
- Перетворіть рядок на нижній регістр.
- Вирізати рядок Java.

Виконання коду:



Лістинг коду:

```
package com.education.ztu;

public class StringMethods {

    // Метод, що приймає рядок і виконує зазначені операції
    public static void processString(String input) {
        // Роздрукувати останній символ рядка
        System.out.println("Останній символ: " + input.charAt(input.length() - 1));

        // Перевірити, чи закінчується рядок підрядком "!!!"
        System.out.println("Закінчується на '!!!': " + input.endsWith("!!!"));

        // Перевірити, чи починається рядок підрядком "I learn "
        System.out.println("Починається з 'I learn ': " + input.startsWith("I learn "));

        // Перевірити, чи містить рядок підрядком "Java"
        System.out.println("Містить 'Java': " + input.contains("Java"));
    }
}
```

```

// Знайти позицію підрядка "Java" у рядку
System.out.println("Позиція 'Java': " + input.indexOf("Java"));

// Замінити всі символи "a" на "o"
String replacedString = input.replace('a', 'o');
System.out.println("Замінити 'a' на 'o': " + replacedString);

// Перетворити рядок на верхній регістр
System.out.println("Верхній регістр: " + input.toUpperCase());

// Перетворити рядок на нижній регістр
System.out.println("Нижній регістр: " + input.toLowerCase());

// Вирізати рядок "Java"
int start = input.indexOf("Java");
if (start != -1) {
    String cutString = input.substring(0, start) + input.substring(start + 4);
    System.out.println("Вирізати 'Java': " + cutString);
} else {
    System.out.println("'Java' не знайдено для вирізання.");
}
}

public static void main(String[] args) {
    String text = "I learn Java!!!";
    processString(text);
}
}

```

Завдання 3. Створити рядок за допомогою класу `StringBuilder` або `StringBuffer` та його методів:

- Дано два числа, наприклад, 4 і 36, необхідно скласти наступні рядки:

4 + 36 = 40

4 - 36 = -32

4 * 36 = 144

- Використати метод `StringBuilder.append()`.

- Замініть символ “=” на слово “рівно”. Використати методи `StringBuilder.insert()`, `StringBuilder.deleteCharAt()`.

- Замініть символ “=” на слово “рівно”. Використати метод `StringBuilder.replace()`.

- Змінити послідовність розташування символів в рядку на протилежну. Використати метод `StringBuilder.reverse()`.

- Визначити довжину та `capacity`.

Виконання коду:


```
// Заміна "=" на "рівно" за допомогою replace()
while ((indexOfEquals = sb.indexOf("=")) != -1) {
    sb.replace(indexOfEquals, indexOfEquals + 1, "рівно");
}
System.out.println("Заміна '=' на 'рівно' (використовуючи replace):");
System.out.println(sb.toString());

// Перевертання рядка
sb.reverse();
System.out.println("Перевернутий рядок:");
System.out.println(sb.toString());

// Визначення довжини та ємності
System.out.println("Довжина рядка: " + sb.length());
System.out.println("Capacity: " + sb.capacity());
}
```

Завдання 4. Вивести у форматovanому вигляді чек з купленими товарами використовуючи можливості класу `Formatter`: Дата та час покупки: 28.03.2019 13:25:12

```
=====
№ Товар Категорія Ціна
=====
1. Джинси Жіночий одяг 1500,78 ₴
2. Спідниця Жіночий одяг 1000,56 ₴
3. Краватка Чоловічий одяг 500,78 ₴
=====
Разом: 3002,34 ₴
Доповнити список товарів до 10 шт.
Виконання коду:
```



Лістинг коду:

```
package com.education.ztu;

import java.util.Formatter;
```

```

public class CheckFormatter {
    public static void main(String[] args) {
        Formatter formatter = new Formatter();

        // Дата та час покупки
        String date = "28.03.2019";
        String time = "13:25:12";

        // Форматування заголовка
        formatter.format("Дата та час покупки: %s %s\n", date, time);
        formatter.format("=====\n");
        formatter.format("%-2s %-10s %-15s %-10s\n", "№", "Товар", "Категорія", "Ціна");
        formatter.format("=====\n");

        // Форматування рядків для кожного товару
        Object[][] items = {
            {1, "Джинси", "Жіночий одяг", 1500.78},
            {2, "Спідниця", "Жіночий одяг", 1000.56},
            {3, "Краватка", "Чоловічий одяг", 500.78},
            {4, "Сорочка", "Чоловічий одяг", 800.22},
            {5, "Футболка", "Чоловічий одяг", 300.10},
            {6, "Сукня", "Жіночий одяг", 2000.00},
            {7, "Куртка", "Жіночий одяг", 2700.50},
            {8, "Шкарпетки", "Чоловічий одяг", 50.75},
            {9, "Капелюх", "Чоловічий одяг", 400.00},
            {10, "Шарф", "Чоловічий одяг", 150.30}
        };

        double total = 0;
        for (Object[] item : items) {
            formatter.format("%-2d %-10s %-15s %-10.2f €\n", item[0], item[1], item[2],
item[3]);
            total += (double) item[3];
        }

        // Виведення загальної суми
        formatter.format("=====\n");
        formatter.format("Разом: %.2f €\n", total);

        // Виведення результату на екран
        System.out.println(formatter);

        // Закриття formatter
        formatter.close();
    }
}

```

Завдання 5. Реалізувати інтернаціоналізацію для відображення чеку з товарами українською, англійською та будь-якою третьою мовою на ваш вибір. Для цього використати класи `Locale` та `ResourceBundle`. Для виведення валюти країни використати можливості класу `NumberFormat`.

- створити директорію `resources` в корені проекту та позначити її як директорію з ресурсами.
- створити три файли з розширенням `properties` для кожної локалі (наприклад: `data_ua_UA`) та заповнити даними (для кирилиці використати `escape` послідовності).

					ІПТР.420001.123-ЗЛ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

- об'єднати їх у Resource Bundle
- реалізувати функціонал отримання та роботи з даними для кожної локалі.

Виконання коду:

```
Run InternationalizedCheck x
C:\Program Files\Java\jdk-23\bin\java.exe "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=63280:D:\IntelliJ IDEA Community Edition
Date and time of purchase: 23.12.2024 03:59:46
=====
Num Item      Category      Price
=====
1  Jeans      Жіночий одяг  $1,500.78
2  Skirt       Жіночий одяг  $1,000.56
3  Tie         Чоловічий одяг $500.78
=====
Total: $3,002.12
```

Лістинг коду:

```
package com.education.ztu;

import java.text.NumberFormat;
import java.text.SimpleDateFormat;
import java.util.*;

public class InternationalizedCheck {
    public static void main(String[] args) {
        Locale enLocale = new Locale("en", "US");
        Locale frLocale = new Locale("fr", "FR");

        // Вибираємо локаль (спробуйте спочатку англійську, потім французьку)
        Locale selectedLocale = enLocale; // Змініть на frLocale для тесту французької
        локалі

        ResourceBundle bundle = ResourceBundle.getBundle("data", selectedLocale);

        // Виведення заголовка
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss");
        System.out.println(bundle.getString("date") + ": " + dateFormat.format(new
Date()));
        System.out.println("=====");
        System.out.printf("%-3s %-10s %-15s %-10s\n", bundle.getString("item_number"),
bundle.getString("item_name"), bundle.getString("item_category"),
bundle.getString("item_price"));
        System.out.println("=====");

        // Товари
        Object[][] items = {
            {1, bundle.getString("jeans"), "Жіночий одяг", 1500.78},
            {2, bundle.getString("skirt"), "Жіночий одяг", 1000.56},
            {3, bundle.getString("tie"), "Чоловічий одяг", 500.78}
        };

        double total = 0;
        NumberFormat currencyFormat = NumberFormat.getCurrencyInstance(selectedLocale);

        for (Object[] item : items) {
            System.out.printf("%-3d %-10s %-15s %-10s\n", item[0], item[1], item[2],
currencyFormat.format((double) item[3]));
            total += (double) item[3];
        }
    }
}
```

					ІРТР.420001.123-ЗЛ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        // Виведення загальної суми
        System.out.println("=====");
        System.out.println(bundle.getString("total") + ": " +
currencyFormat.format(total));
    }
}

```

Завдання 6. Робота з датами:

- Створіть об'єкт будь-якого класу для роботи з датами на власний вибір, вказуючи дату та час початку сьогоднішньої лабораторної з Java.
- Вивести на консоль день тижня, день у році, місяць, рік, години, хвилини, секунди.
- Перевірити чи рік високосний.
- Створіть об'єкт будь-якого класу для роботи з датами, який представляє поточний час.
- Порівняйте його з датою початку лабораторної з Java, використовуючи методи isAfter(), isBefore().
- Змініть значення елементів дати та часу на власний розсуд використовуючи методи обраного вами класу для роботи з датами.

Виконання коду:

```

Run DateExample x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=63322:0:\IntelliJ IDEA Community Edition
День тижня: MONDAY
День року: 309
Місяць: NOVEMBER
Рік: 2024
Години: 10
Хвилини: 0
Секунди: 0
Рік високосний: true
Поточна дата після початку лабораторної: true
Поточна дата перед початком лабораторної: false
Змінена дата та час: 06.11.2024 07:15:00
Process finished with exit code 0

```

Лістинг коду:

```

package com.education.ztu;

import java.time.*;
import java.time.format.DateTimeFormatter;

public class DateExample {
    public static void main(String[] args) {
        // Дата та час початку лабораторної
        LocalDateTime labStartTime = LocalDateTime.of(2024, Month.NOVEMBER, 4, 10, 0, 0);

        // Виведення дня тижня, дня року, місяця, року, годин, хвилин, секунд
        System.out.println("День тижня: " + labStartTime.getDayOfWeek());
        System.out.println("День року: " + labStartTime.getDayOfYear());
        System.out.println("Місяць: " + labStartTime.getMonth());
        System.out.println("Рік: " + labStartTime.getYear());
        System.out.println("Години: " + labStartTime.getHour());
        System.out.println("Хвилини: " + labStartTime.getMinute());
        System.out.println("Секунди: " + labStartTime.getSecond());
    }
}

```

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		31

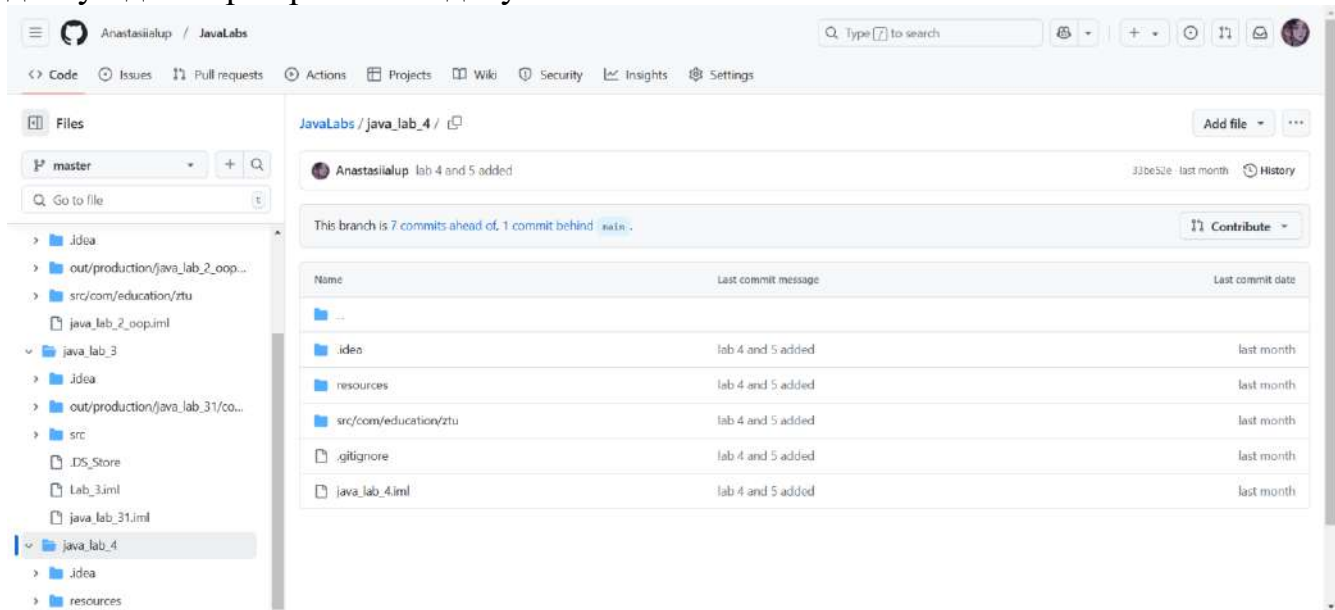

```
// Перевірка, чи рік високосний
System.out.println("Рік високосний: " + Year.isLeap(labStartTime.getYear()));

// Поточний час
LocalDateTime currentTime = LocalDateTime.now();

// Порівняння дат
System.out.println("Поточна дата після початку лабораторної: " +
currentTime.isAfter(labStartTime));
System.out.println("Поточна дата перед початком лабораторної: " +
currentTime.isBefore(labStartTime));

// Зміна значень дати та часу
LocalDateTime modifiedDateTime =
labStartTime.plusDays(2).minusHours(3).plusMinutes(15);
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss");
System.out.println("Змінена дата та час: " + modifiedDateTime.format(formatter));
}
```

Завдання 7. В GitLab проєкті Java_labs_ztu, створити директорію Lab_4 та запустити в Lab_4 виконану лабораторну роботу. Надати доступ для перевірки викладачу.



Висновок:

У лабораторній роботі №4:

Вивчено класи String, StringBuffer, StringBuilder та їх методи для роботи з текстом.

Реалізовано локалізацію та інтернаціоналізацію чека за допомогою Locale і ResourceBundle.

Виконано завдання з датами: порівняння, перевірка високості, зміна параметрів часу.

Робота поглибила знання роботи з текстами, датами та локалізацією у Java.

					<p style="text-align: center;"><i>ІПТР.420001.123-ЗЛ</i></p>	Арк.
						32
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

Лабораторна робота №5

Тема: Java Collections Framework

Мета роботи: робота з Java Collections Framework

Завдання 1. Створити консольний Java проект java_lab_5 з пакетом com.education.ztu

Завдання 2. Створити клас Product та задати йому поля та методи на власний вибір.

Лістинг коду:

```
package com.education.ztu;

public class Product {
    private String name;
    private double price;
    private String category;

    public Product(String name, double price, String category) {
        this.name = name;
        this.price = price;
        this.category = category;
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }

    public String getCategory() {
        return category;
    }

    @Override
    public String toString() {
        return "Product{" +
            "name='" + name + '\'' +
            ", price=" + price +
            ", category='" + category + '\'' +
            '}';
    }
}
```

Завдання 3. Створити динамічний масив, що містить об'єкти класу Product:

- Використовуємо клас ArrayList або LinkedList.
- Продемонструвати роботу з масивом використовуючи різні методи (add, addAll, get, indexOf, lastIndexOf, iterator, listIterator, remove, set, sort, subList, clear, contains, isEmpty, retainAll, size, toArray)

Виконання коду:

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		33


```

1500.0, "Electronics")));
System.out.println("Is empty? " + products.isEmpty());
System.out.println("Size: " + products.size());

// Перетворення на масив
Product[] productsArray = products.toArray(new Product[0]);

// Очищення
products.clear();
System.out.println("Products after clear: " + products);
}
}

```

Завдання 4. Створити чергу, що містить об'єкти класу Product:

- Використовуємо клас ArrayDeque.
- Продемонструвати роботу з чергою використовуючи методи (push, offerLast, getFirst, peekLast, pop, removeLast, pollLast та інші)

Виконання коду:

```

Run ProductQueueDemo
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=63574:D:\IntelliJ IDEA Community Edition
First element (getFirst): Product{name='Tablet', price=600.0, category='Electronics'}
Last element (peekLast): Product{name='Smartphone', price=800.0, category='Electronics'}
Removed first element (pop): Product{name='Tablet', price=600.0, category='Electronics'}
Removed last element (removeLast): Product{name='Smartphone', price=800.0, category='Electronics'}
Queue after adding another element: [Product{name='Laptop', price=1500.0, category='Electronics'}, Product{name='Headphones', price=150.0, category='Electron
Process finished with exit code 0

```

Лістинг коду:

```

package com.education.ztu;

import java.util.ArrayDeque;
import java.util.Deque;

public class ProductQueueDemo {
    public static void main(String[] args) {
        Deque<Product> productQueue = new ArrayDeque<>();

        // Додавання елементів
        productQueue.offerLast(new Product("Laptop", 1500.0, "Electronics"));
        productQueue.offerLast(new Product("Smartphone", 800.0, "Electronics"));
        productQueue.push(new Product("Tablet", 600.0, "Electronics"));

        // Отримання елементів
        System.out.println("First element (getFirst): " + productQueue.getFirst());
        System.out.println("Last element (peekLast): " + productQueue.peekLast());

        // Видалення елементів
        System.out.println("Removed first element (pop): " + productQueue.pop());
        System.out.println("Removed last element (removeLast): " +
productQueue.removeLast());

        // Додавання ще елементів і перевірка черги
        productQueue.offer(new Product("Headphones", 150.0, "Electronics"));
        System.out.println("Queue after adding another element: " + productQueue);
    }
}

```

Завдання 5. Створити множину, що містить об'єкти класу Product:

- Використовуємо клас TreeSet.
- Продемонструвати роботу з множиною використовуючи методи (add, first, last, headSet, subSet, tailSet, ceiling, floor, higher, lower, pollFirst, pollLast, descendingSet)

Виконання коду:

```
Run ProductSetDemo x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=63609:D:\IntelliJ IDEA Community Edition
First product: Product{name='Shoes', price=120.0, category='Fashion'}
Last product: Product{name='Laptop', price=1500.0, category='Electronics'}
Head set (< 800): [Product{name='Shoes', price=120.0, category='Fashion'}, Product{name='Watch', price=250.0, category='Accessories'}, Product{name='Tablet',
Sub set (120 - 800): [Product{name='Shoes', price=120.0, category='Fashion'}, Product{name='Watch', price=250.0, category='Accessories'}, Product{name='Table
Tail set (>= 600): [Product{name='Tablet', price=600.0, category='Electronics'}, Product{name='Smartphone', price=800.0, category='Electronics'}, Product{name='
Ceiling (>= 700): Product{name='Smartphone', price=800.0, category='Electronics'}
Floor (<= 1000): Product{name='Smartphone', price=800.0, category='Electronics'}
Higher (> 800): Product{name='Laptop', price=1500.0, category='Electronics'}
Lower (< 250): Product{name='Shoes', price=120.0, category='Fashion'}
Poll first: Product{name='Shoes', price=120.0, category='Fashion'}
Poll last: Product{name='Laptop', price=1500.0, category='Electronics'}
Descending set: [Product{name='Smartphone', price=800.0, category='Electronics'}, Product{name='Tablet', price=600.0, category='Electronics'}, Product{name='
Process finished with exit code 0
```

Лістинг коду:

```
package com.education.ztu;

import java.util.NavigableSet;
import java.util.TreeSet;

public class ProductSetDemo {
    public static void main(String[] args) {
        TreeSet<Product> productSet = new TreeSet<>((p1, p2) ->
            Double.compare(p1.getPrice(), p2.getPrice()));

        // Додавання продуктів
        productSet.add(new Product("Laptop", 1500.0, "Electronics"));
        productSet.add(new Product("Smartphone", 800.0, "Electronics"));
        productSet.add(new Product("Shoes", 120.0, "Fashion"));
        productSet.add(new Product("Watch", 250.0, "Accessories"));
        productSet.add(new Product("Tablet", 600.0, "Electronics"));

        // Методи роботи з множиною
        System.out.println("First product: " + productSet.first());
        System.out.println("Last product: " + productSet.last());
        System.out.println("Head set (< 800): " + productSet.headSet(new Product("Dummy",
800.0, "")));
        System.out.println("Sub set (120 - 800): " + productSet.subSet(new
Product("Dummy", 120.0, ""), new Product("Dummy", 800.0, "")));
        System.out.println("Tail set (>= 600): " + productSet.tailSet(new Product("Dummy",
600.0, "")));

        System.out.println("Ceiling (>= 700): " + productSet.ceiling(new Product("Dummy",
700.0, "")));
        System.out.println("Floor (<= 1000): " + productSet.floor(new Product("Dummy",
1000.0, "")));
        System.out.println("Higher (> 800): " + productSet.higher(new Product("Dummy",
800.0, "")));
    }
}
```

```

        System.out.println("Lower (< 250): " + productSet.lower(new Product("Dummy",
250.0, "")));

        System.out.println("Poll first: " + productSet.pollFirst());
        System.out.println("Poll last: " + productSet.pollLast());

        NavigableSet<Product> descendingSet = productSet.descendingSet();
        System.out.println("Descending set: " + descendingSet);
    }
}

```

Завдання 6. Створити Map що містить пари (ключ, значення) - ім'я продукту та об'єкт продукту (клас Product).

- Використовуємо клас `HashMap`,
- Продемонструвати роботу з `Map` використовуючи методи (`put`, `get`, `get`, `containsKey`, `containsValue`, `clear`, `putIfAbsent`, `keySet`, `values`, `putAll`, `remove`, `size`)
- Викликати метод `entrySet` та продемонструвати роботу з набором значень, що він поверне (`getKey`, `getValue`, `setValue`)

Виконання коду:

```
Run ProductMapDemo x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=63687:D:\IntelliJ IDEA Community Edition
Get product by name 'Laptop': Product{name='Laptop', price=1500.0, category='Electronics'}
Contains key 'Tablet': false
Contains value (Product 'Shoes'): false
All keys: [Laptop, Watch, Shoes, Smartphone]
All values: [Product{name='Laptop', price=1500.0, category='Electronics'}, Product{name='Watch', price=250.0, category='Accessories'}, Product{name='Shoes',
Size of map: 5
Key: Laptop, Value: Product{name='Laptop', price=1500.0, category='Electronics'}
Key: Watch, Value: Product{name='Watch', price=250.0, category='Accessories'}
Key: Shoes, Value: Product{name='Shoes', price=120.0, category='Fashion'}
Key: Tablet, Value: Product{name='Tablet', price=600.0, category='Electronics'}
Key: Smartphone, Value: Product{name='Smartphone', price=800.0, category='Electronics'}
Map after clear: {}

Process finished with exit code 0
```

Лістинг коду:

```
package com.education.ztu;

import java.util.HashMap;
import java.util.Map;
import java.util.Set;

public class ProductMapDemo {
    public static void main(String[] args) {
        Map<String, Product> productMap = new HashMap<>();

        // Додавання продуктів
        productMap.put("Laptop", new Product("Laptop", 1500.0, "Electronics"));
        productMap.put("Smartphone", new Product("Smartphone", 800.0, "Electronics"));
        productMap.put("Shoes", new Product("Shoes", 120.0, "Fashion"));

        // Використання методів HashMap
        System.out.println("Get product by name 'Laptop': " + productMap.get("Laptop"));
        System.out.println("Contains key 'Tablet': " + productMap.containsKey("Tablet"));
        System.out.println("Contains value (Product 'Shoes'): " +
            productMap.containsValue(new Product("Shoes", 120.0, "Fashion")));
    }
}
```

					<p style="text-align: center;"><i>ІПТР.420001.123-ЗЛ</i></p>	Арк.
						37
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		


```

public class CollectionsDemo {
    public static void main(String[] args) {
        List<Product> products = Arrays.asList(
            new Product("Laptop", 1500.0, "Electronics"),
            new Product("Smartphone", 800.0, "Electronics"),
            new Product("Shoes", 120.0, "Fashion"),
            new Product("Watch", 250.0, "Accessories"),
            new Product("Tablet", 600.0, "Electronics")
        );

        // Сортвання
        Collections.sort(products, Comparator.comparingDouble(Product::getPrice));
        System.out.println("Sorted products: " + products);

        // Двійковий пошук
        int index = Collections.binarySearch(products, new Product("Shoes", 120.0,
"Fashion"), Comparator.comparingDouble(Product::getPrice));
        System.out.println("Index of 'Shoes': " + index);

        // Reverse та shuffle
        Collections.reverse(products);
        System.out.println("Reversed products: " + products);

        Collections.shuffle(products);
        System.out.println("Shuffled products: " + products);

        // Fill
        List<Product> copyList = new ArrayList<>(products);
        Collections.fill(copyList, new Product("Default", 0.0, "None"));
        System.out.println("Filled list: " + copyList);

        // Max та min
        Product maxPriceProduct = Collections.max(products,
Comparator.comparingDouble(Product::getPrice));
        Product minPriceProduct = Collections.min(products,
Comparator.comparingDouble(Product::getPrice));
        System.out.println("Max price product: " + maxPriceProduct);
        System.out.println("Min price product: " + minPriceProduct);

        // Copy
        List<Product> copyOfProducts = new ArrayList<>(Arrays.asList(new
Product[products.size()]));
        Collections.copy(copyOfProducts, products);
        System.out.println("Copied list: " + copyOfProducts);

        // Rotate
        Collections.rotate(products, 2);
        System.out.println("Rotated products: " + products);

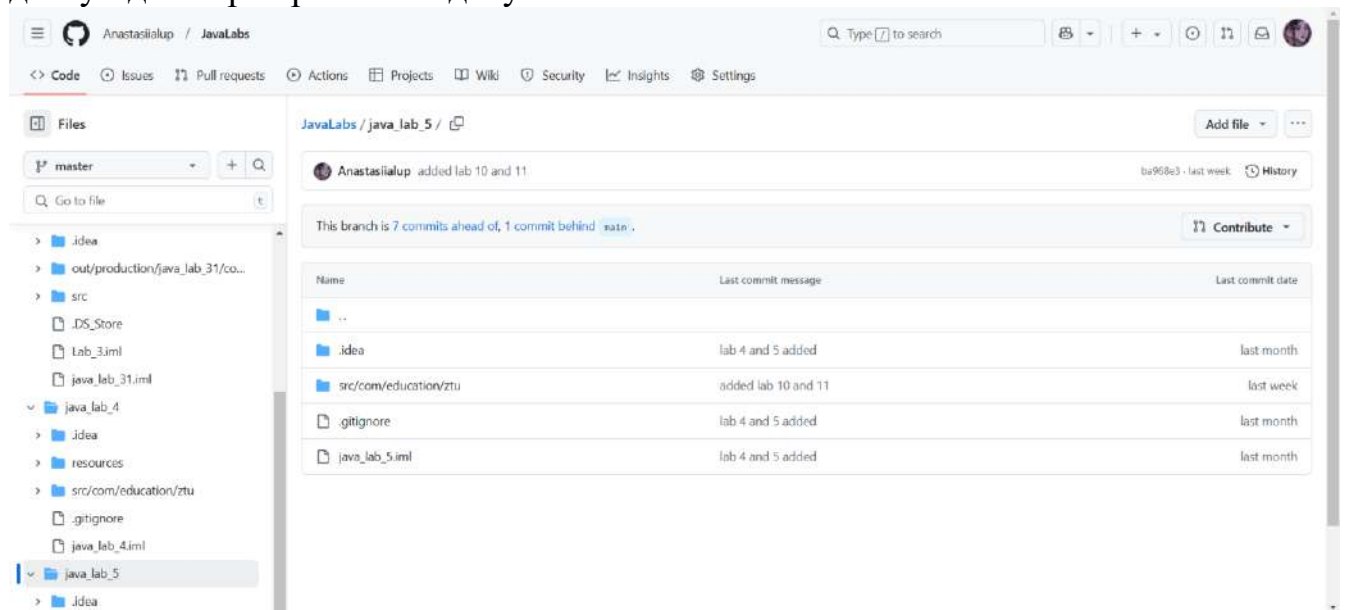
        // Checked collection
        Collection<Product> checkedCollection = Collections.checkedCollection(new
ArrayList<>(products), Product.class);
        System.out.println("Checked collection: " + checkedCollection);

        // Частота
        Product tablet = new Product("Tablet", 600.0, "Electronics");
        int frequency = Collections.frequency(products, tablet);
        System.out.println("Frequency of 'Tablet': " + frequency);
    }
}

```

```
}
}
```

Завдання 7. В GitLab проекті Java_labs_ztu, створити директорію Lab_5 та запустити в Lab_5 виконану лабораторну роботу. Надати доступ для перевірки викладачу.



Висновок:

У ході виконання лабораторної роботи було засвоєно основи роботи з Java Collections Framework. Створено клас Product та використано різні типи колекцій для зберігання об'єктів цього класу:

Динамічні масиви (через ArrayList та LinkedList) дозволяють ефективно додавати, видаляти та маніпулювати елементами. Було продемонстровано методи, такі як add, remove, get, size, clear.

Черга (через ArrayDeque) використовувалась для реалізації структури даних з підтримкою додавання та видалення елементів з обох кінців.

Множина (через TreeSet) продемонструвала можливості зберігання унікальних елементів з підтримкою сортування та пошуку.

Мар (через HashMap) використана для зберігання пар "ключ-значення", дозволяючи ефективно працювати з асоціативними масивами, застосовуючи методи, як put, get, keySet.

За допомогою класу Collections виконано сортування, пошук, реверсування, та інші операції для маніпуляцій з колекціями.

Ця робота дозволила глибше зрозуміти механізми роботи з колекціями та ефективні методи маніпулювання даними в Java.

Лабораторна робота №6

Тема: Обробка виключних ситуацій. Потоки вводу-виводу. Робота з файлами.

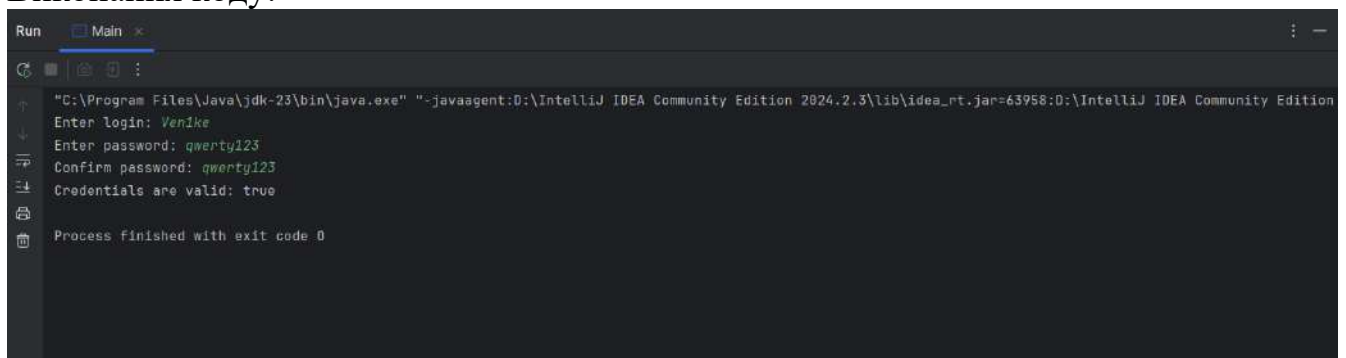
Мета роботи: обробка виключних ситуацій, створення власних класів винятків, робота з потоками вводу-виводу.

Завдання 1. Створити консольний Java проект `java_lab_6` з пакетом `com.education.ztu`. Створіть в корені проекту папку `directory_for_files`. Всі файли з якими ви будете працювати при виконанні завдань повинні знаходитись в ній.

Завдання 2. Перевірка логіну та паролю:

- Створити статичний метод `checkCredentials`, який приймає на вхід три параметри: `login`, `password` і `confirmPassword`.
- `Login` повинен містити лише латинські літери, цифри та знак підкреслення. Довжина `login` має бути меншою за 20 символів. Якщо `login` не відповідає цим вимогам, необхідно викинути `WrongLoginException`.
- `Password` повинен містити лише латинські літери, цифри та знак підкреслення. Довжина `password` має бути менше 20 символів. Також `password` і `confirmPassword` повинні бути рівними. Якщо `password` не відповідає цим вимогам, необхідно викинути `WrongPasswordException`.
- `WrongPasswordException` і `WrongLoginException` - користувацькі класи виключення з двома конструкторами - один за замовчуванням, другий приймає повідомлення виключення і передає його в конструктор класу `Exception`.
- Обробка винятків проводиться усередині методу.
- Використовуємо `multi-catch block`.
- Метод повертає `true`, якщо значення є вірними або `false` в іншому випадку.

Виконання коду:



```
Run Main x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=63958:D:\IntelliJ IDEA Community Edition"
Enter login: Ven1ke
Enter password: qwerty123
Confirm password: qwerty123
Credentials are valid: true
Process finished with exit code 0
```

Лістинг коду:

```
package com.education.ztu;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
```

					ІПТР.420001.123-ЗЛ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Scanner scanner = new Scanner(System.in);

// Запитуємо логін
System.out.print("Enter login: ");
String login = scanner.nextLine();

// Запитуємо пароль
System.out.print("Enter password: ");
String password = scanner.nextLine();

// Запитуємо підтвердження пароля
System.out.print("Confirm password: ");
String confirmPassword = scanner.nextLine();

// Виконуємо перевірку введених даних
boolean result = CredentialsChecker.checkCredentials(login, password,
confirmPassword);
System.out.println("Credentials are valid: " + result);
}
}

```

```

package com.education.ztu;

public class CredentialsChecker {
    public static boolean checkCredentials(String login, String password, String
confirmPassword) {
        try {
            if (!login.matches("[A-Za-z0-9_]+") || login.length() >= 20) {
                throw new WrongLoginException("Login should contain only Latin letters,
numbers, or underscores, and be less than 20 characters.");
            }

            if (!password.matches("[A-Za-z0-9_]+") || password.length() >= 20 ||
!password.equals(confirmPassword)) {
                throw new WrongPasswordException("Password should contain only Latin
letters, numbers, or underscores, be less than 20 characters, and match the confirm
password.");
            }

            return true; // Credentials are valid
        } catch (WrongLoginException | WrongPasswordException e) {
            System.out.println("Error: " + e.getMessage());
            return false;
        }
    }
}

```

Завдання 3. Запис звіту про покупки в текстовий файл та читання з нього:

- Перевикористати код для формування звіту з покупок з лабораторної роботи 4.
- Після покупки, записати звіт у файл, який містить інформацію про вміст кошика.
- Використовуємо клас FileWriter або PrintWriter для запису звіту.

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

- Використовуємо FileReader для читання звіту та відображення в консолі.
- Не використовувати try-with-resources.

Виконання коду:

```

Run PurchaseReportApp
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=64163:D:\IntelliJ IDEA Community Edition
Report written to file successfully.
Reading report from file:
Date and time of purchase: 23.12.2024 04:23:13
=====
Item No Item Name Category Price
=====
1 Jeans Women's Clothing $1,500.78
2 Skirt Women's Clothing $1,000.56
3 Tie Men's Clothing $500.78
=====
Total: $3,002.12
Process finished with exit code 0

```

purchase_report.txt – Блокнот

Файл Правка Формат Вид Справка

```

date and time of purchase: 23.12.2024 04:23:13
=====
Item No Item Name Category Price
=====
1 Jeans Women's Clothing $1,500.78
2 Skirt Women's Clothing $1,000.56
3 Tie Men's Clothing $500.78
=====
Total: $3,002.12

```

Лістинг коду:

```

package com.education.ztu;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class PurchaseReport {
    private static final String DIRECTORY_PATH = "directory_for_files";
    private static final String FILE_PATH = DIRECTORY_PATH + "/purchase_report.txt";

    // Метод для запису звіту про покупки у файл
    public static void writeReport(String reportContent) {
        // Перевірка і створення папки, якщо вона не існує
        File directory = new File(DIRECTORY_PATH);
        if (!directory.exists()) {
            directory.mkdirs(); // Створює папку
        }

        FileWriter writer = null;
        try {
            writer = new FileWriter(FILE_PATH);
            writer.write(reportContent);
            System.out.println("Report written to file successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (writer != null) {
                try {
                    writer.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

```

```

    } catch (IOException e) {
        System.err.println("Error writing to file: " + e.getMessage());
    } finally {
        if (writer != null) {
            try {
                writer.close();
            } catch (IOException e) {
                System.err.println("Error closing writer: " + e.getMessage());
            }
        }
    }
}

// Метод для читання звіту про покупки з файлу
public static void readReport() {
    FileReader reader = null;
    try {
        reader = new FileReader(FILE_PATH);
        int character;
        System.out.println("Reading report from file:");
        while ((character = reader.read()) != -1) {
            System.out.print((char) character);
        }
        System.out.println();
    } catch (IOException e) {
        System.err.println("Error reading from file: " + e.getMessage());
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e) {
                System.err.println("Error closing reader: " + e.getMessage());
            }
        }
    }
}
}

```

```

package com.education.ztu;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.text.NumberFormat;
import java.text.SimpleDateFormat;
import java.util.*;

public class PurchaseReportApp {

    private static final String DIRECTORY_PATH = "directory_for_files";
    private static final String FILE_PATH = DIRECTORY_PATH + "/purchase_report.txt";

    public static void main(String[] args) {
        // Перевірка наявності папки для файлів
        File directory = new File(DIRECTORY_PATH);
        if (!directory.exists()) {
            directory.mkdirs(); // Створення папки, якщо вона не існує
        }
    }
}

```

```

    }

    // Формуємо звіт про покупки
    String report = generatePurchaseReport();

    // Записуємо звіт у файл
    writeReportToFile(report);

    // Читаємо звіт з файлу і виводимо на консоль
    readReportFromFile();
}

// Метод для генерації звіту про покупки
private static String generatePurchaseReport() {
    // Вибір локалі (можна змінити на французьку або іншу)
    Locale selectedLocale = new Locale("en", "US");
    ResourceBundle bundle = ResourceBundle.getBundle("data", selectedLocale);

    // Дата та час форматуємо
    String date = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss").format(new Date());

    // Товари
    Object[][] items = {
        {1, "Jeans", "Women's Clothing", 1500.78},
        {2, "Skirt", "Women's Clothing", 1000.56},
        {3, "Tie", "Men's Clothing", 500.78}
    };

    double total = 0;
    StringBuilder report = new StringBuilder();

    // Формуємо звіт
    report.append(bundle.getString("date")).append(": ").append(date).append("\n");
    report.append("=====\n");
    report.append(String.format("%-3s %-10s %-15s %-10s\n", "Item No", "Item Name",
"Category", "Price"));
    report.append("=====\n");

    // Додаємо інформацію про кожен товар
    NumberFormat currencyFormat = NumberFormat.getCurrencyInstance(selectedLocale);
    for (Object[] item : items) {
        report.append(String.format("%-3d %-10s %-15s %-10s\n", item[0], item[1],
item[2], currencyFormat.format(item[3])));
        total += (double) item[3];
    }

    // Додаємо загальну суму
    report.append("=====\n");
    report.append(bundle.getString("total")).append(":
").append(currencyFormat.format(total));

    return report.toString();
}

// Метод для запису звіту в файл
private static void writeReportToFile(String report) {
    try (FileWriter writer = new FileWriter(FILE_PATH)) {
        writer.write(report);
        System.out.println("Report written to file successfully.");
    } catch (IOException e) {

```

```

        System.err.println("Error writing to file: " + e.getMessage());
    }
}

// Метод для читання звіту з файлу
private static void readReportFromFile() {
    try (FileReader reader = new FileReader(FILE_PATH)) {
        int character;
        System.out.println("Reading report from file:");
        while ((character = reader.read()) != -1) {
            System.out.print((char) character);
        }
        System.out.println();
    } catch (IOException e) {
        System.err.println("Error reading from file: " + e.getMessage());
    }
}
}

```

Завдання 4. Копіювання файлу до іншого файлу:

- Написати клас, який копіює вміст текстового файлу та картинки з одного файлу до іншого.
- Використовуємо класи `BufferedReader`, `FileReader`, `BufferedWriter`, `FileWriter`, `FileInputStream`, `FileOutputStream`.
- Використати `try-with-resources`.

Виконання коду:



Лістинг коду:

```

package com.education.ztu;

import java.io.*;

public class FileCopyExample {

    public static void main(String[] args) {
        // Шляхи до вихідних файлів
        String textSourceFile = "sourcetext.txt";
        String imageSourceFile = "sourceimage.png";

        // Шляхи до цільових файлів
        String textDestinationFile = "sourcetextcopy.txt";
        String imageDestinationFile = "sourceimagecopy.png";

        // Створюємо вихідні файли, якщо вони не існують
        createFileIfNotExist(textSourceFile);
        createFileIfNotExist(imageSourceFile);

        // Створюємо файли для копії, якщо їх не існує
    }
}

```

					IPTP.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		46

```

        createFileIfNotExist(textDestinationFile);
        createFileIfNotExist(imageDestinationFile);

        // Копіюємо вміст текстового файлу
        copyTextFile(textSourceFile, textDestinationFile);

        // Копіюємо вміст зображення
        copyImageFile(imageSourceFile, imageDestinationFile);
    }

    // Метод для створення файлу, якщо він не існує
    private static void createFileIfNotExist(String fileName) {
        File file = new File(fileName);
        if (!file.exists()) {
            try {
                if (file.createNewFile()) {
                    System.out.println("Файл " + fileName + " був успішно створений.");
                }
            } catch (IOException e) {
                System.out.println("Помилка при створенні файлу " + fileName + ": " +
e.getMessage());
            }
        }
    }

    // Метод для копіювання текстового файлу
    private static void copyTextFile(String sourceFile, String destinationFile) {
        try (
            BufferedReader reader = new BufferedReader(new FileReader(sourceFile));
            BufferedWriter writer = new BufferedWriter(new
FileWriter(destinationFile))
        ) {
            String line;
            while ((line = reader.readLine()) != null) {
                writer.write(line);
                writer.newLine(); // Додаємо новий рядок після кожного запису
            }
            System.out.println("Текстовий файл був успішно скопійований в " +
destinationFile);
        } catch (IOException e) {
            System.out.println("Помилка при копіюванні текстового файлу: " +
e.getMessage());
        }
    }

    // Метод для копіювання зображення
    private static void copyImageFile(String sourceFile, String destinationFile) {
        try (
            FileInputStream fileInputStream = new FileInputStream(sourceFile);
            FileOutputStream fileOutputStream = new FileOutputStream(destinationFile)
        ) {
            byte[] buffer = new byte[1024];
            int length;
            while ((length = fileInputStream.read(buffer)) > 0) {
                fileOutputStream.write(buffer, 0, length);
            }
            System.out.println("Зображення було успішно скопійоване в " +
destinationFile);
        } catch (IOException e) {
            System.out.println("Помилка при копіюванні зображення: " + e.getMessage());
        }
    }

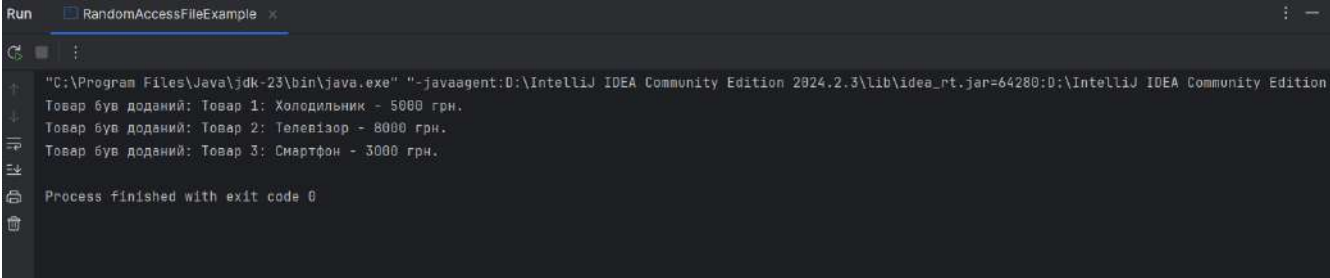
```

```
}  
}  
}
```

Завдання 5. Робота з класом RandomAccessFile:

- Дописати текст в декількох місцях в текстовому файлі. Можна використати текстовий файл зі списком товарів (наприклад, дописати декілька товарів) або будь-який інший файл з текстом.

Виконання коду:



Лістинг коду:

```
package com.education.ztu;  
  
import java.io.*;  
  
public class RandomAccessFileExample {  
  
    public static void main(String[] args) {  
        // Шлях до текстового файлу  
        String fileName = "products.txt";  
  
        // Створюємо файл, якщо його не існує  
        createFileIfNotExist(fileName);  
  
        // Допишуємо товар в кілька місць в файлі  
        addProductToFile(fileName, "Товар 1: Холодильник - 5000 грн.");  
        addProductToFile(fileName, "Товар 2: Телевізор - 8000 грн.");  
        addProductToFile(fileName, "Товар 3: Смартфон - 3000 грн.");  
    }  
  
    // Метод для створення файлу, якщо він не існує  
    private static void createFileIfNotExist(String fileName) {  
        File file = new File(fileName);  
        if (!file.exists()) {  
            try {  
                if (file.createNewFile()) {  
                    System.out.println("Файл " + fileName + " був успішно створений.");  
                }  
            } catch (IOException e) {  
                System.out.println("Помилка при створенні файлу " + fileName + ": " +  
e.getMessage());  
            }  
        }  
    }  
  
    // Метод для додавання товару в файл в певне місце  
    private static void addProductToFile(String fileName, String product) {  
        try (RandomAccessFile file = new RandomAccessFile(fileName, "rw")) {  
            // Знаходимо першу порожню лінію або додаємо товар в кінець файлу
```



```

        file.seek(file.length()); // Переміщаємо курсор в кінець файлу
        file.writeBytes(product + "\n");
        System.out.println("Товар був доданий: " + product);
    } catch (IOException e) {
        System.out.println("Помилка при додаванні товару в файл: " + e.getMessage());
    }
}
}

```

Завдання 6. Робота з класом File:

- Створити нову папку з ім'ям inner_directory.
- Вивести абсолютний шлях створеної папки.
- Вивести ім'я батьківської директорії.
- Створити два текстових файли всередині папки inner_directory.
- Один файл видалити.
- Переіменувати папку inner_directory в renamed_inner_directory
- Вивести список файлів та папок в папці directory_for_files, їх розмір та тип (файл, папка).

Виконання коду:

```

Run FileExample x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2024.2.3\lib\idea_rt.jar=64325:D:\IntelliJ IDEA Community Edition
Не вдалося створити папку.
Абсолютний шлях створеної папки: C:\Users\Acer\Desktop\Java\Java\java_lab_6\inner_directory
Батьківська директорія: Це коренева директорія проєкту
Файл file1.txt був успішно створений.
Файл file2.txt вже існує.
Файл file1.txt був успішно видалений.
Папка з ім'ям renamed_inner_directory вже існує.
Список файлів і папок у директорії C:\Users\Acer\Desktop\Java\Java\java_lab_6\directory_for_files:
Файл: purchase_report.txt, Розмір: 369 байт

Process finished with exit code 0

```

Лістинг коду:

```

package com.education.ztu;

import java.io.File;

public class FileExample {

    public static void main(String[] args) {
        // 1. Створення нової папки
        File innerDirectory = new File("inner_directory");
        if (innerDirectory.mkdir()) {
            System.out.println("Папка створена: " + innerDirectory.getAbsolutePath());
        } else {
            System.out.println("Не вдалося створити папку.");
        }

        // 2. Виведення абсолютного шляху створеної папки
        System.out.println("Абсолютний шлях створеної папки: " +
            innerDirectory.getAbsolutePath());

        // 3. Виведення ім'я батьківської директорії
        File parentDirectory = innerDirectory.getParentFile();
        System.out.println("Батьківська директорія: " + (parentDirectory != null ?
            parentDirectory.getAbsolutePath() : "Це коренева директорія проєкту"));
    }
}

```

```

// 4. Створення двох текстових файлів всередині папки
createFile(innerDirectory, "file1.txt");
createFile(innerDirectory, "file2.txt");

// 5. Видалення одного файлу
File fileToDelete = new File(innerDirectory, "file1.txt");
if (fileToDelete.delete()) {
    System.out.println("Файл " + fileToDelete.getName() + " був успішно
видалений.");
} else {
    System.out.println("Не вдалося видалити файл " + fileToDelete.getName());
}

// 6. Переіменування папки, якщо нова назва ще не існує
File renamedDirectory = new File("renamed_inner_directory");
if (!renamedDirectory.exists()) {
    if (innerDirectory.renameTo(renamedDirectory)) {
        System.out.println("Папка була переіменована в: " +
renamedDirectory.getAbsolutePath());
    } else {
        System.out.println("Не вдалося переіменувати папку.");
    }
} else {
    System.out.println("Папка з ім'ям " + renamedDirectory.getName() + " вже
існує.");
}

// 7. Виведення списку файлів та папок у папці "directory_for_files", їх розмір та
тип (файл, папка)
File directoryForFiles = new File("directory_for_files");
if (directoryForFiles.exists() && directoryForFiles.isDirectory()) {
    File[] files = directoryForFiles.listFiles();
    if (files != null && files.length > 0) {
        System.out.println("Список файлів і папок у директорії " +
directoryForFiles.getAbsolutePath() + ":");
        for (File file : files) {
            String type = file.isDirectory() ? "Папка" : "Файл";
            System.out.println(type + ": " + file.getName() + ", Розмір: " +
file.length() + " байт");
        }
    } else {
        System.out.println("У директорії немає файлів або папок.");
    }
} else {
    System.out.println("Директорія " + directoryForFiles.getAbsolutePath() + " не
існує.");
}

// Метод для створення файлів всередині вказаної папки
private static void createFile(File directory, String fileName) {
    File file = new File(directory, fileName);
    try {
        if (file.createNewFile()) {
            System.out.println("Файл " + fileName + " був успішно створений.");
        } else {
            System.out.println("Файл " + fileName + " вже існує.");
        }
    } catch (Exception e) {
        System.out.println("Помилка при створенні файлу " + fileName + ": " +

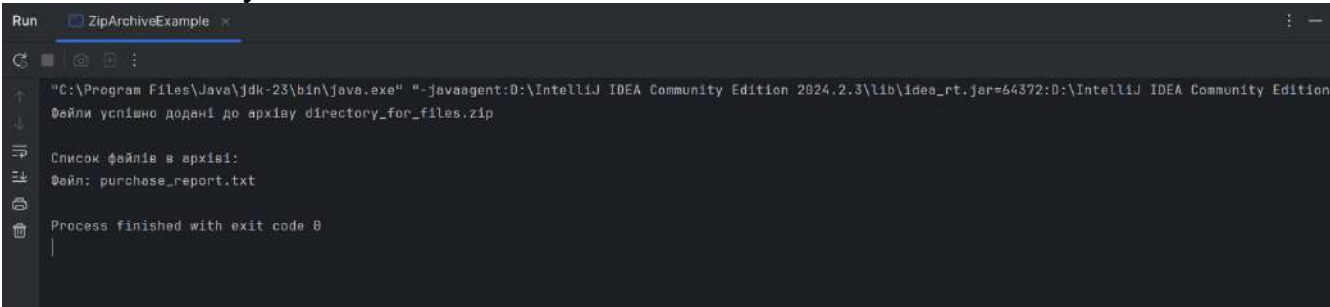
```

```
e.getMessage());
    }
}
}
```

Завдання 7. Створення архіву:

- Додати всі створені файли в папці directory_for_files до архіву.
- Використати клас ZipOutputStream.
- Вивести список файлів з архіву. Використати клас ZipInputStream.

Виконання коду:



Лістинг коду:

```
package com.education.ztu;

import java.io.*;
import java.nio.file.*;
import java.util.zip.*;

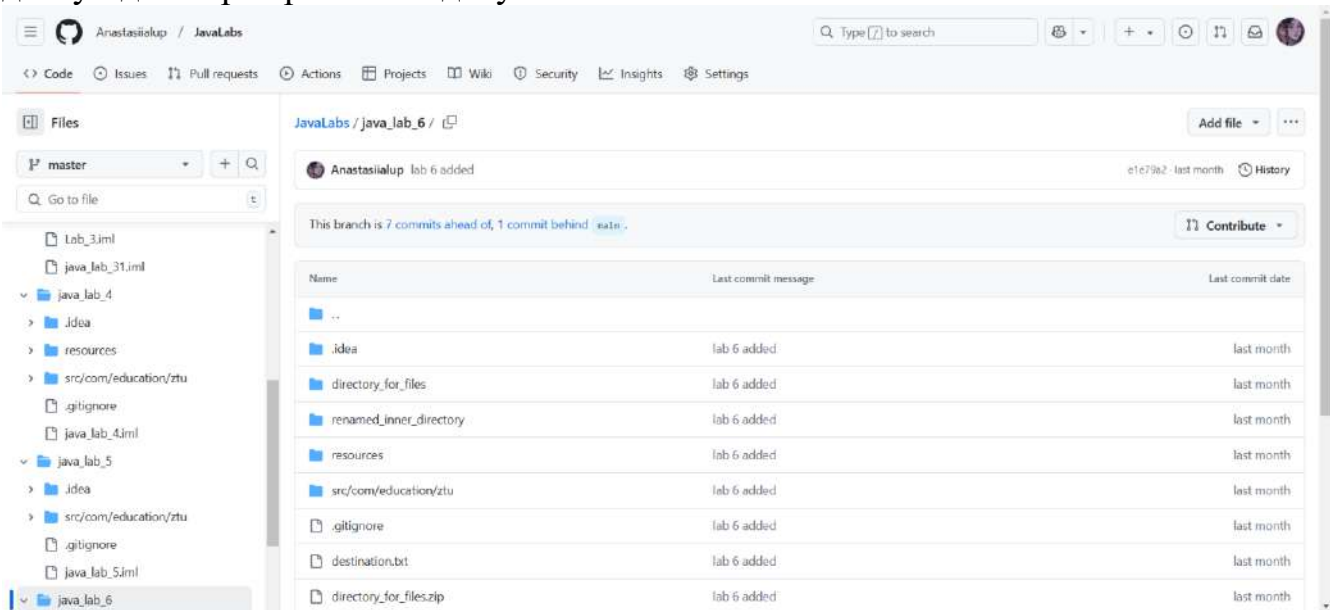
public class ZipArchiveExample {
    public static void main(String[] args) {
        File directory = new File("directory_for_files"); // Папка з файлами для архівування
        File zipFile = new File("directory_for_files.zip"); // Архів

        // Створення архіву та додавання файлів
        try (ZipOutputStream zipOut = new ZipOutputStream(new FileOutputStream(zipFile))) {
            File[] filesToZip = directory.listFiles();
            if (filesToZip != null) {
                for (File file : filesToZip) {
                    try (FileInputStream fileIn = new FileInputStream(file)) {
                        ZipEntry zipEntry = new ZipEntry(file.getName());
                        zipOut.putNextEntry(zipEntry);

                        byte[] buffer = new byte[1024];
                        int length;
                        while ((length = fileIn.read(buffer)) > 0) {
                            zipOut.write(buffer, 0, length);
                        }
                        zipOut.closeEntry();
                    }
                }
                System.out.println("Файли успішно додані до архіву " + zipFile.getName());
            } else {
                System.out.println("Папка порожня або не існує.");
            }
        } catch (IOException e) {
            System.out.println("Помилка під час архівування файлів: " + e.getMessage());
        }
    }
}
```

```
// Читання файлів з архіву
System.out.println("\nСписок файлів в архіві:");
try (ZipInputStream zipIn = new ZipInputStream(new FileInputStream(zipFile))) {
    ZipEntry entry;
    while ((entry = zipIn.getNextEntry()) != null) {
        System.out.println("Файл: " + entry.getName());
        zipIn.closeEntry();
    }
} catch (IOException e) {
    System.out.println("Помилка під час читання архіву: " + e.getMessage());
}
}
```

Завдання 8. В GitLab проекті Java_labs_ztu, створити директорію Lab_6 та запустити в Lab_6 виконану лабораторну роботу. Надати доступ для перевірки викладачу.



Висновок:
У лабораторній роботі було опрацьовано обробку виключень та роботу з файлами в Java. Створено користувацькі виключення для перевірки логіну та паролю. Також реалізовано запис та читання даних з файлів, копіювання файлів, дописування в них та архівування за допомогою різних класів, таких як FileWriter, FileReader, RandomAccessFile, ZipOutputStream і інші. Це допомогло краще зрозуміти, як працювати з файлами та виключеннями в Java.

Лабораторна робота №7

Тема: Багатопоточне програмування в Java

Мета роботи: практика роботи з потоками в Java

Завдання 1. Створити консольний Java проект `java_lab_7` з пакетом `com.education.ztu`.

Завдання 2. Створити клас, що розширює `Thread`:

- Створити клас `MyThread`, що розширює `Thread`.
- Перевизначити метод `run()`. У циклі `for` вивести на консоль повідомлення «Я люблю програмувати!!!» 100 разів.
- Створити екземпляр класу та запустити новий потік.
- Вивести ім'я створеного потоку, його пріоритет, перевірити чи він живий, чи є потоком демоном.
- Змінити ім'я, пріоритет створеного потоку та вивести в консоль оновлені значення.
- Після завершення роботи створеного потоку (використати метод `join()`) вивести ім'я головного потоку, та його пріоритет.
- Відобразити в консолі, коли ваш потік буде в стані `NEW`, `RUNNING`, `TERMINATED`.

Лістинг коду:

```
package com.education.ztu;

public class MyThread extends Thread {

    public MyThread(String name) {
        super(name);
    }

    @Override
    public void run() {
        System.out.println("Потік у стані RUNNING: " + this.getName());
        for (int i = 0; i < 100; i++) {
            System.out.println("Я люблю програмувати!!! " + (i + 1));
        }
        System.out.println("Потік завершив роботу: " + this.getName());
    }
}
```

					ІПТР.420001.123-ЗЛ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    public static void main(String[] args) {
        // Створюємо екземпляр класу MyThread
        MyThread myThread = new MyThread("InitialThread");

        // Виводимо інформацію про потік перед запуском
        System.out.println("Потік у стані NEW: " + myThread.getName());
        System.out.println("Ім'я потоку: " + myThread.getName());
        System.out.println("Пріоритет потоку: " + myThread.getPriority());
        System.out.println("Чи є потік демоном: " + myThread.isDaemon());
        System.out.println("Чи живий потік: " + myThread.isAlive());

        // Змінюємо ім'я і пріоритет потоку
        myThread.setName("MyProgrammingThread");
        myThread.setPriority(Thread.MAX_PRIORITY);
        System.out.println("Оновлене ім'я потоку: " + myThread.getName());
        System.out.println("Оновлений пріоритет потоку: " +
myThread.getPriority());

        // Запускаємо потік
        myThread.start();

        try {
            // Використовуємо join(), щоб дочекатися завершення потоку
            myThread.join();
        } catch (InterruptedException e) {
            System.out.println("Потік було перервано.");
        }

        // Виводимо інформацію про головний потік
        Thread mainThread = Thread.currentThread();
        System.out.println("Ім'я головного потоку: " + mainThread.getName());
        System.out.println("Пріоритет головного потоку: " +
mainThread.getPriority());

        // Перевіряємо стан потоку після завершення
        System.out.println("Чи живий потік: " + myThread.isAlive());
        System.out.println("Потік у стані TERMINATED: " + myThread.getName());
    }
}

```

Результат виконання:

```

Run MyThread x
C:\Users\Mastia\jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=53938:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin" -Dfile.encoding=UTF-8
Потік у стані NEW: InitialThread
Ім'я потоку: InitialThread
Пріоритет потоку: 5
Чи є потік демоном: false
Чи живий потік: false
Оновлене ім'я потоку: MyProgrammingThread
Оновлений пріоритет потоку: 10
Потік у стані RUNNING: MyProgrammingThread
Я люблю програмувати!!! 1
Я люблю програмувати!!! 2

```

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		54

```

Run MyThread
Я люблю програмувати!!! 99
Я люблю програмувати!!! 100
Потік завершив роботу: MyProgrammingThread
Ім'я головного потоку: main
Пріоритет головного потоку: 5
Чи живий потік: false
Потік у стані TERMINATED: MyProgrammingThread
Process finished with exit code 0

```

Завдання 3. Створити клас, що реалізує інтерфейс `Runnable` для виводу в консоль чисел від 0 до 10000, що діляться на 10 без залишку:

- Створити клас `MyRunnable`, який реалізує інтерфейс `Runnable`.
- Імплементувати метод `run()`.
- Визначити умову, якщо потік хочуть перервати, то завершити роботу потоку та вивести повідомлення «Розрахунок завершено!!!»
- Створити три потоки, які виконують завдання друку значень.
- Використовуємо статичний метод `Thread.sleep()`, щоб зробити паузу на 2 секунди для головного потоку, а після цього викликати для створених потоків метод `interrupt()`.

Лістинг коду:

```

package com.education.ztu;

class MyRunnable implements Runnable {

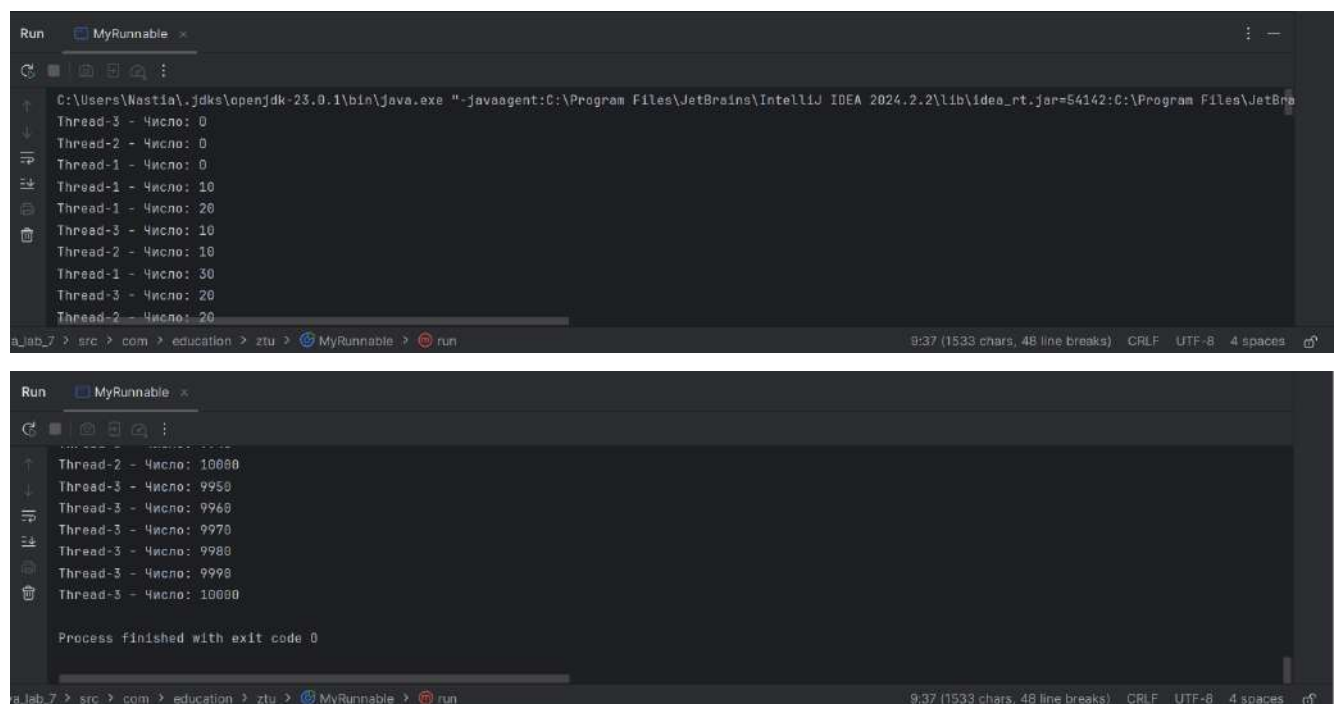
    @Override
    public void run() {
        try {
            for (int i = 0; i <= 10000; i++) {
                if (Thread.currentThread().isInterrupted()) {
                    System.out.println("Розрахунок завершено!!!");
                    return; // Завершуємо роботу потоку, якщо він перерваний
                }

                // Друк чисел, що діляться на 10 без залишку
                if (i % 10 == 0) {
                    System.out.println(Thread.currentThread().getName() + " -
Число: " + i);
                }
            }
        } catch (Exception e) {
            System.out.println(Thread.currentThread().getName() + " було
перервано.");
        }
    }
}

```

```
    }  
}  
  
public static void main(String[] args) {  
    // Створюємо три потоки, що використовують MyRunnable  
    Thread thread1 = new Thread(new MyRunnable(), "Thread-1");  
    Thread thread2 = new Thread(new MyRunnable(), "Thread-2");  
    Thread thread3 = new Thread(new MyRunnable(), "Thread-3");  
  
    // Запускаємо потоки  
    thread1.start();  
    thread2.start();  
    thread3.start();  
  
    try {  
        // Затримка головного потоку на 2 секунди  
        Thread.sleep(2000);  
  
        // Перериваємо всі три потоки  
        thread1.interrupt();  
        thread2.interrupt();  
        thread3.interrupt();  
    } catch (InterruptedException e) {  
        System.out.println("Головний потік був перерваний.");  
    }  
}
```

Результат виконання:



Завдання 4. Створити клас, що реалізує інтерфейс Runnable для виведення арифметичної прогресії від 1 до 100 з кроком 1:

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

- Створити клас, який реалізує інтерфейс Runnable.
- Створити об'єкт зі статичною змінною result для збереження значення арифметичної прогресії.
- Перевизначити метод run(). Створити цикл for. У циклі виводимо через пробіл значення змінної result. Та додаємо наступне значення до змінної result та чекаємо 0,2 секунду.
- Забезпечити коректну роботу використовуючи синхронізований метод.
- Створити три потоки, які виконують завдання друку значень.

Лістинг коду:

```
package com.education.ztu;

public class ArithmeticProgression implements Runnable {
    // Статична змінна для збереження значення арифметичної прогресії
    private static int result = 1;
    private static final int STEP = 1;
    private static final int LIMIT = 100;

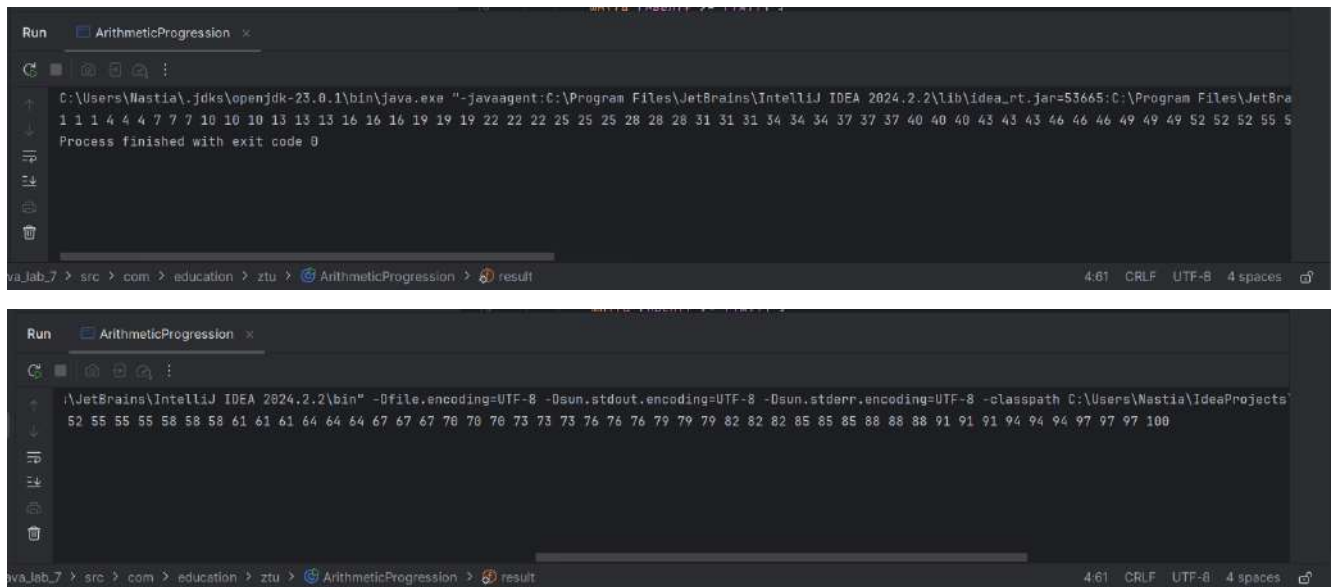
    // Синхронізований метод для обчислення прогресії та виведення результату
    public synchronized void calculateAndPrintProgression() {
        if (result <= LIMIT) {
            System.out.print(result + " ");
            result += STEP;
        }
    }

    @Override
    public void run() {
        while (result <= LIMIT) {
            calculateAndPrintProgression();
            try {
                // Затримка на 0,2 секунди
                Thread.sleep(200);
            } catch (InterruptedException e) {
                System.out.println("Потік було перервано.");
            }
        }
    }

    public static void main(String[] args) {
        // Створюємо три потоки, що використовують клас ArithmeticProgression
        Thread thread1 = new Thread(new ArithmeticProgression(), "Thread-1");
        Thread thread2 = new Thread(new ArithmeticProgression(), "Thread-2");
        Thread thread3 = new Thread(new ArithmeticProgression(), "Thread-3");

        // Запускаємо потоки
        thread1.start();
        thread2.start();
        thread3.start();
    }
}
```

Результат виконання:



```
Run ArithmeticProgression1 x
C:\Users\Nastia\jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=53665:C:\Program Files\JetBra
1 1 1 4 4 4 7 7 7 10 10 10 13 13 13 16 16 16 19 19 19 22 22 22 25 25 25 28 28 28 31 31 31 34 34 34 37 37 37 40 40 40 43 43 43 46 46 46 49 49 49 52 52 52 55 5
Process finished with exit code 0

va_lab_7 > src > com > education > ztu > ArithmeticProgression > result 4:61 CRLF UTF-8 4 spaces

Run ArithmeticProgression1 x
I:\JetBrains\IntelliJ IDEA 2024.2.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\Nastia\IdeaProjects
52 55 55 55 58 58 58 61 61 61 64 64 64 67 67 67 70 70 70 73 73 73 76 76 76 79 79 79 82 82 82 85 85 85 88 88 88 91 91 91 94 94 94 97 97 97 100

va_lab_7 > src > com > education > ztu > ArithmeticProgression > result 4:61 CRLF UTF-8 4 spaces
```

Завдання 5. Переробити 4 завдання використовуючи блок синхронізації.

Лістинг коду:

```
package com.education.ztu;

public class ArithmeticProgression1 implements Runnable {
    private static int result = 1;
    private static final int STEP = 1;
    private static final int LIMIT = 100;
    private static final Object lock = new Object(); // Об'єкт для синхронізації

    @Override
    public void run() {
        while (true) {
            synchronized (lock) {
                if (result > LIMIT) {
                    break;
                }
                System.out.print(result + " ");
                result += STEP;
            }

            try {
                Thread.sleep(200); // Затримка на 0,2 секунди
            } catch (InterruptedException e) {
                System.out.println("Потік було перервано.");
            }
        }
    }

    public static void main(String[] args) {
        Thread thread1 = new Thread(new ArithmeticProgression1(), "Thread-1");
    }
}
```

					ІПТР.420001.123-ЗЛ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Thread thread2 = new Thread(new ArithmeticProgression(), "Thread-2");
Thread thread3 = new Thread(new ArithmeticProgression(), "Thread-3");

thread1.start();
thread2.start();
thread3.start();
}
}

```

Результат виконання:

Завдання 6. Створити два потоки Reader та Printer. Reader зчитує введені дані з консолі та записує в змінну. Після цього інформує потік Printer та засипає на 1 секунду, а потік Reader виводить дотриманий рядок. І так повторюється знову, поки користувач не завершить роботу програми.

- Змінну треба використати як об'єкт для синхронізації.
- Тут необхідно використати wait() і notify().

Лістинг коду:

```

package com.education.ztu;

import java.util.Scanner;

public class ReaderPrinterExample {
    private static String sharedData = ""; // Змінна для синхронізації

    public static void main(String[] args) {
        Object lock = new Object();

        // Потік Reader для зчитування введених даних

```

```

Thread readerThread = new Thread(() -> {
    Scanner scanner = new Scanner(System.in);
    while (true) {
        synchronized (lock) {
            System.out.print("Введіть текст (або 'exit' для завершення):
");

            sharedData = scanner.nextLine();
            lock.notify(); // Інформуємо Printer про наявність нових даних

            if (sharedData.equalsIgnoreCase("exit")) {
                break; // Завершення, якщо користувач ввів "exit"
            }

            try {
                lock.wait(); // Очікування, поки Printer виведе дані
                Thread.sleep(1000); // Засипаємо на 1 секунду перед
повтором
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        scanner.close();
    });

// Потік Printer для виведення отриманих даних
Thread printerThread = new Thread(() -> {
    while (true) {
        synchronized (lock) {
            try {
                lock.wait(); // Чекаємо нових даних від Reader

                if (sharedData.equalsIgnoreCase("exit")) {
                    break; // Завершення, якщо введено "exit"
                }

                System.out.println("Printer отримав: " + sharedData);
                lock.notify(); // Інформуємо Reader, що дані виведені
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});

readerThread.start();
printerThread.start();
}

```

Результат виконання:

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		60

```

Run ReaderPrinterExample x
C:\Users\Nastia\.jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=54170:C:\Program Files\JetBra
Введіть текст (або 'exit' для завершення): привіт
Printer отримав: привіт
Введіть текст (або 'exit' для завершення): бувай
Printer отримав: бувай
Введіть текст (або 'exit' для завершення): exit
Process finished with exit code 0

```

Завдання 7. Створити програму для знаходження суми цифр в масиві на 1 000 000 елементів:

- Заповнити масив числами використовуючи клас Random.
- Реалізувати задачу в однопоточному та багатопоточному середовищі.
- Для багатопоточного середовища використати ExecutorService на 5 потоків та об'єкти потоків, що імплементують інтерфейси Runnable або Callable.
- Заміряти час виконання обох варіантів завдання використовуючи System.currentTimeMillis() та вивести результати в консоль.

Лістинг коду:

```

package com.education.ztu;

import java.util.Random;
import java.util.concurrent.*;

public class ArraySumExample {
    private static final int ARRAY_SIZE = 1_000_000;
    private static final int THREAD_COUNT = 5;
    private static final int[] array = new int[ARRAY_SIZE];

    static {
        Random random = new Random();
        for (int i = 0; i < ARRAY_SIZE; i++) {
            array[i] = random.nextInt(10); // Заповнюємо числами від 0 до 9
        }
    }

    // Однопоточний варіант
    public static long singleThreadSum() {
        long sum = 0;
        for (int value : array) {
            sum += value;
        }
        return sum;
    }
}

```

```

// Багатопоточний варіант з використанням ExecutorService
public static long multiThreadSum() throws InterruptedException,
ExecutionException {
    ExecutorService executor = Executors.newFixedThreadPool(THREAD_COUNT);
    int chunkSize = ARRAY_SIZE / THREAD_COUNT;
    long totalSum = 0;

    // Локальний клас ArraySumTask
    class ArraySumTask implements Callable<Long> {
        private final int start;
        private final int end;

        public ArraySumTask(int start, int end) {
            this.start = start;
            this.end = end;
        }

        @Override
        public Long call() {
            long sum = 0;
            for (int i = start; i < end; i++) {
                sum += array[i];
            }
            return sum;
        }
    }

    // Розподіл завдань
    Future<Long>[] futures = new Future[THREAD_COUNT];
    for (int i = 0; i < THREAD_COUNT; i++) {
        int start = i * chunkSize;
        int end = (i == THREAD_COUNT - 1) ? ARRAY_SIZE : start + chunkSize;
        futures[i] = executor.submit(new ArraySumTask(start, end));
    }

    // Збираємо результати
    for (Future<Long> future : futures) {
        totalSum += future.get();
    }

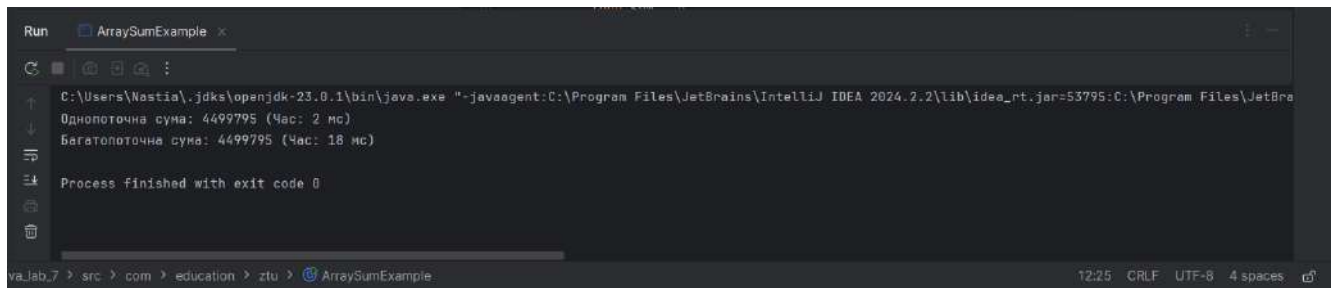
    executor.shutdown();
    return totalSum;
}

public static void main(String[] args) throws InterruptedException,
ExecutionException {
    // Однопоточний запуск
    long startSingleThread = System.currentTimeMillis();
    long singleThreadSum = singleThreadSum();
    long endSingleThread = System.currentTimeMillis();
    System.out.println("Однопоточна сума: " + singleThreadSum + " (Час: " +
(endSingleThread - startSingleThread) + " мс)");

    // Багатопоточний запуск
    long startMultiThread = System.currentTimeMillis();
    long multiThreadSum = multiThreadSum();
    long endMultiThread = System.currentTimeMillis();
    System.out.println("Багатопоточна сума: " + multiThreadSum + " (Час: " +
(endMultiThread - startMultiThread) + " мс)");
}

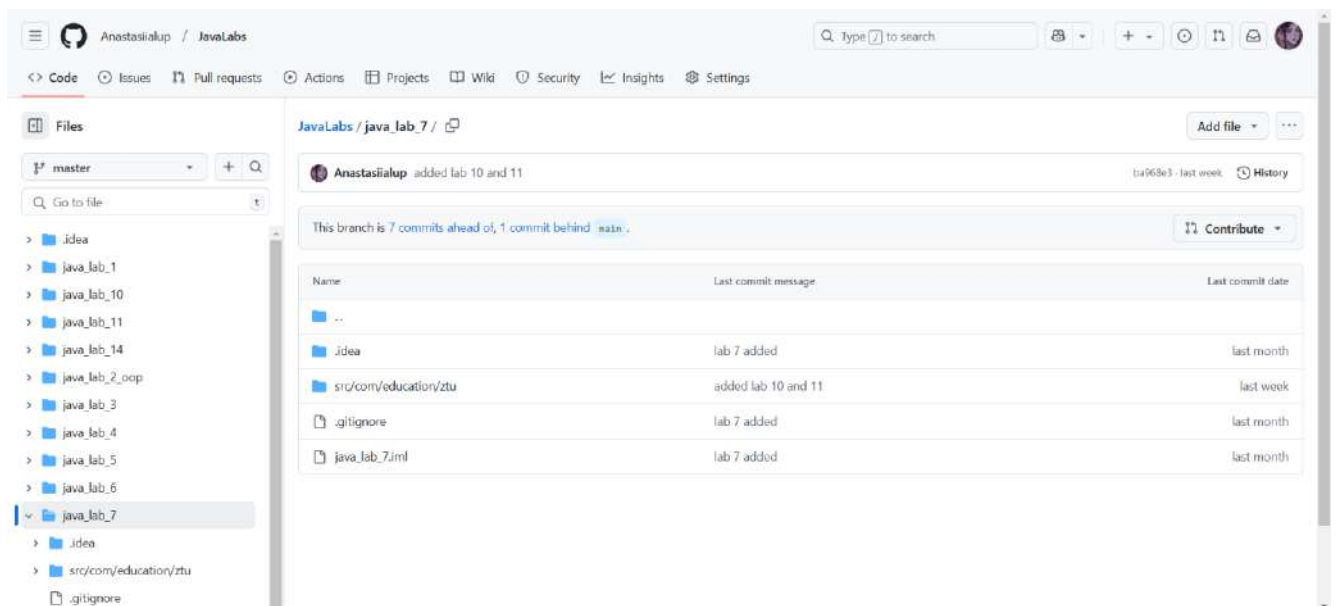
```

Результат виконання:



```
Run ArraySumExample x
C:\Users\Nastia\.jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=53795:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin" -Didea.config.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\conf -Didea.copyright.notification=false -Didea.log.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\log -Didea.version=2024.2.2 -jar C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin\idea_rt.jar 53795
Однопоточна сума: 4499795 (Час: 2 мс)
Багатопоточна сума: 4499795 (Час: 18 мс)
Process finished with exit code 0
```

Завдання 8. В GitLab проекті Java_labs_ztu, створити директорію Lab_7 та запусити в Lab_7 виконану лабораторну роботу. Надати доступ для перевірки викладачу.



Висновок:

У цій лабораторній роботі я ознайомилася з основними способами створення та використання багатопоточності в Java. Завдання продемонстрували реалізацію потоків через клас Thread і інтерфейс Runnable, використання синхронізації через ключове слово synchronized та методи wait() і notify(). Особливу увагу було приділено порівнянню ефективності однопоточного та багатопоточного підходів. Робота показала важливість правильного управління потоками для уникнення гонок даних і блокувань, що є основою для розробки багатозадачних застосунків.

Лабораторна робота №8

Тема: Лямбда вирази. Функціональні інтерфейси. Посилання на методи. Stream API.

Мета роботи: практика роботи з лямбда виразами, функціональними інтерфейсами; використання посилань на методи та Stream API при розробці програм на Java.

Завдання 1. Створити консольний Java проект java_lab_8 з пакетом com.education.ztu.

Завдання 2. Описати власний функціональний інтерфейс Printable з методом void print() та написати лямбда вираз цього інтерфейсу.

Лістинг коду:

```
// Завдання 2: Власний функціональний інтерфейс Printable
// Використання лямбда-виразу для реалізації методу print()
Printable printable = () -> System.out.println("Hello from Printable interface!");
printable.print();
```

```
@FunctionalInterface
interface Printable {
    void print(); // Метод для реалізації, що не приймає параметрів і нічого не повертає
}
```

Завдання 3. Написати лямбда вирази для вбудованих функціональних інтерфейсів:

а) Створити лямбда вираз, який повертає значення true, якщо рядок можна привести до числа, використовуючи функціональний інтерфейс Predicate.

Створити вираз лямбда, який перевіряє, що рядок можна привести до числа, використовуючи функціональний інтерфейс Predicate.

Написати програму, яка перевіряє, що рядок можна привести до числа, використовуючи метод and() функціонального інтерфейсу Predicate.

					IPTP.420001.123-ЗЛ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

- b) Написати лямбда вираз, який приймає на вхід рядок і виводить на консоль повідомлення "Пара почалася о 8:30", "Пара закінчилася о 9:50". Використовуємо функціональний інтерфейс Consumer і метод за замовчуванням andThen.
- c) Написати лямбда вираз, який виводить в консоль речення в з літерами у верхньому регістрі. Використовуємо функціональний інтерфейс Supplier.
- d) Написати лямбда вираз, який приймає на вхід рядок з набором чисел через пробіл та повертає добуток цих чисел. Використовуємо функціональний інтерфейс Function<String, Integer>.

Лістинг коду:

```
// Завдання 3a: Predicate для перевірки, чи можна рядок привести до числа
// Лямбда-вираз для перевірки, чи рядок є числом
Predicate<String> isNumeric = str -> {
    try {
        Double.parseDouble(str); // Перетворюємо рядок у число
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
};

String testString1 = "123.45";
String testString2 = "NotANumber";

System.out.println("Is numeric (123.45): " + isNumeric.test(testString1));
System.out.println("Is numeric (NotANumber): " +
isNumeric.test(testString2));

// Додаткова перевірка: чи рядок не порожній
Predicate<String> nonEmpty = str -> !str.isEmpty();

// Поєднуємо дві перевірки за допомогою методу and()
System.out.println("String is non-empty and numeric (123.45): " +
    nonEmpty.and(isNumeric).test(testString1));
System.out.println("String is non-empty and numeric (empty string): " +
    nonEmpty.and(isNumeric).test(""));

// Завдання 3b: Consumer для повідомлень
Consumer<String> startMessage = str -> System.out.println(str + " Пара
почалася о 8:30.");
Consumer<String> endMessage = str -> System.out.println(str + " Пара
закінчилася о 9:50.");

// Об'єднуємо два Consumer за допомогою методу andThen()
startMessage.andThen(endMessage).accept("Інформатика:");

// Завдання 3c: Supplier для виводу тексту у верхньому регістрі
Supplier<String> upperCaseMessage = () -> "Приклад тексту у верхньому
регістрі".toUpperCase();
System.out.println(upperCaseMessage.get());
```

```
// Завдання 3d: Function для обчислення добутку чисел у рядку
Function<String, Integer> multiplyNumbers = str -> {
    String[] numbers = str.split("\\s+"); // Розділяємо рядок на числа
    int product = 1;
    for (String num : numbers) {
        product *= Integer.parseInt(num);
    }
    return product;
};

String numberString = "2 3 4";
System.out.println("Product of numbers (2 3 4): " +
multiplyNumbers.apply(numberString));
}
```

Результат виконання:

```

package com.education.ztu;

import java.util.function.*;

public class Main {
    // Задача 2: Власний функціональний інтерфейс Printable
    // Використання лямбда-виразу для реалізації методу print()
    Printable printable = () -> System.out.println("Hello from Printable interface!");
    printable.print();

    // Задача 3a: Predicate для перевірки, чи можна рядок привести до числа
    // Лямбда-вираз для перевірки, чи рядок є числом
    Predicate<String> isNumeric = str -> {

```

Run Main

```

C:\Users\Nastia\jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=54419:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin" -Dfile.encoding=UTF-8
Hello from Printable interface!
Is numeric (123.45): true
Is numeric (NotANumber): false
String is non-empty and numeric (123.45): true
String is non-empty and numeric (empty string): false
Інформатика: Пара почалася о 8:39.
Інформатика: Пара закінчилася о 9:50.
ПРИКЛАД ТЕКСТУ В ВЕРХНЬОМУ РЕГІСТРІ
Product of numbers (2 3 4): 24

Process finished with exit code 0

```

Завдання 4. Stream API.

- Створити стрім з масиву Product з полями name, brand, price, count.
- Отримати всі бренди та вивести в консоль. (map)
- Отримати 2 товари ціна яких менше тисячі. (filter, limit)
- Отримати суму всіх видів товарів, що є на складі. (reduce)
- Згрупувати товари по бренду (Collectors.groupingBy())

					<p style="text-align: center;"><i>ІПТР.420001.123-ЗЛ</i></p>	Арк.
						66
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

- Відсортувати товари за зростанням ціни та повернути масив (sorted, Collectors)
- За бажанням дописати функціонал, що використовує інші методи стрімів.

Лістинг коду:

```
public class StreamApiDemo {
    public static void main(String[] args) {
        List<Product> products = Arrays.asList(
            new Product("Laptop", "Apple", 1500.0, 5),
            new Product("Smartphone", "Samsung", 800.0, 10),
            new Product("Shoes", "Nike", 120.0, 50),
            new Product("Watch", "Casio", 250.0, 30),
            new Product("Tablet", "Apple", 600.0, 15)
        );

        // Завдання 4a: Отримати всі бренди
        System.out.println("Brands:");
        products.stream()
            .map(Product::getBrand) // Отримуємо значення brand для кожного
            .distinct() // Унікальні бренди
            .forEach(System.out::println);

        // Завдання 4b: Отримати 2 товари з ціною менше тисячі
        System.out.println("\nProducts with price < 1000:");
        products.stream()
            .filter(product -> product.getPrice() < 1000) // Фільтруємо
            .limit(2) // Обмежуємо вибір до 2 товарів
            .forEach(System.out::println);

        // Завдання 4c: Отримати суму всіх видів товарів на складі
        int totalCount = products.stream()
            .map(Product::getCount) // Отримуємо кількість кожного товару
            .reduce(0, Integer::sum); // Сумуємо всі кількості
        System.out.println("\nTotal count of products: " + totalCount);

        // Завдання 4d: Згрупувати товари по бренду
        System.out.println("\nProducts grouped by brand:");
        Map<String, List<Product>> groupedByBrand = products.stream()
            .collect(Collectors.groupingBy(Product::getBrand)); // Групуємо
            groupedByBrand.forEach((brand, productList) -> {
                System.out.println(brand + ": " + productList); // Виводимо бренд та
            });

        // Завдання 4e: Відсортувати товари за зростанням ціни та повернути масив
        System.out.println("\nProducts sorted by price:");
        List<Product> sortedByPrice = products.stream()
            .sorted(Comparator.comparingDouble(Product::getPrice)) // Сортуємо
            .collect(Collectors.toList()); // Перетворюємо в список
            sortedByPrice.forEach(System.out::println);
```

					IPTP.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		67

```
// Клас Product
class Product {
    private String name;
    private String brand;
    private double price;
    private int count;

    public Product(String name, String brand, double price, int count) {
        this.name = name;
        this.brand = brand;
        this.price = price;
        this.count = count;
    }

    public String getName() {
        return name;
    }

    public String getBrand() {
        return brand;
    }

    public double getPrice() {
        return price;
    }

    public int getCount() {
        return count;
    }

    // Перевизначений метод toString для зручного виводу об'єкта
    @Override
    public String toString() {
        return String.format("%s (%s) - $%.2f, count: %d", name, brand, price,
count);
    }
}
```

Завдання 5. Посилання на методи чи конструктори.

В попередньому завданні, де це можливо, виклики переробити на посилання на методи чи конструктори

Лістинг коду:

```
// Завдання 5: Посилання на методи та конструктори
System.out.println("\nUsing method references:");
products.stream()
    .map(Product::toString) // Використовуємо метод toString для кожного
продукту
    .forEach(System.out::println);
```

					IPTP.420001.123-3Л	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 6. Використання Optional та його методів.

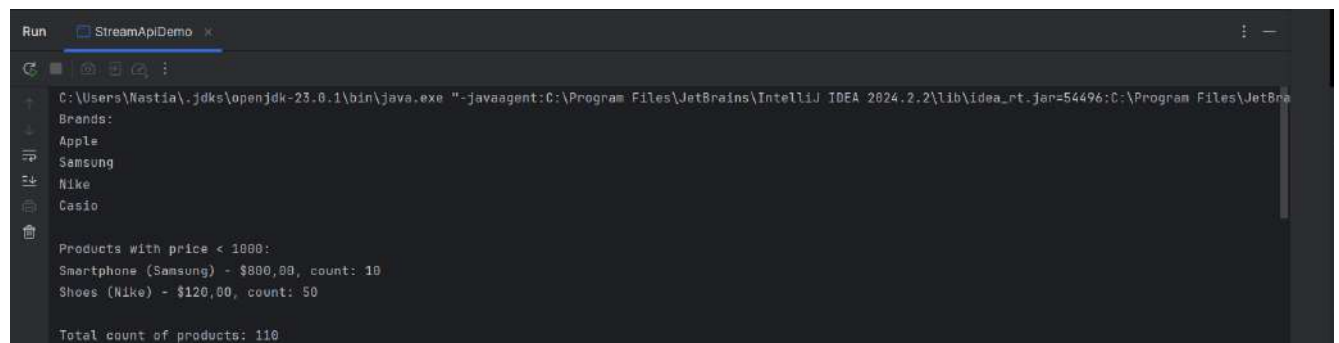
Знайти максимальне значення з масиву чисел, в іншому випадку повернути рядок «Числа відсутні».

Лістинг коду:

```
// Завдання 6: Optional для пошуку максимуму
System.out.println("\nFind max value from array:");
Integer[] numbers = {2, 10, 15, 7, 22};
Optional<Integer> maxNumber = Arrays.stream(numbers)
    .max(Integer::compareTo); // Знаходимо максимальне число
System.out.println("Max value: " + maxNumber.orElseGet(() -> {
    System.out.println("No numbers present.");
    return -1;
}));

// Приклад для порожнього масиву
Integer[] emptyArray = {};
String result = Arrays.stream(emptyArray)
    .max(Integer::compareTo) // Знаходимо максимум
    .map(String::valueOf) // Перетворюємо на рядок
    .orElse("Числа відсутні"); // Значення за замовчуванням
System.out.println("Max value (empty array): " + result);
}
```

Результат виконання:



```
Run StreamApiDemo x
C:\Users\Nastia\.jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=54496:C:\Program Files\JetBra
Brands:
Apple
Samsung
Nike
Casio
Products with price < 1000:
Smartphone (Samsung) - $800,00, count: 10
Shoes (Nike) - $120,00, count: 50
Total count of products: 110
```

					ІПТР.420001.123-ЗЛ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```
Run StreamApiDemo x
Products grouped by brand:
Nike: [Shoes (Nike) - $120,00, count: 50]
Apple: [Laptop (Apple) - $1500,00, count: 5, Tablet (Apple) - $600,00, count: 15]
Casio: [Watch (Casio) - $250,00, count: 30]
Samsung: [Smartphone (Samsung) - $800,00, count: 10]

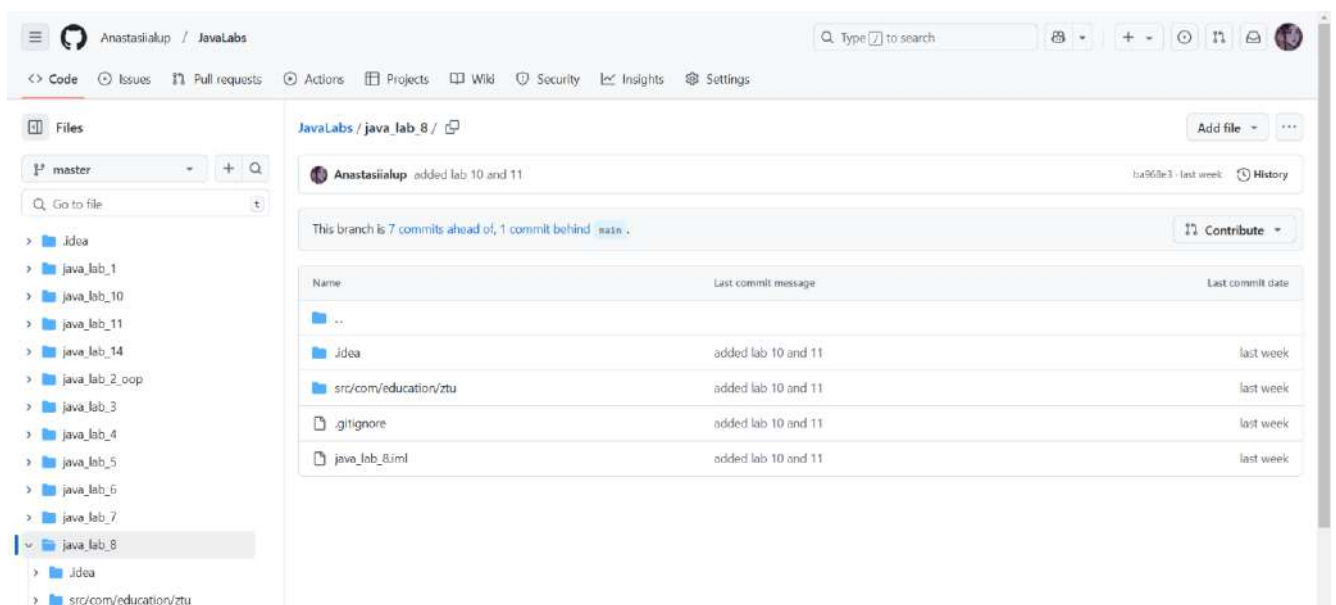
Products sorted by price:
Shoes (Nike) - $120,00, count: 50
Watch (Casio) - $250,00, count: 30
Tablet (Apple) - $600,00, count: 15
Smartphone (Samsung) - $800,00, count: 10
Laptop (Apple) - $1500,00, count: 5

Using method references:
Laptop (Apple) - $1500,00, count: 5
Smartphone (Samsung) - $800,00, count: 10
Shoes (Nike) - $120,00, count: 50
Watch (Casio) - $250,00, count: 30
Tablet (Apple) - $600,00, count: 15

Find max value from array:
Max value: 22
Max value (empty array): Числа відсутні

Process finished with exit code 0
```

Завдання 7. В GitLab проекті Java_labs_ztu, створити директорію Lab_8 та запусити в Lab_8 виконану лабораторну роботу. Надати доступ для перевірки викладачу.



Висновок:

Лабораторна робота №8 з Java допомогла освоїти функціональні інтерфейси та лямбда-вирази, використання Stream API для роботи з колекціями, а також методи

					IPTP.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		70

посилання та клас `Optional`. Це сприяло покращенню навичок обробки даних і спрощення коду, що робить програмування більш ефективним і зрозумілим.

					ІТР.420001.123-ЗЛ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №9

Тема: Регулярні вирази. Рефлексія. Анотації.

Мета роботи: практика роботи з регулярними виразами, використання рефлексії, створення власних анотацій.

Завдання 1. Створити консольний Java проект `java_lab_9` з пакетом `com.education.ztu`.

Завдання 2. Робота з регулярними виразами:

- Використати власний текст, що містить дані 5-10 співробітників компанії (ПІБ, вік, посада, досвід роботи, адреса, емайл, телефон і т. д.)
- Знайти в тексті всі номери телефонів та емайли.
- Змінити формати відображення дат народження (наприклад: 20.05.1995 на 1995-05-20)
- Змінити посади кільком співробітникам.
- Результати роботи відобразити в консолі.

Лістинг коду:

```
package com.education.ztu;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class RegexDemo {
    public static void main(String[] args) {
        String text = """
            John Doe, 30, Software Engineer, 5 years, New York,
            john.doe@example.com, +1-123-456-7890
            Jane Smith, 25, Data Analyst, 2 years, Los Angeles,
            jane.smith@example.com, +1-987-654-3210
            Mike Johnson, 40, Project Manager, 15 years, Chicago,
            mike.johnson@example.com, +1-555-444-3333
            Sarah Brown, 28, UX Designer, 4 years, Boston,
            sarah.brown@example.com, +1-666-777-8888
            """;

        // 1. Знайти всі номери телефонів та email
        Pattern emailPattern = Pattern.compile("\\b[\\w.-]+@[\\w.-]+\\.\\w{2,}\\b");
        Pattern phonePattern = Pattern.compile("\\+\\d-\\d{3}-\\d{3}-\\d{4}");

        // Пошук та виведення email
```


Завдання 3. Робота з користувацьким класом методами Reflection

API:

- Створити власний клас в якому міститимуться публічні та приватні поля, конструктори і методи з аргументами та без.
- Отримати об'єкт класу Class для користувацького класу трьома способами.
- Отримати всі поля, методи, конструктори, що визначені тільки в цьому класі (не враховувати наслідування) та вивести ці значення в консоль (назву, типи параметрів та значення, що повертається).
- Створити екземпляр класу.
- Викликати метод класу.
- Попрацювати з приватним полем (встановити та отримати значення)
- Результати роботи відобразити в консолі.

Лістинг коду:

```
package com.education.ztu;

import java.lang.reflect.*;

public class ReflectionDemo {
    public static void main(String[] args) throws Exception {
        // 1. Отримати об'єкт класу Class трьома способами
        // 1.1. Через клас
        Class<MyClass> class1 = MyClass.class;
        // 1.2. Через екземпляр об'єкта
        MyClass instance = new MyClass();
        Class<? extends MyClass> class2 = instance.getClass();
        // 1.3. Через повне ім'я класу
        Class<?> class3 = Class.forName("com.education.ztu.MyClass");

        // 2. Отримати всі поля класу
        System.out.println("Fields:");
        for (Field field : class1.getDeclaredFields()) {
            // Вивести ім'я поля та його тип
            System.out.println(field.getName() + " (" + field.getType() + ")");
        }

        // 3. Отримати всі методи класу
        System.out.println("\nMethods:");
        for (Method method : class1.getDeclaredMethods()) {
            // Вивести ім'я методу та тип, який він повертає
            System.out.println(method.getName() + " (Return type: " +
                method.getReturnType() + ")");
        }
    }
}
```

```

// 4. Отримати всі конструктори класу
System.out.println("\nConstructors:");
for (Constructor<?> constructor : class1.getDeclaredConstructors()) {
    // Вивести ім'я конструктора
    System.out.println(constructor.getName());
}

// 5. Створити екземпляр класу через рефлексію
// Знаходимо конструктор, який приймає рядок
Constructor<MyClass> constructor = class1.getConstructor(String.class);
// Створюємо новий екземпляр з параметром
MyClass newInstance = constructor.newInstance("Hello Reflection");

// 6. Викликати метод класу через рефлексію
// Знаходимо метод "sayHello"
Method publicMethod = class1.getDeclaredMethod("sayHello");
// Викликаємо метод на новоствореному екземплярі
publicMethod.invoke(newInstance);

// 7. Працювати з приватним полем через рефлексію
// Знаходимо приватне поле "privateField"
Field privateField = class1.getDeclaredField("privateField");
// Даємо доступ до приватного поля
privateField.setAccessible(true);
// Змінюємо значення поля
privateField.set(newInstance, 42);
// Отримуємо нове значення поля та виводимо його
System.out.println("Private field value: " +
privateField.get(newInstance));
}
}

class MyClass {
    public String publicField = "Public Field";
    private int privateField;
    public MyClass() {
    }

    // Конструктор з параметром
    public MyClass(String message) {
        System.out.println("Constructor called with message: " + message);
    }

    // Публічний метод
    public void sayHello() {
        System.out.println("Hello from MyClass!");
    }
}

```

Результат виконання:

					ІПТР.420001.123-ЗЛ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Run ReflectionDemo x
C:\Users\Nastia\.jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=54843:C:\Program Files\JetBra
Fields:
publicField (class java.lang.String)
privateField (int)
Methods:
sayHello (Return type: void)
Constructors:
com.education.ztu.MyClass
com.education.ztu.MyClass
Constructor called with message: Hello Reflection
Hello from MyClass!
Private field value: 42
Process finished with exit code 0

```

Завдання 4. Створення власної анотації:

- Створити власну анотацію, задати їй необхідні поля та значення за замовчуванням для них.
- Встановити їй обмеження застосування через анотацію `@Target`
- Встановити їй політику утримання через анотацію `@Retention`
- Додати анотацію до відповідного об'єкту в коді.
- Отримати дані анотації з об'єкту та вивести в консоль.

Лістинг коду:

```

package com.education.ztu;

import java.lang.annotation.*;
import java.lang.reflect.Method;

// Оголошення анотації CustomAnnotation
@Retention(RetentionPolicy.RUNTIME) // Анотація буде доступна на етапі виконання
@Target(ElementType.METHOD) // Анотація застосована тільки до методів
@interface CustomAnnotation {
    String value() default "Default Value"; // Створення параметра анотації з
дефолтним значенням
    int version() default 1; // Створення параметра анотації з дефолтним значенням
}

// Головний клас, де буде використовуватись рефлексія для отримання анотацій
public class AnnotationDemo {
    public static void main(String[] args) throws Exception {
        // Отримуємо метод 'annotatedMethod' з класу AnnotatedClass
        Method method = AnnotatedClass.class.getMethod("annotatedMethod");

        // Перевіряємо, чи є на цьому методі анотація CustomAnnotation
        if (method.isAnnotationPresent(CustomAnnotation.class)) {
            // Якщо анотація є, отримуємо її
        }
    }
}

```

```

        CustomAnnotation annotation =
method.getAnnotation(CustomAnnotation.class);

        // Виводимо значення параметрів анотації
        System.out.println("Annotation value: " + annotation.value()); //
Виведе "Custom Value"
        System.out.println("Annotation version: " + annotation.version()); //
Виведе 2
    }
}

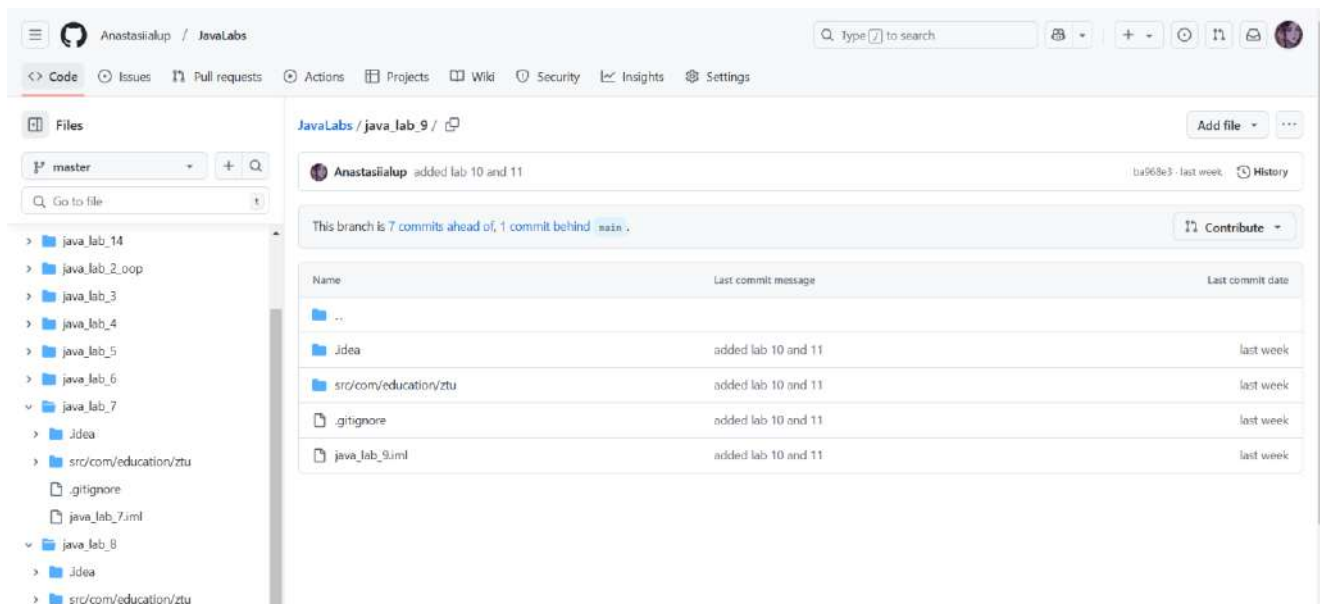
// Клас, у якому метод має анотацію CustomAnnotation
class AnnotatedClass {
    // Застосовуємо анотацію до методу
    @CustomAnnotation(value = "Custom Value", version = 2)
    public void annotatedMethod() {
        System.out.println("This is an annotated method.");
    }
}

```

Результат виконання:

Завдання 5. В GitLab проекті Java_labs_ztu, створити директорію Lab_9 та запусити в Lab_9 виконану лабораторну роботу. Надати доступ для перевірки викладачу.

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		77



Висновок:

У дев'ятій лабораторній роботі вивчались регулярні вирази, рефлексія та анотації в Java. Завдання включали пошук і заміну тексту за допомогою регулярних виразів, динамічне отримання інформації про класи через рефлексію та створення власних анотацій для розширення функціональності програми. Це дозволяє працювати з метаданими та створювати більш гнучкі програми.

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		78

Лабораторна робота №10

Тема: Серіалізація. Логування. Документування коду.

XML та JSON парсери.

Мета роботи: практика роботи з XML та JSON парсерами, використання серіалізації, логування та документування коду.

Завдання 1. Створити maven Java проєкт `java_lab_10` з пакетом `com.education.ztu`.

Додати в проєкт код з лабораторної роботи №3 з пакету `game`. Для реалізації завдань додати необхідні залежності в файл `pom.xml`.

Завдання 2. Серіалізація:

- Додати до сутностей в пакеті `game` `serialVersionUID` (згенерувати за допомогою IntelliJ IDEA)
- Виключити деякі поля з серіалізації на власний розсуд (використати ключове слово `transient`)
- Серіалізувати та десеріалізувати сутності.

Лістинг коду:

```
package com.education.ztu;

import com.education.ztu.game.Game;
import com.education.ztu.game.Team;
import com.education.ztu.game.Player;
import com.google.gson.Gson;
import org.apache.log4j.Logger;

import java.io.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
```

```
# Root logger configuration
log4j.rootLogger=DEBUG, console, file

# Console appender
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
```

					ІПТР.420001.123-ЗЛ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

```
%c{1}:%L - %m%n

# File appender
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.file=logs/game.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L
- %m%n

# Logger levels
log4j.logger.com.education.ztu.game=DEBUG
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.education.ztu</groupId>
  <artifactId>java lab 10</artifactId>
  <version>1.0-SNAPSHOT</version>
  <dependencies>
    <!-- JSON Parsing -->
    <dependency>
      <groupId>com.google.code.gson</groupId>
      <artifactId>gson</artifactId>
      <version>2.8.9</version>
    </dependency>
    <!-- Log4J -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-api</artifactId>
      <version>2.0.9</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
      <version>2.0.9</version>
    </dependency>
  </dependencies>
</project>
```

```
System.out.println("=== Завдання 2: Сепіалізація ===");
try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("game.ser"));
     ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("game.ser"))) {
  oos.writeObject(game);
  logger.info("Game serialized to file game.ser");

  Game deserializedGame = (Game) ois.readObject();
  System.out.println("Deserialized Game: " + deserializedGame);
  logger.info("Game deserialized: " + deserializedGame);
} catch (IOException | ClassNotFoundException e) {
  logger.error("Serialization error", e);
}
System.out.println("Завдання 2: Completed\n");
```

Результат виконання:

					ІПТР.420001.123-ЗЛ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		


```

Run Main x
C:\Users\Mostie\jdk\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=56780:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin" -Didea.config.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\conf -Didea.copyright.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\copyright -Didea.home.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin -Didea.platform.prefix=Java -Didea.vendor.id=idea -Djava.awt.headless=true -Djava.class.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin\idea_rt.jar -Djava.class.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin\idea_rt.jar -Djava.class.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin\idea_rt.jar
=== Завдання 1: Ініціалізація гри ===
Game{name='Adventure', team=Team{teamName='Warriors', players=[Player{name='Alice', score=100}]}}
2024-12-23 04:11:41 INFO Main:24 - Game initialized: Game{name='Adventure', team=Team{teamName='Warriors', players=[Player{name='Alice', score=100}]}}
Завдання 1: Completed

```

Завдання 3. Логування:

- Додати логування до коду в пакеті game. Використати бібліотеки Log4J, SLF4J.
- Вивести логи в консоль та в файл.
- Використати різні рівні логування (trace, debug, info, warn, error, fatal)

```

System.out.println("=== Завдання 3: Логування ===");
logger.debug("Debug log example");
logger.warn("Warning log example");
logger.error("Error log example");
System.out.println("Логи записані в консоль і файл.");
System.out.println("Завдання 3: Completed\n");

```

Результат виконання:

```

Run Main x
=== Завдання 2: Сериалізація ===
2024-12-23 04:11:41 INFO Main:31 - Game serialized to file game.ser
Deserialized Game: Game{name='Adventure', team=Team{teamName='Warriors', players=[Player{name='Alice', score=100}]}}
2024-12-23 04:11:41 INFO Main:35 - Game deserialized: Game{name='Adventure', team=Team{teamName='Warriors', players=[Player{name='Alice', score=100}]}}
Завдання 2: Completed

```

Завдання 4. Документування коду:

- Додати документаційні коментарі до коду в пакеті game
- Згенерувати документацію (щоб згенерувати JavaDoc у IntelliJ IDEA необхідно натиснути Tools → Generate JavaDoc → вказати шлях, куди зберегти документацію)

Результат JavaDoc:

					ІПТР.420001.123-ЗЛ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

Overview PACKAGE TREE INDEX SEARCH HELP	
com.education.ztu.game	
Contents	Package com.education.ztu.game
Description	package com.education.ztu.game
Related Packages	Related Packages
Classes and Interfaces	Package Description
	com.education.ztu
	Classes
	Class Description
	Game Represents a game entity.
	GameLogger Handles logging for the game package.
	JsonParser Handles JSON serialization and deserialization.
	Player
	Team Represents a team in the game.
	XmlParser

Завдання 5. XML парсери:

- Реалізувати читання та збереження XML файлу використовуючи DOM парсер.
- XML файл використати будь який за бажанням.

```
System.out.println("=== Завдання 4: XML Парсинг ===");
try {
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = factory.newDocumentBuilder();

    // Запис у XML
    String xmlContent = "<?xml version='1.0' encoding='UTF-8'?'>\n"
        + "<game name='Adventure'\n"
        + "    <team name='Warriors'\n"
        + "        <player\n"
        + "            <name>Alice</name>\n"
        + "            <score>100</score>\n"
        + "        </player>\n"
        + "    </team>\n"
        + "</game>";

    try (FileWriter writer = new FileWriter("example.xml")) {
        writer.write(xmlContent);
    }

    String xmlFilePath = "example.xml"; // Додано змінну для шляху до файлу XML
    logger.info("XML written to file: " + xmlFilePath);

    // Читання з XML
    Document document = builder.parse(new File(xmlFilePath));
    document.getDocumentElement().normalize();
    String gameName = document.getDocumentElement().getAttribute("name");
    String teamName =
document.getElementsByTagName("team").item(0).getAttributes().getNamedItem("name")
.getTextContent();
    String playerName =
document.getElementsByTagName("name").item(0).getTextContent();
    String playerScore =
```

```
document.getElementsByTagName("score").item(0).getTextContent();

    System.out.println("Parsed from XML: Game{name='" + gameName + "', team='" +
teamName + "', player{name='" + playerName + "', score=" + playerScore + "}}");
    logger.info("XML parsed successfully");
} catch (ParserConfigurationException | SAXException | IOException e) {
    logger.error("XML Parsing error", e);
}
System.out.println("Завдання 4: Completed\n");
```

Результат виконання:

Завдання 6. JSON парсер:

- Провести перетворення сутностей з Java в JSON і навпаки з JSON в Java (використайте бібліотеки Gson або Jackson) Сутності для перетворень виберіть на власний розсуд.

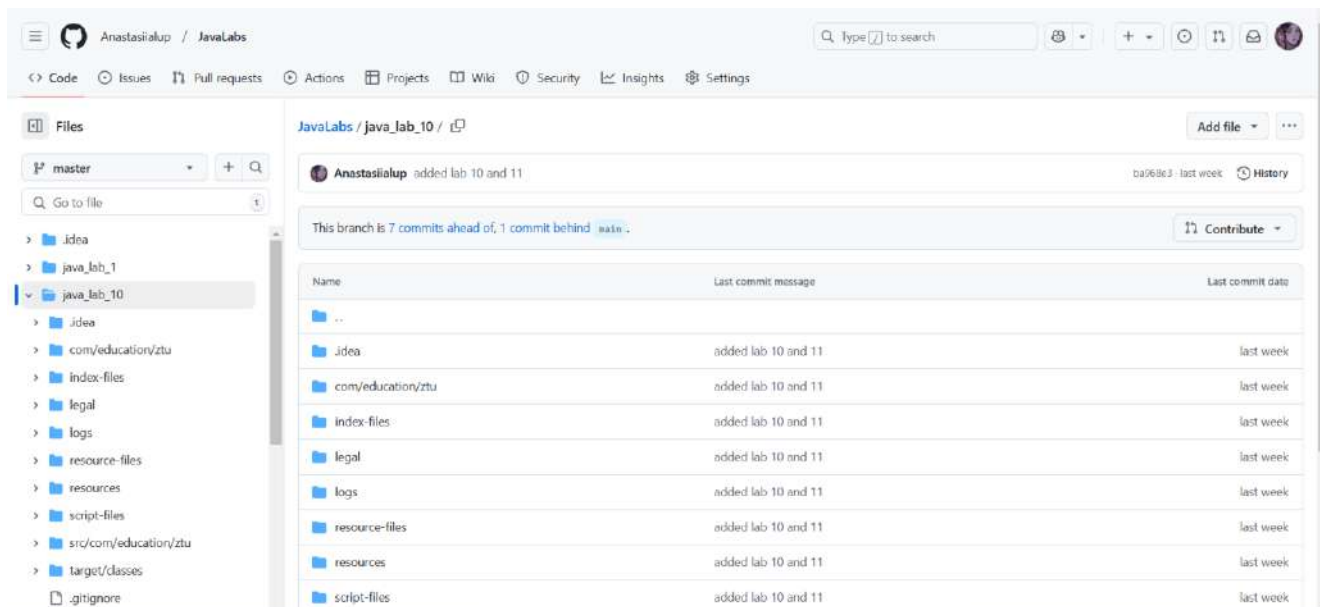
```
System.out.println("=== Завдання 5: JSON Парсинг ===");
Gson gson = new Gson();
String json = gson.toJson(game);
System.out.println("Serialized to JSON: " + json);

Game gameFromJson = gson.fromJson(json, Game.class);
System.out.println("Deserialized from JSON: " + gameFromJson);
logger.info("Game serialized to JSON and back: " + gameFromJson);
System.out.println("Завдання 5: Completed\n");

System.out.println("Усі завдання виконані успішно!");
}
```

Результат виконання:

Завдання 7. В GitLab проекті Java_labs_ztu, створити директорію Lab_10 та запусити в Lab_10 виконану лабораторну роботу. Надати доступ для перевірки викладачу.



Висновок:

У 10-й лабораторній роботі основна увага приділялася серіалізації, логуванню та роботі з різними форматами даних, зокрема XML та JSON. Завдання включали: Серіалізація об'єктів: Використовувались механізми серіалізації для збереження та відновлення стану об'єкта гри, що дозволяє зберігати дані в зовнішньому файлі. Логування: Виведення важливої інформації через систему логів (Log4j), що допомагає в діагностиці помилок і відстеженні процесів.

Парсинг XML та JSON: Розбір та запис даних у форматах XML і JSON, що дозволяє працювати з зовнішніми файлами та обмінюватися даними між системами.

Ця лабораторна робота сприяє розвитку навичок обробки даних у різних форматах, серіалізації та логування в Java-додатках.

Лабораторна робота №11

Тема: Java та бази даних

Мета роботи: набути навичок створення зв'язку Java програми з базою даних та взаємодії з нею в процесі роботи програми.

Завдання 1. Створити консольний Java проект java_lab_11 з пакетом com.education.ztu.

Необхідно реалізувати програму використовуючи DDL та DML оператори SQL для роботи зі списком товарів.

Main.java — основний клас програми, який містить логіку для виконання CRUD операцій над товарами за допомогою класу ProductDAO.

Робить: Створює програму, яка працює з товарними даними за допомогою SQL операторів (DDL, DML).

Лістинг коду:

```
package com.education.ztu;

import com.education.ztu.dao.ProductDAO;
import com.education.ztu.model.Product;

import java.sql.SQLException;

public class Main {
    public static void main(String[] args) {
        ProductDAO productDAO = new ProductDAO();

        try {
            // Create table
            System.out.println("Adding new products...");
            productDAO.create(new Product("Laptop", "Electronics", 1200.99));
            productDAO.create(new Product("Phone", "Electronics", 799.49));

            // Retrieve all products
            System.out.println("Products in the database:");
            productDAO.getAll().forEach(System.out::println);

            // Update product
            Product product = productDAO.read(1);
            if (product != null) {
                product.setPrice(1100.00);
                productDAO.update(product);
            }
        }
    }
}
```

```

        // Delete product
        productDAO.delete(2);
        System.out.println("Products after deletion:");
        productDAO.getAll().forEach(System.out::println);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

Завдання 2. Створити з'єднання з базою даних:

- Обрати базу даних з якою буде працювати ваша програма, скачати для неї JDBC драйвер та додати в проект.
- Створити базу даних store.
- Налаштувати з'єднання (параметри з'єднання повинні бути збережені в properties файлі та зчитуватись з ResourceBundle)

Створення з'єднання з базою даних:

Файли:

DatabaseConnection.java — клас для налаштування з'єднання з базою даних за допомогою JDBC. Читає параметри з db.properties.

Робить: Створює підключення до бази даних, використовуючи параметри з файлу властивостей.

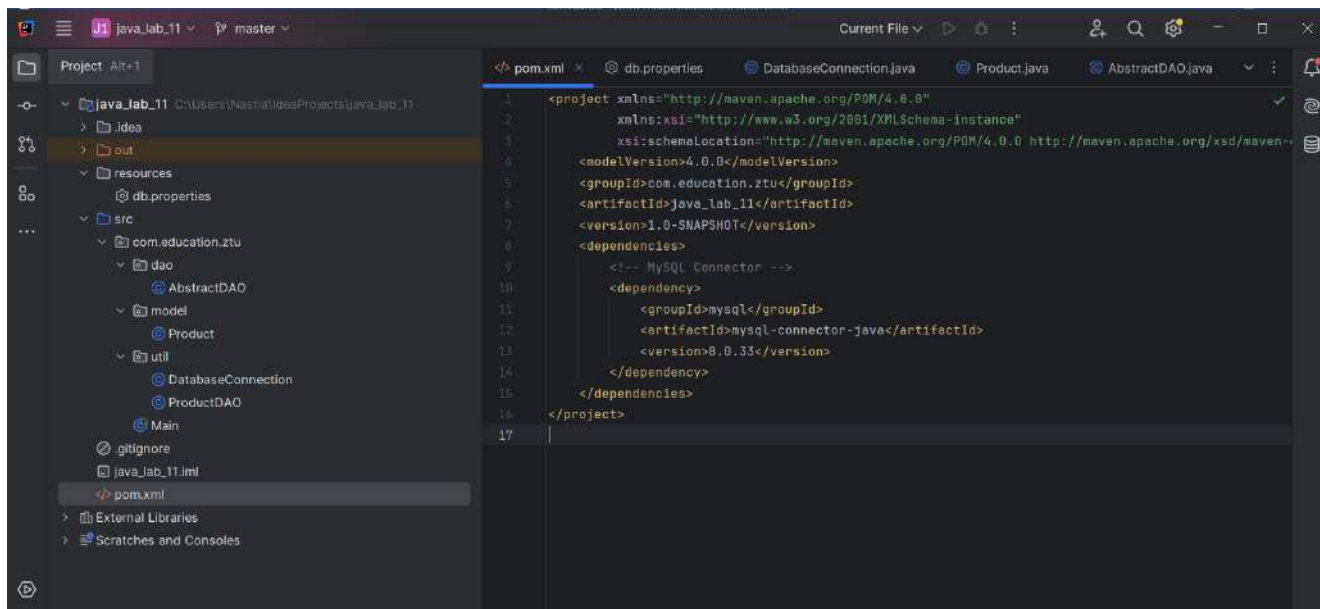
Підключення до БД, я використовую mysql:

```

db.url=jdbc:mysql://localhost:3306/store
db.username=root
db.password=147258369asasASAS

```

					ІТР.420001.123-ЗЛ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		



Лістинг коду:

```
package com.education.ztu.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.ResourceBundle;

public class DatabaseConnection {
    private static Connection connection;

    public static Connection getConnection() {
        if (connection == null) {
            try {
                ResourceBundle bundle = ResourceBundle.getBundle("db");
                String url = bundle.getString("db.url");
                String username = bundle.getString("db.username");
                String password = bundle.getString("db.password");

                connection = DriverManager.getConnection(url, username, password);
            } catch (SQLException e) {
                e.printStackTrace();
                throw new RuntimeException("Failed to establish database
connection");
            }
        }
        return connection;
    }
}
```

Завдання 3. Робота з базою даних використовуючи клас Statement:

- Створити необхідні таблицю чи таблиці в базі даних

					ІРТР.420001.123-ЗЛ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

- Заповнити їх даними для 10 товарів (тут краще використати batch- команди)
- Отримати всі записи з бази з інформацією про товари та вивести їх в консоль.

Робота з базою даних за допомогою класу Statement:

Файли:

ProductDAO.java — клас для роботи з базою даних через JDBC. Використовує Statement для отримання даних та вставлення їх у таблицю.

Робить: Створює таблицю, заповнює її даними для товарів (через batch- виконання) і отримує всі записи з таблиці.

Лістинг коду:

```
package com.education.ztu.dao;

import com.education.ztu.model.Product;
import com.education.ztu.util.DatabaseConnection;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductDAO extends AbstractDAO<Product> {
    private final Connection connection = DatabaseConnection.getConnection();

    @Override
    public void create(Product product) throws SQLException {
        String sql = "INSERT INTO products (name, category, price) VALUES (?, ?, ?)";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setString(1, product.getName());
            ps.setString(2, product.getCategory());
            ps.setDouble(3, product.getPrice());
            ps.executeUpdate();
        }
    }

    @Override
    public Product read(int id) throws SQLException {
        String sql = "SELECT * FROM products WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return new Product(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getString("category"),
                    rs.getDouble("price")
                );
            }
        }
    }
}
```



```

        );
    }
}

return null;
}

@Override
public void update(Product product) throws SQLException {
    String sql = "UPDATE products SET name = ?, category = ?, price = ? WHERE
id = ?";
    try (PreparedStatement ps = connection.prepareStatement(sql)) {
        ps.setString(1, product.getName());
        ps.setString(2, product.getCategory());
        ps.setDouble(3, product.getPrice());
        ps.setInt(4, product.getId());
        ps.executeUpdate();
    }
}

@Override
public void delete(int id) throws SQLException {
    String sql = "DELETE FROM products WHERE id = ?";
    try (PreparedStatement ps = connection.prepareStatement(sql)) {
        ps.setInt(1, id);
        ps.executeUpdate();
    }
}

@Override
public List<Product> getAll() throws SQLException {
    List<Product> products = new ArrayList<>();
    String sql = "SELECT * FROM products";
    try (Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql)) {
        while (rs.next()) {
            products.add(new Product(
                rs.getInt("id"),
                rs.getString("name"),
                rs.getString("category"),
                rs.getDouble("price")
            ));
        }
    }
    return products;
}
}

```

Клас Продукт

```

package com.education.ztu.model;

public class Product {
    private int id;
    private String name;
    private String category;
    private double price;

    public Product(int id, String name, String category, double price) {
        this.id = id;
        this.name = name;
        this.category = category;
    }
}

```

					IPTP.420001.123-3Л	Арк.
						89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        this.price = price;
    }

    public Product(String name, String category, double price) {
        this(0, name, category, price);
    }

    // Getters and setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getCategory() { return category; }
    public void setCategory(String category) { this.category = category; }

    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }

    @Override
    public String toString() {
        return String.format("Product{id=%d, name='%s', category='%s', price=%.2f}", id, name, category, price);
    }
}

```

Завдання 4. Робота з базою даних використовуючи клас PreparedStatement:

- Додати ще 5 товарів використовуючи.
- Отримати дані про товари з певної категорії чи певного бренду та вивести їх в консоль.
- Після цього видалити всі записи з бази.

Робота з базою даних через PreparedStatement:

Файли:

ProductDAO.java (продовження) — реалізація методів для додавання нових товарів за допомогою PreparedStatement, отримання товарів за категорією чи брендом і видалення товарів.

Робить: Додає нові товари, отримує дані про товари за категорією чи брендом, видаляє записи з бази.

Лістинг коду:

```
package com.education.ztu.dao;
```

					ІПТР.420001.123-ЗЛ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import com.education.ztu.model.Product;
import com.education.ztu.util.DatabaseConnection;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductDAO extends AbstractDAO<Product> {
    private final Connection connection = DatabaseConnection.getConnection();

    @Override
    public void create(Product product) throws SQLException {
        String sql = "INSERT INTO products (name, category, price) VALUES (?, ?, ?)";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setString(1, product.getName());
            ps.setString(2, product.getCategory());
            ps.setDouble(3, product.getPrice());
            ps.executeUpdate();
        }

    }

    @Override
    public Product read(int id) throws SQLException {
        String sql = "SELECT * FROM products WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return new Product(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getString("category"),
                    rs.getDouble("price")
                );
            }
        }
        return null;
    }

    @Override
    public void update(Product product) throws SQLException {
        String sql = "UPDATE products SET name = ?, category = ?, price = ? WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setString(1, product.getName());
            ps.setString(2, product.getCategory());
            ps.setDouble(3, product.getPrice());
            ps.setInt(4, product.getId());
            ps.executeUpdate();
        }

    }

    @Override
    public void delete(int id) throws SQLException {
        String sql = "DELETE FROM products WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setInt(1, id);
            ps.executeUpdate();
        }

    }
}

```

					IPTP.420001.123-3Л	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@Override
public List<Product> getAll() throws SQLException {
    List<Product> products = new ArrayList<>();
    String sql = "SELECT * FROM products";
    try (Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql)) {
        while (rs.next()) {
            products.add(new Product(
                rs.getInt("id"),
                rs.getString("name"),
                rs.getString("category"),
                rs.getDouble("price")
            ));
        }
    }
    return products;
}
}

```

Завдання 5. Робота з транзакціями та точками збереження:

Додати два товари (один запит повинен бути з синтаксичними помилками)

Створити точку збереження після додавання першого товару.

Вивести в консоль товари, що були додані після відпрацювання двох запитів.

Робота з транзакціями та точками збереження:

Файли:

ProductDAO.java — додатково до попередніх завдань, додаються транзакції і точка збереження після першого додавання товару.

Робить: Створює точку збереження після додавання одного товару, що дозволяє відкотити зміни, якщо другий товар не додано через синтаксичні помилки.

```

package com.education.ztu.dao;

import com.education.ztu.model.Product;
import com.education.ztu.util.DatabaseConnection;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductDAO extends AbstractDAO<Product> {
    private final Connection connection = DatabaseConnection.getConnection();

    @Override
    public void create(Product product) throws SQLException {

```

```

String sql = "INSERT INTO products (name, category, price) VALUES (?, ?,
?)" ;

try (PreparedStatement ps = connection.prepareStatement(sql)) {
    ps.setString(1, product.getName());
    ps.setString(2, product.getCategory());
    ps.setDouble(3, product.getPrice());
    ps.executeUpdate();
}

@Override
public Product read(int id) throws SQLException {
    String sql = "SELECT * FROM products WHERE id = ?";
    try (PreparedStatement ps = connection.prepareStatement(sql)) {
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return new Product(
                rs.getInt("id"),
                rs.getString("name"),
                rs.getString("category"),
                rs.getDouble("price")
            );
        }
    }
    return null;
}

@Override
public void update(Product product) throws SQLException {
    String sql = "UPDATE products SET name = ?, category = ?, price = ? WHERE
id = ?";
    try (PreparedStatement ps = connection.prepareStatement(sql)) {
        ps.setString(1, product.getName());
        ps.setString(2, product.getCategory());
        ps.setDouble(3, product.getPrice());
        ps.setInt(4, product.getId());
        ps.executeUpdate();
    }
}

@Override
public void delete(int id) throws SQLException {
    String sql = "DELETE FROM products WHERE id = ?";
    try (PreparedStatement ps = connection.prepareStatement(sql)) {
        ps.setInt(1, id);
        ps.executeUpdate();
    }
}

@Override
public List<Product> getAll() throws SQLException {
    List<Product> products = new ArrayList<>();
    String sql = "SELECT * FROM products";
    try (Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql)) {
        while (rs.next()) {
            products.add(new Product(
                rs.getInt("id"),
                rs.getString("name"),
                rs.getString("category"),
                rs.getDouble("price")
            ));
        }
    }
}

```

```

    });
}
}
return products;
}
}

```

Завдання 6. Реалізація взаємодії з базою даних товарів використовуючи DAO:

Створити абстрактний клас AbstractDAO з абстрактними методами для подальшої реалізації CRUD операцій.

Реалізувати ProductDAO, що наслідується від AbstractDAO та імплементувати необхідні методи.

В тестовому класі продемонструвати взаємодію з базою даних використовуючи ProductDAO.

Реалізація взаємодії з базою даних через DAO:

Файли:

AbstractDAO.java — абстрактний клас для визначення CRUD методів для роботи з будь-якими сутностями в базі даних.

ProductDAO.java — конкретна реалізація DAO для товарів, яка імплементує методи CRUD для роботи з таблицею товарів.

Робить: Використовує шаблон проектування DAO для забезпечення стандартних операцій над товарними даними.

Лістинг коду:

```

package com.education.ztu.dao;

import java.sql.SQLException;
import java.util.List;

public abstract class AbstractDAO<T> {
    public abstract void create(T entity) throws SQLException;
    public abstract T read(int id) throws SQLException;
    public abstract void update(T entity) throws SQLException;
    public abstract void delete(int id) throws SQLException;
    public abstract List<T> getAll() throws SQLException;
}

```

```

package com.education.ztu.dao;

import com.education.ztu.model.Product;
import com.education.ztu.util.DatabaseConnection;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductDAO extends AbstractDAO<Product> {
    private final Connection connection = DatabaseConnection.getConnection();

    @Override
    public void create(Product product) throws SQLException {
        String sql = "INSERT INTO products (name, category, price) VALUES (?, ?, ?)";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setString(1, product.getName());
            ps.setString(2, product.getCategory());
            ps.setDouble(3, product.getPrice());
            ps.executeUpdate();
        }

    }

    @Override
    public Product read(int id) throws SQLException {
        String sql = "SELECT * FROM products WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return new Product(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getString("category"),
                    rs.getDouble("price")
                );
            }
            return null;
        }

    }

    @Override
    public void update(Product product) throws SQLException {
        String sql = "UPDATE products SET name = ?, category = ?, price = ? WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setString(1, product.getName());
            ps.setString(2, product.getCategory());
            ps.setDouble(3, product.getPrice());
            ps.setInt(4, product.getId());
            ps.executeUpdate();
        }

    }

    @Override
    public void delete(int id) throws SQLException {
        String sql = "DELETE FROM products WHERE id = ?";
        try (PreparedStatement ps = connection.prepareStatement(sql)) {
            ps.setInt(1, id);
            ps.executeUpdate();
        }

    }
}

```

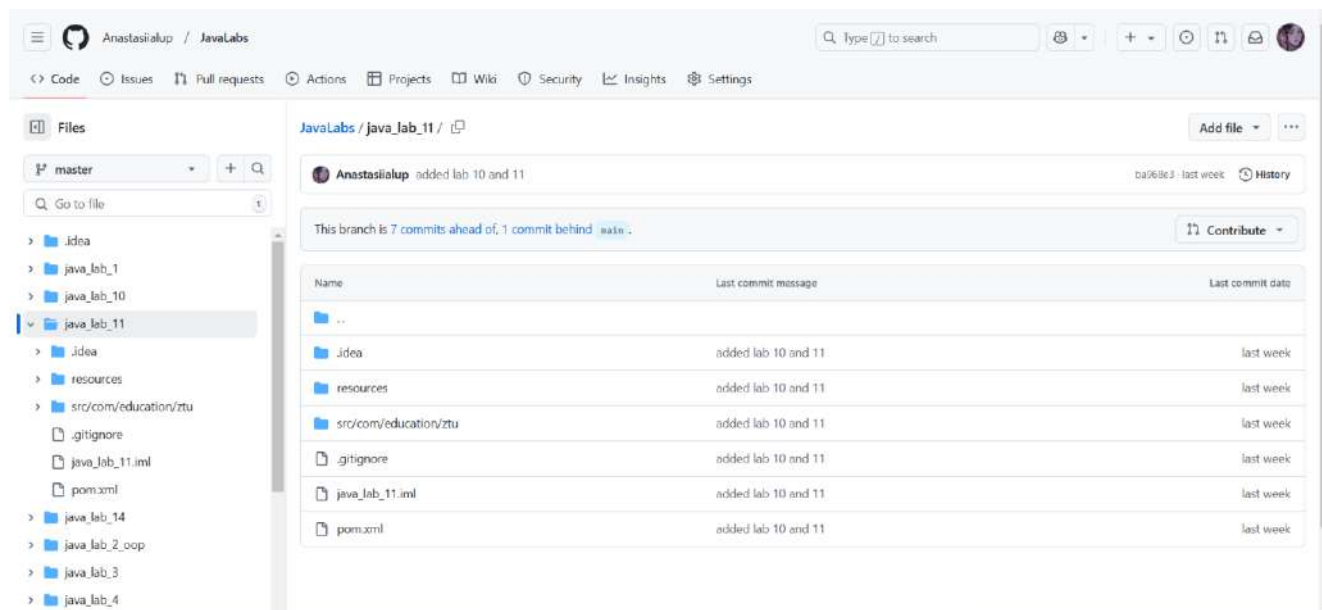
```

    }

    @Override
    public List<Product> getAll() throws SQLException {
        List<Product> products = new ArrayList<>();
        String sql = "SELECT * FROM products";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                products.add(new Product(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getString("category"),
                    rs.getDouble("price")
                ));
            }
        }
        return products;
    }
}

```

Завдання 7. В GitLab проекті Java_labs_ztu, створити директорію Lab_11 та запусити в Lab_11 виконану лабораторну роботу. Надати доступ для перевірки викладачу.



Висновок:

Лабораторна робота №11 дала змогу набути практичних навичок роботи з базами даних в Java. В рамках роботи було здійснено підключення до бази даних, створено таблиці для зберігання товарів та виконано операції додавання,

					IPTP.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		96

оновлення, читання і видалення записів за допомогою SQL запитів. Також були реалізовані механізми для роботи з транзакціями та точками збереження.

Створення абстрактного класу для DAO та реалізація конкретного класу для роботи з товарами дозволило покращити структуру програми та реалізувати CRUD операції.

Загалом, лабораторна робота дозволила не лише зрозуміти основи взаємодії Java з базами даних, а й впровадити концепції ООП та принципи проектування програмного забезпечення.

					ІТР.420001.123-ЗЛ	Арк.
						97
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №14

Тема: Spring

Мета роботи: практика роботи з Spring Framework

Завдання 1. Створити проект на основі Git репозиторію

https://git.ztu.edu.ua/kipz_pvi/spring

Завдання 2. Створити базу даних та налаштувати з'єднання з нею в файлі application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/todo
spring.datasource.username=user
spring.datasource.password=password

flyway.user=user
flyway.password=password
flyway.schemas=todo
flyway.url=jdbc:mysql://localhost:3306/todo

todo.title=TODO items
```

У файлі application.properties налаштовано підключення до бази даних MySQL (todo) та облікові дані (user, password).

Додано параметри Flyway для міграцій (flyway.user, flyway.schemas, flyway.url).

Параметр todo.title задає заголовок для відображення елементів TODO.

Підключення до бази даних налаштовано, Flyway готовий до міграцій.

Завдання 3. Робота з Spring WebMVC:

Створення нового контролера:

- Контролер дозволяє редагувати та видаляти завдання класу TodoItem.
- Використовуються анотації @Controller, @PostMapping, @DeleteMapping.
- Отримання параметрів запиту реалізовано через @PathParam та @ModelAttribute.
- Реалізовано переадресацію та перенаправлення запитів.
- Додано обробку помилок через @ExceptionHandler.

Код контролера:

```
package com.education.ztu.spring.controller;
```

```
import com.education.ztu.spring.entity.TODOItem;
import com.education.ztu.spring.service.TODOService;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
```

```
@Controller("/todo")
```

```
public class TODOController {
```

					ІПТР.420001.123-ЗЛ	Арк.
						98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

private final TodoService todoService;

public TodoController(TodoService todoService) {
    this.todoService = todoService;
}

@PostMapping("/edit")
public String editTodoItem(@ModelAttribute TodoItem item) {
    todoService.updateTodoItem(item);
    return "redirect:/";
}

@DeleteMapping("/delete/{id}")
public String deleteTodoItem(@PathVariable Long id) {
    todoService.deleteTodoItem(id);
    return "redirect:/";
}

@ExceptionHandler(Exception.class)
public String handleException(Exception e, Model model) {
    model.addAttribute("error", e.getMessage());
    return "error.html";
}
}

```

Завдання 4. Робота з Thymeleaf

1. Створення сторінки для редагування об'єкту

- Додано сторінку edit.html для редагування завдань.
- Внесено зміни у файл index.html для додавання посилань на сторінки редагування та видалення.

Код сторінки edit.html:

```

<!doctype html>
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Todo Item</title>
</head>
<body>
    <h1>Edit Todo Item</h1>
    <form method="POST" th:action="@{/todo/edit}" th:object="${todo}">

```

```

        <input type="hidden" th:field="*{id}" />
        <label for="data">Task:</label>
        <input type="text" id="data" th:field="*{data}" />
        <label for="completed">Completed:</label>
        <input type="checkbox" id="completed" th:field="*{completed}" />
        <button type="submit">Save</button>
    </form>
</body>
</html>

```

Зміни в index.html:

```

<li class="list-group-item border-0 d-flex align-items-center ps-0" th:each="t:
${todos}">
    <input class="form-check-input me-3" type="checkbox"
th:checked="*{t.completed}" disabled />
    <span th:text="*{t.data}"></span>
    <a th:href="@{/todo/edit(id=*{t.id})}" class="btn btn-primary ms-2">Edit</a>
    <form th:action="@{/todo/delete/{id}(id=*{t.id})}" method="post" class="d-
inline">
        <input type="hidden" name="_method" value="delete" />
        <button type="submit" class="btn btn-danger">Delete</button>
    </form>
</li>

```

Завдання 5. Робота з Spring Data JDBC

1. Реалізація логіки видалення та оновлення статусу завдань

- Додано методи до TodoItemRepository для видалення та оновлення завдань.

Зміни у TodoItemRepository:

```

@Repository
public interface TodoItemRepository extends CrudRepository<TodoItem, Long> {

    @Modifying
    @Query("UPDATE todo_items SET name = :data, is_completed = :completed
WHERE id = :id")
    void updateTodoItem(@Param("id") Long id, @Param("data") String data,
@Param("completed") Boolean completed);

    @Modifying
    @Query("DELETE FROM todo_items WHERE id = :id")
    void deleteTodoItemById(@Param("id") Long id);
}

```

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		

```
}
```

Завдання 6. Робота з @Component, @Service, @Bean

1. Оновлення TodoService

- Додано логіку видалення та оновлення завдань.

Зміни у TodoService:

@Component

```
public class TodoService {
```

```
    private final TodoItemRepository todoItemRepository;
```

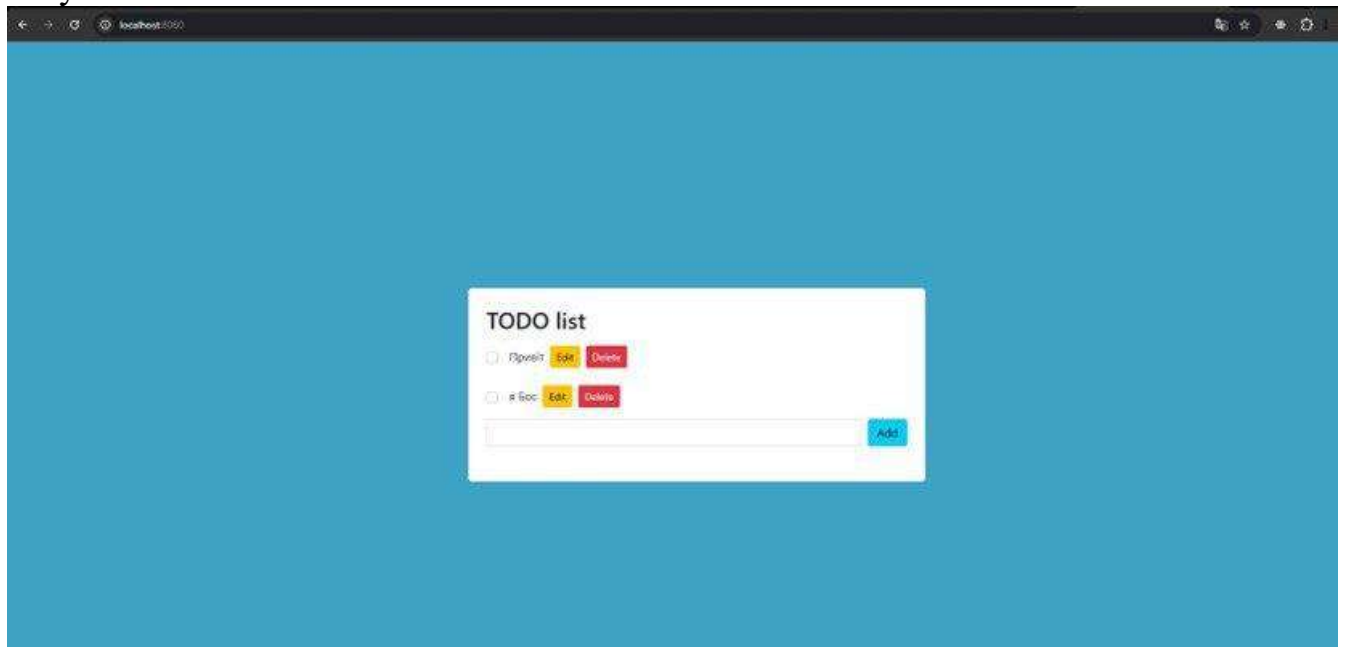
```
    public TodoService(TodoItemRepository todoItemRepository) {  
        this.todoItemRepository = todoItemRepository;  
    }
```

```
    public void updateTodoItem(TodoItem todoItem) {  
        todoItemRepository.updateTodoItem(todoItem.getId(), todoItem.getData(),  
        todoItem.getCompleted());  
    }
```

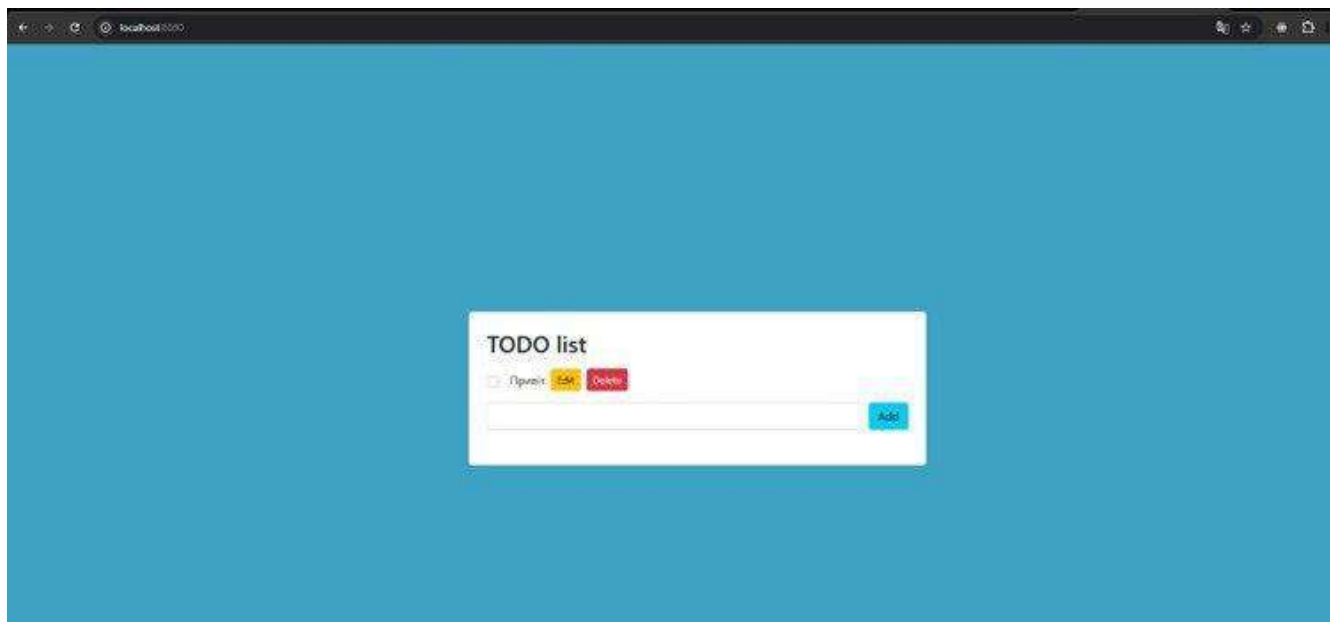
```
    public void deleteTodoItem(Long id) {  
        todoItemRepository.deleteTodoItemById(id);  
    }
```

```
}
```

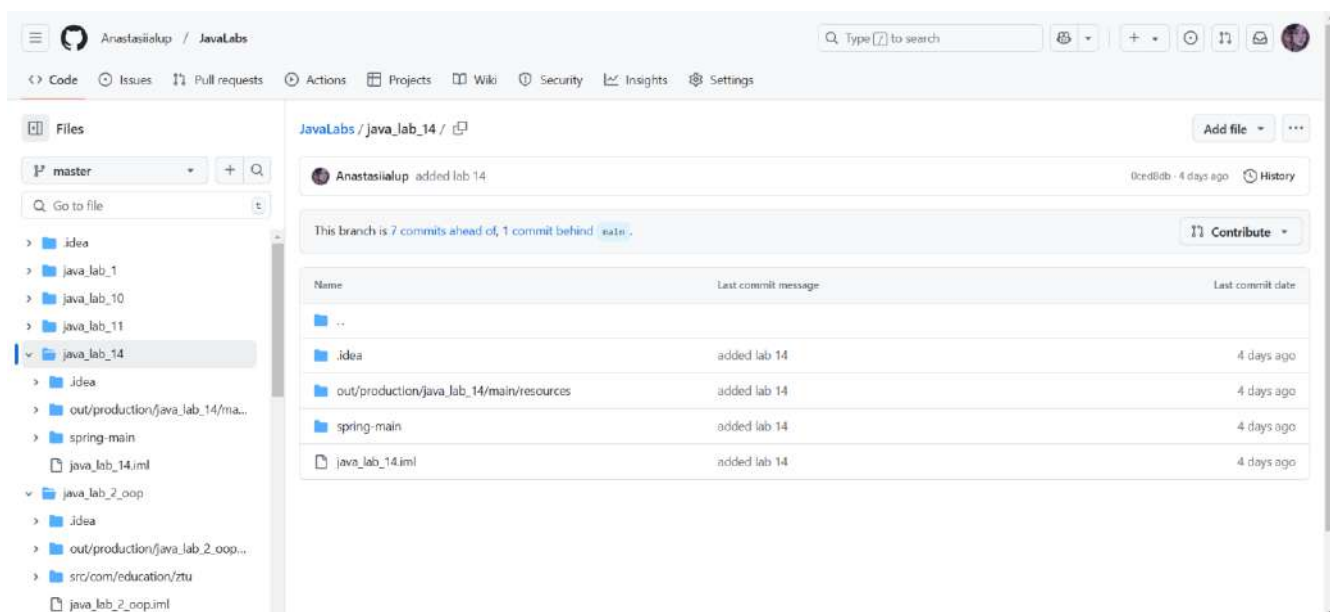
Результат виконання:



					ІПТР.420001.123-3Л	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		



Завдання 6. В GitLab проекті Java_labs_ztu, створити директорію Lab_14 та запустити в Lab_14 виконану лабораторну роботу. Надати доступ для перевірки викладачу.



Висновок

В ході виконання лабораторної роботи реалізовано веб-додаток для управління списком завдань з використанням Spring Boot. Впроваджено CRUD-функціональність, роботу з Thymeleaf для відображення та редагування завдань, а також обробку помилок. Реалізація забезпечує основні вимоги для роботи з TODO List.

					ІПТР.420001.123-ЗЛ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		