

## Зміст

Лабораторна робота № 1 .....	2
Лабораторна робота № 2 .....	7
Лабораторна робота № 3 .....	12
Лабораторна робота № 4 .....	18
Лабораторна робота № 5 .....	26
Лабораторна робота № 6 .....	32
Лабораторна робота № 7 .....	42
Лабораторна робота № 8 .....	51
Лабораторна робота № 9 .....	60
Лабораторна робота № 10 .....	67
Лабораторна робота № 11 .....	77
Лабораторна робота № 12 .....	89
Лабораторна робота № 13 .....	95
Лабораторна робота № 14 .....	106

					ДУ «Житомирська політехніка».24.121.15.000–Звіт						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Леус В.О.			Звіт з лабораторної роботи			Лім.	Арк.	Аркушів	
Перевір.		Піонтківський В.І								1	12
Керівник								ФІКТ Гр. ІПЗ-22-3			
Н. контр.											
Зав. каф.											

## Лабораторна робота № 1

**Тема:** Знайомство з мовою програмування Java. Написання простих програм на мові програмування Java

**Мета роботи:** Вивчити реалізацію базових алгоритмічних конструкцій у мові програмування Java; знайомство з правилами оформлення програмного коду.

### Хід роботи

#### Написання простих програм:

##### Програма 1 :

Ім'я класу: com.education.ztu.Task1

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

##### Task\_01.java:

```
package com.education.ztu.TASK_01;

public class Task_01 {
    public static void main(String[] args) {
        System.out.println("Hello, World!!!");
    }
}
```

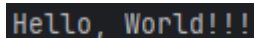


Рис.1.1. Результат роботи

##### Програма 2 :

Ім'я класу: com.education.ztu.Task2

Напишіть клас, який реалізує функціональність додавання двох цілих чисел.

Для зчитування даних використовувати методи класу Scanner.

##### Task\_02.java:

```
package com.education.ztu.TASK_02;

import java.util.Scanner;

public class Task_02 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first number: ");
        int a = scanner.nextInt();
        System.out.print("Enter the second number: ");
        int b = scanner.nextInt();
        int sum = a + b;
        System.out.println("Suma: " + sum);
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

```
Enter the first number: 2
Enter the second number: 2
Suma: 4
```

**Рис.1.2. Результат роботи**

### 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.

#### Main.java:

```
package com.education.ztu.TASK_03;

public class Main {
    public static void main(String[] args) {
        Task_03.Main(new String[]{"2", "3", "5", "8"});
    }
}
```

#### Task\_03.java:

```
package com.education.ztu.TASK_03;

public class Task_03 {
    public static void Main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            System.out.print(args[i]);
            if (i < args.length - 1) {
                System.out.print(" ");
            }
        }
    }
}
```

```
2 3 5 8
```

**Рис.1.3. Результат роботи**

### Програма 4 :

Ім'я класу: com.education.ztu.Task4

Напишіть клас, який реалізує функціональні можливості визначення найбільшого спільного дільника двох цілих додатних чисел.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		3

Для зчитування даних використовувати методи класу Scanner.

#### Task\_04.java:

```
package com.education.ztu.TASK_04;

import java.util.Scanner;
public class Task_04 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first number: ");
        int a = scanner.nextInt();
        System.out.print("Enter the second number: ");
        int b = scanner.nextInt();

        int gcd = findGCD(a, b);
        System.out.println("The greatest common divisor: " + gcd);
    }
    public static int findGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }
}
```

```
Enter the first number: 23
Enter the second number: 46
The greatest common divisor: 23
```

Рис.1.4. Результат роботи

#### Програма 5 :

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.

#### Task\_05.java:

```
package com.education.ztu.TASK_05;

import java.util.Scanner;
public class Task_05 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();

        int sum = sumDigits(number);
        System.out.println("The sum of the numbers: " + sum);
    }
    public static int sumDigits(int number) {
        int sum = 0;
        while (number != 0) {
            sum += number % 10;
            number /= 10;
        }
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    return sum;
}
}

```

```

Enter a number: 222
The sum of the numbers: 6

```

**Рис.1.5. Результат роботи**

### 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.

#### Task\_06.java:

```

package com.education.ztu.TASK_06;

import java.util.Scanner;

public class Task_06 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of items: ");
        int n = scanner.nextInt();
        int[] fibonacci = new int[n];
        fibonacci[0] = 1;
        fibonacci[1] = 1;
        for (int i = 2; i < n; i++) {
            fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];
        }
        System.out.print("A straight array: ");
        for (int i = 0; i < n; i++) {
            System.out.print(fibonacci[i] + " ");
        }
        System.out.print("\nReverse array: ");
        for (int i = n - 1; i >= 0; i--) {
            System.out.print(fibonacci[i] + " ");
        }
    }
}

```

```

Enter the number of items: 56
A straight array: 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524558 5702827 9227465 14930352 24146809 39078726 63245581 102557117 167761390 271416066 439126023 710683363 1149754886 1860438809 3010293175 4870847984 7881141159 12752037143 20633178127 33385215270 54018252413 87403467583 141421875000 228825137513 370246602513 601068377513 971314500000 1572382877513 2543697377513 4116080155026 6658463032539 10774550407565 17432913432604 28207463840169 45640377272773 73847841112942 119488218385615 193328595658558 312816813931500 506145412317113 818974008248671 1325119421165784 2144093533482897 3469212654648671 5614332175814455 9083545709301348 14697877884949023 23781423606550371 38479299321499419 62260725028049790 100742146649549209 163002871677609000 263745018727158719 426747895376657919 690490814094257129 1117238064821855048 1807629881216112967 2924867896038008016 4732496957254123065 7657364848470131084 12390061745728154100 20047426594198285165 32437488342668439265 52484914937866624429 84922393530594863694 137409808473461483963 222332223410748345662 360002111884230829656 582331935294979175320 942664158705717521014 1524996094000000000000 2467660252705717521014 3992656346705717521014 6460316601705717521014 10452972948411435042028 17013289290117152563042 27466262238528587575070 44479551528645739638112 72942833818762902203152 117422385347391491838172 190365237266054393951284 307788023613446396189456 500153260879500788140740 808518498045557184322024 1308671758925057972511764 2117190019804557972511764 3425758278729515944023528 5542948298534073916535292 8968706577258589858057056 14511654875792663774592348 23479561453051253632649404 37991266330343843407241752 61460818206136507181834096 100452484536479350814475848 164913302742615858000819944 265365787279095158815295792 430279090021714458815295792 705642397290810616815295792 1135921487312525775630591584 1841563884593336391445887376 2977485371905852167076478960 4819049259218368000000000000 7796534631123200000000000000 12615583903028000000000000000 20412118534151200000000000000 33027702437179200000000000000 53442291371331200000000000000 86460014305504000000000000000 139902205676832000000000000000 226362219982336000000000000000 366264425658176000000000000000 592626645640000000000000000000 958889071316000000000000000000 1551155716960000000000000000000 2509982388272000000000000000000 4061138105232000000000000000000 6571120493504000000000000000000 10632258600960000000000000000000 17203379094464000000000000000000 27835637698000000000000000000000 45039016792464000000000000000000 72874695886928000000000000000000 117913712581392000000000000000000 190788408468320000000000000000000 308702121049728000000000000000000 500490529518048000000000000000000 809178650566368000000000000000000 1318669180084416000000000000000000 2127847830650784000000000000000000 3446516911217152000000000000000000 5574364741867936000000000000000000 9020881653085120000000000000000000 14595246394952960000000000000000000 23616128048038400000000000000000000 38211374442000000000000000000000000 61827502540940800000000000000000000 100038877002944000000000000000000000 161856379543888000000000000000000000 261895256546832000000000000000000000 423754036090720000000000000000000000 685609315634624000000000000000000000 1109363591275392000000000000000000000 1795072906910080000000000000000000000 2904436498185472000000000000000000000 4699509395095552000000000000000000000 7603982293205888000000000000000000000 12303491688301440000000000000000000000 19907481081507200000000000000000000000 32210972769808640000000000000000000000 52118453851315840000000000000000000000 84328425833124480000000000000000000000 136446879684439040000000000000000000000 220775305517563520000000000000000000000 357222185199992960000000000000000000000 577997490717556480000000000000000000000 935272786437550080000000000000000000000 1513270277155107200000000000000000000000 2448543063692659200000000000000000000000 3961813340847769600000000000000000000000 6410356404540428800000000000000000000000 10372169745388188800000000000000000000000 16782526150028614400000000000000000000000 27154695895416803200000000000000000000000 43937222045445440000000000000000000000000 71091817895864320000000000000000000000000 115029040041280000000000000000000000000000 186047857937158400000000000000000000000000 301076897978444800000000000000000000000000 487125945915603200000000000000000000000000 788202843833728000000000000000000000000000 1275328791749337600000000000000000000000000 2063454683498675200000000000000000000000000 3338579375248000000000000000000000000000000 5401704066997344000000000000000000000000000 8741958752245760000000000000000000000000000 14143662819243520000000000000000000000000000 22885366971487360000000000000000000000000000 37027071023731200000000000000000000000000000 60108775075975040000000000000000000000000000 97130479128218880000000000000000000000000000 157248004204193280000000000000000000000000000 254365529280167680000000000000000000000000000 411603054360335360000000000000000000000000000 665840579440503040000000000000000000000000000 1077455829600768000000000000000000000000000000 1743291079761024000000000000000000000000000000 2820746329921280000000000000000000000000000000 4564037579382400000000000000000000000000000000 7384784079040000000000000000000000000000000000 11948821577600000000000000000000000000000000000 19332859075200000000000000000000000000000000000 31281681572800000000000000000000000000000000000 51422904070400000000000000000000000000000000000 83204127068000000000000000000000000000000000000 134626347046400000000000000000000000000000000000 217838567024000000000000000000000000000000000000 352450787001600000000000000000000000000000000000 570262996979200000000000000000000000000000000000 922681594009600000000000000000000000000000000000 1505344188019200000000000000000000000000000000000 2438006376038400000000000000000000000000000000000 3961350564057600000000000000000000000000000000000 6400356938112000000000000000000000000000000000000 10361713876224000000000000000000000000000000000000 16722070814336000000000000000000000000000000000000 27083784688672000000000000000000000000000000000000 43805855503008000000000000000000000000000000000000 70887830317344000000000000000000000000000000000000 114693610634688000000000000000000000000000000000000 185581421269376000000000000000000000000000000000000 300275031904064000000000000000000000000000000000000 485856453173440000000000000000000000000000000000000 786131484346880000000000000000000000000000000000000 1271987937520320000000000000000000000000000000000000 2058124875040640000000000000000000000000000000000000 3329102812560960000000000000000000000000000000000000 5387227688102400000000000000000000000000000000000000 8716330500663040000000000000000000000000000000000000 14103558181224960000000000000000000000000000000000000 22819886361886720000000000000000000000000000000000000 37043444543111680000000000000000000000000000000000000 60863330886223360000000000000000000000000000000000000 98886771429335040000000000000000000000000000000000000 160750102315558400000000000000000000000000000000000000 261536814634892800000000000000000000000000000000000000 422286916954227200000000000000000000000000000000000000 683823731588454400000000000000000000000000000000000000 1106109863176908800000000000000000000000000000000000000 1792219726353817600000000000000000000000000000000000000 2900329589530726400000000000000000000000000000000000000 4692549315884544000000000000000000000000000000000000000 7692868805415296000000000000000000000000000000000000000 12585418121230720000000000000000000000000000000000000000 20277967242461440000000000000000000000000000000000000000 32860385384722880000000000000000000000000000000000000000 53138352627184320000000000000000000000000000000000000000 86006319869645760000000000000000000000000000000000000000 139144672539330000000000000000000000000000000000000000000 225151025078660000000000000000000000000000000000000000000 364295697617990000000000000000000000000000000000000000000 593440370157320000000000000000000000000000000000000000000 957635042696650000000000000000000000000000000000000000000 1555270115243950000000000000000000000000000000000000000000 2542905187791250000000000000000000000000000000000000000000 4100540260338550000000000000000000000000000000000000000000 6643445446081900000000000000000000000000000000000000000000 10746397302163800000000000000000000000000000000000000000000 17389842508327600000000000000000000000000000000000000000000 28532787714491500000000000000000000000000000000000000000000 46012629920655400000000000000000000000000000000000000000000 74542472126819200000000000000000000000000000000000000000000 120554901047438400000000000000000000000000000000000000000000 195097329968057600000000000000000000000000000000000000000000 315649758888672000000000000000000000000000000000000000000000 510747187789280000000000000000000000000000000000000000000000 826394616689888000000000000000000000000000000000000000000000 1337141794479744000000000000000000000000000000000000000000000 2163536411169600000000000000000000000000000000000000000000000 3500678102339200000000000000000000000000000000000000000000000 5664214513508800000000000000000000000000000000000000000000000 9164892624678400000000000000000000000000000000000000000000000 14829107138286720000000000000000000000000000000000000000000000 23993321651895040000000000000000000000000000000000000000000000 38812443203191040000000000000000000000000000000000000000000000 62705764806382080000000000000000000000000000000000000000000000 101518086409573120000000000000000000000000000000000000000000000 164223172819146240000000000000000000000000000000000000000000000 265741259228392480000000000000000000000000000000000000000000000 430064432047538720000000000000000000000000000000000000000000000 705387604866684960000000000000000000000000000000000000000000000 1140710777685831040000000000000000000000000000000000000000000000 1846033950504972160000000000000000000000000000000000000000000000 3001357123324113280000000000000000000000000000000000000000000000 4847680296143254400000000000000000000000000000000000000000000000 7849003468962395520000000000000000000000000000000000000000000000 12696686641781536640000000000000000000000000000000000000000000000 20544369814600677760000000000000000000000000000000000000000000000 33432052987419818880000000000000000000000000000000000000000000000 53969736160239060000000000000000000000000000000000000000000000000 87438422320478120000000000000000000000000000000000000000000000000 141808158480956240000000000000000000000000000000000000000000000000 231646316961912480000000000000000000000000000000000000000000000000 373454475442868720000000000000000000000000000000000000000000000000 605300641885737440000000000000000000000000000000000000000000000000 988754917328606080000000000000000000000000000000000000000000000000 1614059834657212160000000000000000000000000000000000000000000000000 2602814751985824320000000000000000000000000000000000000000000000000 4216874583971648640000000000000000000000000000000000000000000000000 6820889335957472960000000000000000000000000000000000000000000000000 11041768671914945920000000000000000000000000000000000000000000000000 17862648007872418880000000000000000000000000000000000000000000000000 29104316679744837760000000000000000000000000000000000000000000000000 46966964687617256960000000000000000000000000000000000000000000000000 75131681367334513920000000000000000000000000000000000000000000000000 12109864503
```

Створити масив символів латинського алфавіту та вести їх числові коди в такому форматі: A ==> 65 B ==> 66 C ==> 67

#### Task\_07.java:

```
package com.education.ztu.TASK_07;

public class Task_07 {
    public static void main(String[] args) {
        char[] alphabet = new char[26];
        for (int i = 0; i < 26; i++) {
            alphabet[i] = (char) ('A' + i);
            System.out.println(alphabet[i] + " ==> " + (int) alphabet[i]);
        }
    }
}
```

```
R ==> 82
S ==> 83
T ==> 84
U ==> 85
V ==> 86
W ==> 87
X ==> 88
Y ==> 89
Z ==> 90
```

**Рис.1.7. Результат роботи**

**Висновок:** я вивчив реалізацію базових алгоритмічних конструкцій у мові програмування Java та знайомився з правилами оформлення програмного коду.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

## Лабораторна робота № 2

**Тема:** Створення структури класу заданої предметної області.

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

### Хід роботи

**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 оператору `instanceof`.

**Task\_07.java:**

**Task\_07.java:**

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Task\_07.java:  
Task\_07.java:  
Task\_07.java:  
Task\_07.java:  
Task\_07.java:

```
John Doe
VINNYTSYA
20
I am a teacher.
I am an employee.
I am a student.
Is teacher's car engine running? false
Engine started.
Is teacher's car engine running? true
Created teachers: 1
Created students: 1
Created employees: 1
Teacher is a Person.
```

**Рис.2.1. Результат роботи**

**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 enum Location.

**Location.java:**

```
package com.education.ztu.TASK_03;

enum Location {
    USA,
    CANADA,
    UKRAINE,
    GERMANY,
    FRANCE;
}
```

**Main.java:**

```
package com.education.ztu.TASK_03;

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

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				8
Змн.	Арк.	№ докум.	Підпис	Дата		



```

System.out.println("Addition: " + Operation.addition(1, 2, 3, 4));
System.out.println("Subtraction: " + Operation.subtraction(10, 3, 2));
System.out.println("Multiplication: " + Operation.multiplication(2, 3,
4));

System.out.println("Division: " + Operation.division(100.0, 5.0, 2.0));
System.out.println("Average: " + Operation.average(5, 10, 15));
System.out.println("Maximum: " + Operation.maximum(1, 2, 10, 3));
System.out.println("Minimum: " + Operation.minimum(1, 2, 10, 3));

System.out.println("\nAll locations:");
for (Location loc : Location.values()) {
    System.out.println(loc);
}
}
}

```

### Operation.java:

```

package com.education.ztu.TASK_03;

public class Operation {
    public static int addition(int... numbers) {
        int sum = 0;
        for (int num : numbers) {
            sum += num;
        }
        return sum;
    }
    public static int subtraction(int... numbers) {
        int result = numbers[0];
        for (int i = 1; i < numbers.length; i++) {
            result -= numbers[i];
        }
        return result;
    }
    public static int multiplication(int... numbers) {
        int result = 1;
        for (int num : numbers) {
            result *= num;
        }
        return result;
    }
    public static double division(double... numbers) {
        double result = numbers[0];
        for (int i = 1; i < numbers.length; i++) {
            result /= numbers[i];
        }
        return result;
    }
    public static double average(int... numbers) {
        int sum = addition(numbers);
        return (double) sum / numbers.length;
    }
    public static int maximum(int... numbers) {
        int max = numbers[0];
        for (int num : numbers) {
            if (num > max) {
                max = num;
            }
        }
        return max;
    }
    public static int minimum(int... numbers) {

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```
int min = numbers[0];
for (int num : numbers) {
    if (num < min) {
        min = num;
    }
}
return min;
}
```

```
Addition: 10
Subtraction: 5
Multiplication: 24
Division: 10.0
Average: 10.0
Maximum: 10
Minimum: 1

All locations:
USA
CANADA
UKRAINE
GERMANY
FRANCE
```

Рис.2.2. Результат роботи

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

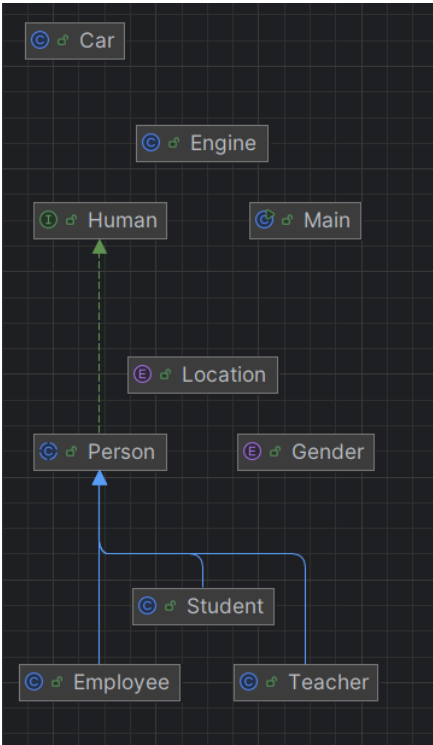


Рис.2.3. UML-діаграма до завдання 2



**Рис.2.4. UML-діаграма до завдання 3**

**Висновок:** я створити ієрархію класів заданої предметної області, попрацював з статичними методами.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				11
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.

#### Хід роботи

**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.

**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.

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

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

- Продемонструвати роботу порівнянь на прикладі сортування учасників команд.

#### AgeComparator.java:

```
package com.education.ztu.game;

import java.util.Comparator;

public class AgeComparator implements Comparator<Participant> {
    @Override
    public int compare(Participant p1, Participant p2) {
        return Integer.compare(p1.getAge(), p2.getAge());
    }
}
```

#### Employee.java:

```
package com.education.ztu.game;

public class Employee extends Participant{
    public Employee(String name, int age) {
        super(name, age);
    }
}
```

#### Game.java:

```
package com.education.ztu.game;

public class Game {
    public static void main(String[] args) {
        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);

        Team<Schoolar> schoolarTeam = new Team<>("Dragon");
        schoolarTeam.addNewParticipant(schoolar1);
        schoolarTeam.addNewParticipant(schoolar2);

        Team<Student> studentTeam = new Team<>("Vpered");
        studentTeam.addNewParticipant(student1);
        studentTeam.addNewParticipant(student2);

        Team<Employee> employeeTeam = new Team<>("Robotyagi");
        employeeTeam.addNewParticipant(employee1);
        employeeTeam.addNewParticipant(employee2);

        schoolarTeam.playWith(new Team<>("Rozumnyky"));
        studentTeam.playWith(new Team<>("StudentPower"));
    }
}
```

#### Participant.java:

```
package com.education.ztu.game;

public abstract class Participant implements Cloneable, Comparable<Participant> {
    private String name;
    private int age;
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public Participant(String name, int age) {
    this.name = name;
    this.age = age;
}

public String getName() {
    return name;
}

public int getAge() {
    return age;
}

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

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

@Override
public Participant clone() {
    try {
        return (Participant) super.clone();
    } catch (CloneNotSupportedException e) {
        throw new RuntimeException("Clone not supported for Participant", e);
    }
}

@Override
public boolean equals(Object obj) {
    if (this == obj) return true;
    if (obj == null || getClass() != obj.getClass()) return false;
    Participant that = (Participant) obj;
    return age == that.age && name.equals(that.name);
}

@Override
public int hashCode() {
    return 31 * name.hashCode() + age;
}

@Override
public String toString() {
    return "Participant{" + "name='" + name + '\'' + ", age=" + age + '\'';
}

@Override
public int compareTo(Participant other) {
    return this.name.compareTo(other.name);
}
}

```

### Schoolar.java:

```

package com.education.ztu.game;

public class Schoolar extends Participant{
    public Schoolar(String name, int age) {
        super(name, age);
    }
}

```

		Левус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
}

```

## Student.java:

```

package com.education.ztu.game;

public class Student extends Participant{
    public Student(String name, int age) {
        super(name, age);
    }
}

```

## Team.java:

```

package com.education.ztu.game;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Team<T extends Participant> implements Cloneable {
    private String name;
    private List<T> participants = new ArrayList<>();

    public Team(String name) {
        this.name = name;
    }

    public void addNewParticipant(T participant) {
        participants.add(participant);
        System.out.println("To the team " + name + " was added participant " +
participant.getName());
    }

    public void playWith(Team<T> team) {
        String winnerName;
        Random random = new Random();
        int i = random.nextInt(2);
        if (i == 0) {
            winnerName = this.name;
        } else {
            winnerName = team.name;
        }
        System.out.println("The team " + winnerName + " is winner!");
    }

    @Override
    public Team<T> clone() {
        try {
            Team<T> clonedTeam = (Team<T>) super.clone();
            clonedTeam.participants = new ArrayList<>();
            for (T participant : this.participants) {
                clonedTeam.participants.add((T) participant.clone());
            }
            return clonedTeam;
        } catch (CloneNotSupportedException e) {
            throw new RuntimeException("Clone not supported for Team", e);
        }
    }

    @Override
    public String toString() {
        return "Team{" + "name='" + name + '\'' + ", participants=" + participants

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

```
+ '}'';
}
}
```

## Main.java:

```
package com.education.ztu;

import com.education.ztu.game.*;

public class Main {
    public static void main(String[] args) {
        Scholar scholar1 = new Scholar("Ivan", 13);
        Scholar scholar2 = new Scholar("Mariya", 15);

        Team<Scholar> scholarTeam = new Team<>("Dragon");
        scholarTeam.addNewParticipant(scholar1);
        scholarTeam.addNewParticipant(scholar2);

        System.out.println("Original team: " + scholarTeam);

        Team<Scholar> clonedTeam = scholarTeam.clone();
        System.out.println("Cloned team: " + clonedTeam);

        System.out.println("Equals: " + scholar1.equals(scholar2));
        System.out.println("HashCode of scholar1: " + scholar1.hashCode());
        System.out.println("HashCode of scholar2: " + scholar2.hashCode());
    }
}
```

## Main2.java:

```
package com.education.ztu;

import com.education.ztu.game.*;

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

public class Main2 {
    public static void main(String[] args) {
        Scholar scholar1 = new Scholar("Ivan", 13);
        Scholar scholar2 = new Scholar("Mariya", 15);
        Scholar scholar3 = new Scholar("Sergey", 12);

        List<Scholar> scholars = new ArrayList<>();
        scholars.add(scholar1);
        scholars.add(scholar2);
        scholars.add(scholar3);

        Collections.sort(scholars);
        System.out.println("Sorted by name: " + scholars);

        Collections.sort(scholars, new AgeComparator());
        System.out.println("Sorted by age: " + scholars);
    }
}
```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонківський В.І.				16
Змн.	Арк.	№ докум.	Підпис	Дата		



```

To the team Dragon was added participant Ivan
To the team Dragon was added participant Mariya
Original team: Team{name='Dragon', participants=[Participant{name='Ivan', age=13}, Participant{name='Mariya', age=15}]}
Cloned team: Team{name='Dragon', participants=[Participant{name='Ivan', age=13}, Participant{name='Mariya', age=15}]}
Equals: false
HashCode of scholar1: 71029011
HashCode of scholar2: -1791104196

```

### Рис.3.1. Результат роботи Main

```

Sorted by name: [Participant{name='Ivan', age=13}, Participant{name='Mariya', age=15}, Participant{name='Sergey', age=12}]
Sorted by age: [Participant{name='Sergey', age=12}, Participant{name='Ivan', age=13}, Participant{name='Mariya', age=15}]

```

### Рис.3.2. Результат роботи Main2

**Висновок:** я створив міні проект Game з використанням узагальнень, клонування та порівняння об'єктів.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

## 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.

### Хід роботи

**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.

#### StringPractice.java:

```
package Task_02;

public class StringPractice {

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

        char lastChar = str.charAt(str.length() - 1);
        System.out.println("The last character of the string: " + lastChar);

        boolean endsWithExclamation = str.endsWith("!!!");
        System.out.println("Does the string end in \"!!!\": " + endsWithExclamation);

        boolean startsWithILearn = str.startsWith("I learn ");
        System.out.println("Does the string begin with \"I learn \": " + startsWithILearn);

        boolean containsJava = str.contains("Java");
        System.out.println("Does the string contain a substring \"Java\": " + containsJava);

        int indexOfJava = str.indexOf("Java");
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```

System.out.println("Substring position \"Java\": " + indexOfJava);

String replacedString = str.replace('a', 'o');
System.out.println("Replaced by. \"a\" на \"o\": " + replacedString);

String upperCaseString = str.toUpperCase();
System.out.println("The string is in upper case: " + upperCaseString);

String lowerCaseString = str.toLowerCase();
System.out.println("The string is in lower case: " + lowerCaseString);

String cutJava = str.substring(0, indexOfJava) + str.substring(indexOfJava
+ 4);
System.out.println("A string without \"Java\": " + cutJava);
}
}

```

```

The last character of the string: !
Does the string end in "!!!": true
Does the string begin with "I learn ": true
Does the string contain a substring "Java": true
Substring position "Java": 8
Replaced by. "a" на "o": I leorn Jovo!!!
The string is in upper case: I LEARN JAVA!!!
The string is in lower case: i learn java!!!
A string without "Java": I learn !!!

```

**Рис.4.1. Результат роботи**

**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()`.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

- Визначити довжину та capacity.

### StringBuilderPractice.java:

```
package Task_03;

public class StringBuilderPractice {

    public static void main(String[] args) {
        int num1 = 4;
        int num2 = 36;

        StringBuilder sb1 = new StringBuilder();
        sb1.append(num1).append(" + ").append(num2).append(" = ").append(num1 +
num2);
        System.out.println(sb1.toString());

        StringBuilder sb2 = new StringBuilder();
        sb2.append(num1).append(" - ").append(num2).append(" = ").append(num1 -
num2);
        System.out.println(sb2.toString());

        StringBuilder sb3 = new StringBuilder();
        sb3.append(num1).append(" * ").append(num2).append(" = ").append(num1 *
num2);
        System.out.println(sb3.toString());

        StringBuilder sb1Modified = new StringBuilder(sb1);
        int equalSignIndex1 = sb1Modified.indexOf("=");
        sb1Modified.deleteCharAt(equalSignIndex1);
        sb1Modified.insert(equalSignIndex1, "evenly");
        System.out.println("After insert and deleteCharAt: " +
sb1Modified.toString());

        StringBuilder sb2Modified = new StringBuilder(sb2);
        int equalSignIndex2 = sb2Modified.indexOf("=");
        sb2Modified.replace(equalSignIndex2, equalSignIndex2 + 1, "even");
        System.out.println("After replace: " + sb2Modified.toString());

        StringBuilder sb3Reversed = new StringBuilder(sb3);
        sb3Reversed.reverse();
        System.out.println("After reverse: " + sb3Reversed.toString());

        System.out.println("Length sb1: " + sb1.length());
        System.out.println("Capacity sb1: " + sb1.capacity());
    }
}
```

```
4 + 36 = 40
4 - 36 = -32
4 * 36 = 144
After insert and deleteCharAt: 4 + 36 evenly 40
After replace: 4 - 36 even -32
After reverse: 441 = 63 * 4
Length sb1: 11
Capacity sb1: 16
```

Рис.4.2. Результат роботи

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

**Завдання 4.** Вивести у форматованому 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.

**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).
- Об'єднати їх у `Resource Bundle`
- Реалізувати функціонал отримання та роботи з даними для кожної локалі.

#### FormatterTask.java:

```
package Task_04_05.logic;

import java.util.Formatter;
import java.util.List;

public class FormatterTask {
    public static void printReceipt(List<Product> products) {
        Formatter formatter = new Formatter();
        formatter.format("Дата та час покупки: %s\n", "28.03.2019 13:25:12");
        formatter.format("===== \n");
        formatter.format("№ %-15s %-15s %s\n", "Товар", "Категорія", "Ціна");
        formatter.format("===== \n");
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        double total = 0;
        for (int i = 0; i < products.size(); i++) {
            Product product = products.get(i);
            formatter.format("%d. %-15s %-15s %.2f @%n",
                i + 1, product.getName(), product.getCategory(), prod-
uct.getPrice());
            total += product.getPrice();
        }

        formatter.format("===== %n");
        formatter.format("Разом: %.2f @%n", total);

        System.out.println(formatter);
        formatter.close();
    }
}

```

### LocalizationTask.java:

```

package Task_04_05.logic;

import java.text.NumberFormat;
import java.util.List;
import java.util.Locale;
import java.util.ResourceBundle;

public class LocalizationTask {
    public static void printLocalizedReceipt(List<Product> products, Locale lo-
cale) {
        ResourceBundle bundle = Resource-
Bundle.getBundle("Task_04_05.resources.data", locale);
        NumberFormat currencyFormatter = NumberFormat.getCurrencyInstance(locale);

        System.out.println(bundle.getString("dateTime"));
        System.out.println("=====");
        System.out.printf("%-3s %-15s %-15s %s%n",
            bundle.getString("no"), bundle.getString("product"), bun-
dle.getString("category"), bundle.getString("price"));
        System.out.println("=====");

        double total = 0;
        for (int i = 0; i < products.size(); i++) {
            Product product = products.get(i);
            System.out.printf("%-3d %-15s %-15s %s%n",
                i + 1, product.getName(), product.getCategory(), currencyFor-
matter.format(product.getPrice()));
            total += product.getPrice();
        }

        System.out.println("=====");
        System.out.printf("%s: %s%n", bundle.getString("total"), currencyForma-
ter.format(total));
    }
}

```

### Product.java:

```

package Task_04_05.logic;

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

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

    public String getName() {
        return name;
    }

    public String getCategory() {
        return category;
    }

    public double getPrice() {
        return price;
    }
}

```

### data\_en\_US.properties:

```

dateTime=Date and time of purchase: 28.03.2019 13:25:12
no=No
product=Product
category=Category
price=Price
total=Total

```

### data\_fr\_FR.properties:

```

dateTime=Date et heure d'achat : 28.03.2019 13:25:12
no=N°
product=Produit
category=Catégorie
price=Prix
total=Total

```

### Main.java:

```

package Task_04_05;

import Task_04_05.logic.FormatterTask;
import Task_04_05.logic.LocalizationTask;
import Task_04_05.logic.Product;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class Main {
    public static void main(String[] args) {
        List<Product> products = new ArrayList<>();
        products.add(new Product("Джинси", "Жіночий одяг", 1500.78));
        products.add(new Product("Спідниця", "Жіночий одяг", 1000.56));
        products.add(new Product("Краватка", "Чоловічий одяг", 500.78));

        // Adding more products
        for (int i = 0; i < 7; i++) {
            products.add(new Product("Товар" + (i + 4), "Категорія" + (i + 4),
200.00 + i * 50));
        }

        // Task 4: Formatted check
        FormatterTask.printReceipt(products);
    }
}

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

```
// Task 5: Localised check
System.out.println("\n=== Локалізований чек (English) ===");
LocalizationTask.printLocalizedReceipt(products, Locale.US);

System.out.println("\n=== Локалізований чек (Français) ===");
LocalizationTask.printLocalizedReceipt(products, Locale.FRANCE);
}
```

```
=== Локалізований чек (English) ===
Date and time of purchase: 28.03.2019 13:25:12
=====
No  Product          Category          Price
=====
1   Джинси           Жіночий одяг     $1,500.78
2   Спідниця          Жіночий одяг     $1,000.56
3   Краватка          Чоловічий одяг   $500.78
4   Товар4            Категорія4       $200.00
5   Товар5            Категорія5       $250.00
6   Товар6            Категорія6       $300.00
7   Товар7            Категорія7       $350.00
8   Товар8            Категорія8       $400.00
9   Товар9            Категорія9       $450.00
10  Товар10           Категорія10      $500.00
=====
Total: $5,452.12

=== Локалізований чек (Français) ===
Date et heure d'achat : 28.03.2019 13:25:12
=====
N°  Produit          Catégorie          Prix
=====
```

Рис.4.3. Результат роботи

**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.

### DateTask.java:

```
package Task_06;

import java.time.LocalDateTime;
import java.time.Year;
import java.time.format.DateTimeFormatter;

public class DateTask {

    public static void main(String[] args) {
        LocalDateTime labStart = LocalDateTime.of(2024, 10, 12, 9, 0, 0);

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("EEEE, d 'day of the year', MMMM yyyy, HH:mm:ss");
        String formattedLabStart = labStart.format(formatter);

        System.out.println("Lab start: " + formattedLabStart);

        int year = labStart.getYear();
        boolean isLeapYear = Year.of(year).isLeap();
        System.out.println("Is the year " + year + " a leap year: " + isLeapYear);

        LocalDateTime now = LocalDateTime.now();
        System.out.println("Current time: " + now.format(formatter));

        if (now.isAfter(labStart)) {
            System.out.println("Current time is after the lab start.");
        } else if (now.isBefore(labStart)) {
            System.out.println("Current time is before the lab start.");
        } else {
            System.out.println("Current time matches the lab start time.");
        }

        LocalDateTime updatedLabStart = labStart.plusDays(5).plusHours(3).minusMinutes(10);
        System.out.println("Updated lab start time: " + updatedLabStart.format(formatter));
    }
}
```

```
Lab start: суббота, 12 day of the year, октябрь 2024, 09:00:00
Is the year 2024 a leap year: true
Current time: вторник, 10 day of the year, декабрь 2024, 11:53:32
Current time is after the lab start.
Updated lab start time: четверг, 17 day of the year, октябрь 2024, 11:50:00
```

### Рис.4.4. Результат роботи

**Висновок:** я попрацював з класами String, StringBuffer, StringBuilder та їх методами та попрактикувався з використанням локалізації та інтернаціоналізації, попрацював з датами.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

## Лабораторна робота № 5

**Тема:** Java Collections Framework.

**Мета роботи:** Робота з Java Collections Framework.

### Хід роботи

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

**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).

**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).

**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).

**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) .

**Завдання 7.** Продемонструвати роботу з класом Collections:

- Для роботи використати масив створений через Arrays.asList
- Метод Collections.sort()

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

- Метод Collections.binarySearch()
- Методы Collections.reverse(), Collections.shuffle()
- Метод Collections.fill()
- Методы Collections.max(), Collections.min() • Метод Collections.copy()
- Метод Collections.rotate()
- Метод Collections.checkedCollection()
- Метод Collections.frequency()

### Main.java:

```
package com.education.ztu;

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

public class Main {
    public static void main(String[] args) {
        ArrayList<Product> products = new ArrayList<>();
        products.add(new Product("Milk", 1.50));
        products.add(new Product("Bread", 0.90));
        products.add(new Product("Butter", 2.40));

        products.addAll(Collections.singletonList(new Product("Cheese", 3.10)));
        System.out.println("First product: " + products.get(0));
        System.out.println("Index of Bread: " + products.indexOf(new Product("Bread", 0.90)));
        System.out.println("Last index of Milk: " + products.lastIndexOf(new Product("Milk", 1.50)));
        System.out.println("Size: " + products.size());
    }
}
```

### MainCollections.java:

```
package com.education.ztu;

import java.util.*;

public class MainCollections {
    public static void main(String[] args) {
        List<Product> productList = Arrays.asList(
            new Product("Car", 20000.00),
            new Product("Bike", 500.00),
            new Product("Scooter", 1500.00)
        );

        Collections.sort(productList, Comparator.comparingDouble(Product::getPrice));
        System.out.println("Sorted by price: " + productList);

        Collections.sort(productList, Comparator.comparing(Product::getName));
        System.out.println("Sorted by name: " + productList);

        int index = Collections.binarySearch(productList, new Product("Bike", 500.00), Comparator.comparingDouble(Product::getPrice));
        System.out.println("Binary search for Bike by price: " + index);
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Collections.reverse(productList);
        System.out.println("Reversed list: " + productList);

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

        List<Product> fillList = new ArrayList<>(Arrays.asList(new Product[3]));
        Collections.fill(fillList, new Product("Default
com.education.ztu.Product", 100.0));
        System.out.println("Filled list: " + fillList);

        Product maxProduct = Collections.max(productList, Compar-
tor.comparingDouble(Product::getPrice));
        Product minProduct = Collections.min(productList, Compar-
tor.comparingDouble(Product::getPrice));
        System.out.println("Max price product: " + maxProduct);
        System.out.println("Min price product: " + minProduct);

        List<Product> copyList = new ArrayList<>(Arrays.asList(new Prod-
uct[productList.size()]));
        Collections.copy(copyList, productList);
        System.out.println("Copied list: " + copyList);

        Collections.rotate(productList, 2);
        System.out.println("Rotated list: " + productList);

        Collection<Product> checkedCollection = Collec-
tions.checkedCollection(productList, Product.class);
        System.out.println("Checked collection: " + checkedCollection);

        int frequency = Collections.frequency(productList, new Product("Bike",
500.00));
        System.out.println("Frequency of Bike: " + frequency);
    }
}

```

## MainMap.java:

```

package com.education.ztu;

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

public class MainMap {
    public static void main(String[] args) {
        HashMap<String, Product> map = new HashMap<>();
        map.put("TV", new Product("TV", 300.00));
        map.put("Laptop", new Product("Laptop", 800.00));

        System.out.println("Get TV: " + map.get("TV"));
        System.out.println("Contains Key 'Laptop': " + map.containsKey("Laptop"));
        System.out.println("Contains Value (Laptop): " + map.containsValue(new
Product("Laptop", 800.00)));

        for (Map.Entry<String, Product> entry : map.entrySet()) {
            System.out.println("Key: " + entry.getKey() + ", Value: " + en-
try.getValue());
        }
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

### MainQueue.java:

```
package com.education.ztu;

import java.util.ArrayDeque;

public class MainQueue {
    public static void main(String[] args) {
        ArrayDeque<Product> queue = new ArrayDeque<>();
        queue.push(new Product("Juice", 2.30));
        queue.offerLast(new Product("Chocolate", 1.70));

        System.out.println("First: " + queue.getFirst());
        System.out.println("Peek Last: " + queue.peekLast());
        queue.pop();
        System.out.println("After pop: " + queue);

        queue.removeLast();
        System.out.println("After removeLast: " + queue);
    }
}
```

### MainSet.java:

```
package com.education.ztu;

import java.util.TreeSet;

public class MainSet {
    public static void main(String[] args) {
        TreeSet<Product> set = new TreeSet<>((p1, p2) ->
p1.getName().compareTo(p2.getName()));
        set.add(new Product("Banana", 0.60));
        set.add(new Product("Apple", 0.40));
        set.add(new Product("Orange", 0.80));

        System.out.println("First: " + set.first());
        System.out.println("Last: " + set.last());
        System.out.println("HeadSet (Apple): " + set.headSet(new Product("Apple",
0.40)));
        System.out.println("SubSet (Apple to Banana): " + set.subSet(new Product("Apple", 0.40), new Product("Banana", 0.60)));
    }
}
```

### Product.java:

```
package com.education.ztu;

import java.util.Objects;

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

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

    public String getName() {
        return name;
    }
}
```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

    public double getPrice() {
        return price;
    }

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

    @Override
    public String toString() {
        return "com.education.ztu.Product{name='" + name + "', price=" + price +
        "}'";
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Product product = (Product) o;
        return Double.compare(product.price, price) == 0 &&
            Objects.equals(name, product.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, price);
    }
}

```

```

First product: com.education.ztu.Product{name='Milk', price=1.5}
Index of Bread: 1
Last index of Milk: 0
Size: 4

```

**Рис.5.1. Результат роботи Main**

```

Sorted by price: [com.education.ztu.Product{name='Bike', price=500.0}, com.education.ztu.Prod
Sorted by name: [com.education.ztu.Product{name='Bike', price=500.0}, com.education.ztu.Prod
Binary search for Bike by price: 0
Reversed list: [com.education.ztu.Product{name='Scooter', price=1500.0}, com.education.ztu.Pr
Shuffled list: [com.education.ztu.Product{name='Scooter', price=1500.0}, com.education.ztu.Pr
Filled list: [com.education.ztu.Product{name='Default com.education.ztu.Product', price=100.0
Max price product: com.education.ztu.Product{name='Car', price=20000.0}
Min price product: com.education.ztu.Product{name='Bike', price=500.0}
Copied list: [com.education.ztu.Product{name='Scooter', price=1500.0}, com.education.ztu.Prod
Rotated list: [com.education.ztu.Product{name='Bike', price=500.0}, com.education.ztu.Product
Checked collection: [com.education.ztu.Product{name='Bike', price=500.0}, com.education.ztu.P
Frequency of Bike: 1

```

**Рис.5.2. Результат роботи MainCollections**

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Get TV: com.education.ztu.Product{name='TV', price=300.0}
Contains Key 'Laptop': true
Contains Value (Laptop): true
Key: Laptop, Value: com.education.ztu.Product{name='Laptop', price=800.0}
Key: TV, Value: com.education.ztu.Product{name='TV', price=300.0}

```

**Рис.5.3. Результат роботи MainMap**

```

First: com.education.ztu.Product{name='Juice', price=2.3}
Peek Last: com.education.ztu.Product{name='Chocolate', price=1.7}
After pop: [com.education.ztu.Product{name='Chocolate', price=1.7}]
After removeLast: []

```

**Рис.5.4. Результат роботи MainQueue**

```

First: com.education.ztu.Product{name='Apple', price=0.4}
Last: com.education.ztu.Product{name='Orange', price=0.8}
HeadSet (Apple): []
SubSet (Apple to Banana): [com.education.ztu.Product{name='Apple', price=0.4}]

```

**Рис.5.5. Результат роботи MainSet**

**Висновок:** я попрацював з Java Collections Framework.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

## Лабораторна робота № 6

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

**Мета роботи:** Обробка виключних ситуація, створення власних класів винятків, робота з потоками вводу-виводу.

### Хід роботи

**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.

#### LoginValidator.java:

```
package Task_02;
public class LoginValidator {
    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 must contain only Latin letters, digits, and underscores, and be less than 20 characters long.");
            }

            if (!password.matches("[a-zA-Z0-9_]+$") || password.length() >= 20) {
                throw new WrongPasswordException("Password must contain only Latin letters, digits, and underscores, and be less than 20 characters long.");
            }

            if (!password.equals(confirmPassword)) {
                throw new WrongPasswordException("Password and confirmPassword do not match.");
            }
        }
        return true;
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				32
Змн.	Арк.	№ докум.	Підпис	Дата		



```

not match.");
    }

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

```

### Main.java:

```

package Task_02;

public class Main {
    public static void main(String[] args) {
        boolean isValid = LoginValidator.checkCredentials("123", "password_123",
"password_123");
        System.out.println("Credentials valid: " + isValid);
    }
}

```

### WrongLoginException.java:

```

package Task_02;

public class WrongLoginException extends Exception {
    public WrongLoginException() {
        super("Invalid login format");
    }

    public WrongLoginException(String message) {
        super(message);
    }
}

```

### WrongPasswordException.java:

```

package Task_02;

public class WrongPasswordException extends Exception {
    public WrongPasswordException() {
        super("Invalid password format");
    }

    public WrongPasswordException(String message) {
        super(message);
    }
}

```

Credentials valid: true

**Рис.6.1. Результат роботи**

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

- Перевикористати код для формування звіту з покупок з лабораторної роботи

4.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

- Після покупки, записати звіт у файл, який містить інформацію про вміст кошика.
- Використовуємо клас `FileWriter` або `PrintWriter` для запису звіту.
- Використовуємо `FileReader` для читання звіту та відображення в консолі.
- Не використовувати `try-with-resources`.

### Receipt.java:

```
package Task_03;

import java.io.FileWriter;
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.IOException;
import java.util.Formatter;
import java.util.Locale;

public class Receipt {

    public static void main(String[] args) {
        String filePath = "Java-LAB-06/src/directory_for_files/receipt.txt";

        try {
            writeReceiptToFile(filePath);
            readReceiptFromFile(filePath);
        } catch (IOException e) {
            System.out.println("Error while working with the file: " +
e.getMessage());
        }
    }

    private static void writeReceiptToFile(String filePath) throws IOException {
        Formatter formatter = new Formatter(Locale.UK);
        formatter.format("Дата та час покупки: %s\n", "26.10.2024 13:25:12");
        formatter.format("=====\n");
        formatter.format("%-3s %-15s %-15s %10s\n", "№", "Товар", "Категорія",
"Ціна");
        formatter.format("=====\n");

        formatter.format("%-3d %-15s %-15s %10.2f €\n", 1, "Джинси", "Жіночий
одяг", 1500.78);
        formatter.format("%-3d %-15s %-15s %10.2f €\n", 2, "Спідниця", "Жіночий
одяг", 1000.56);
        formatter.format("%-3d %-15s %-15s %10.2f €\n", 3, "Краватка", "Чоловічий
одяг", 500.78);
        formatter.format("%-3d %-15s %-15s %10.2f €\n", 4, "Сорочка", "Чоловічий
одяг", 750.00);
        formatter.format("%-3d %-15s %-15s %10.2f €\n", 5, "Сукня", "Жіночий
одяг", 2000.45);
        formatter.format("%-3d %-15s %-15s %10.2f €\n", 6, "Шкарпетки", "Чоловічий
одяг", 50.30);
        formatter.format("%-3d %-15s %-15s %10.2f €\n", 7, "Шапка", "Чоловічий
одяг", 300.60);
        formatter.format("%-3d %-15s %-15s %10.2f €\n", 8, "Шарф", "Чоловічий
одяг", 120.00);
        formatter.format("%-3d %-15s %-15s %10.2f €\n", 9, "Блуза", "Жіночий
одяг", 1800.90);
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

```

        formatter.format("%-3d %-15s %-15s %10.2f €\n", 10, "Куртка", "Чоловічий одяг", 3500.75);

        formatter.format("=====\n");
        formatter.format("Разом: %39.2f €\n", 1500.78 + 1000.56 + 500.78 + 750.00 + 2000.45 + 50.30 + 300.60 + 120.00 + 1800.90 + 3500.75);

        try (PrintWriter writer = new PrintWriter(new FileWriter(filePath))) {
            writer.print(formatter.toString());
        }
        formatter.close();
    }

    private static void readReceiptFromFile(String filePath) throws IOException {
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        }
    }
}

```

Дата та час покупки: 26.10.2024 13:25:12			
=====			
№	Товар	Категорія	Ціна
=====			
1	Джинси	Жіночий одяг	1500.78 €
2	Спідниця	Жіночий одяг	1000.56 €
3	Краватка	Чоловічий одяг	500.78 €
4	Сорочка	Чоловічий одяг	750.00 €
5	Сукня	Жіночий одяг	2000.45 €
6	Шкарпетки	Чоловічий одяг	50.30 €
7	Шапка	Чоловічий одяг	300.60 €
8	Шарф	Чоловічий одяг	120.00 €
9	Блуза	Жіночий одяг	1800.90 €
10	Куртка	Чоловічий одяг	3500.75 €
=====			
Разом:			11525.12 €

**Рис.6.2. Результат роботи**

#### 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`.

#### **FileCopier.java:**

```

package Task_04;
import java.io.*;

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public class FileCopier {

    public static void copyTextFile(String sourcePath, String destinationPath) {
        try (BufferedReader reader = new BufferedReader(new FileReader(
            sourcePath));
            BufferedWriter writer = new BufferedWriter(new FileWriter(
            destinationPath))) {

            String line;
            while ((line = reader.readLine()) != null) {
                writer.write(line);
                writer.newLine();
            }
            System.out.println("The text file was copied successfully.");

        } catch (IOException e) {
            System.out.println("Error copying a text file: " + e.getMessage());
        }
    }

    public static void copyBinaryFile(String sourcePath, String destinationPath) {
        try (FileInputStream inputStream = new FileInputStream(sourcePath);
            FileOutputStream outputStream = new FileOutputStream(
            destinationPath)) {

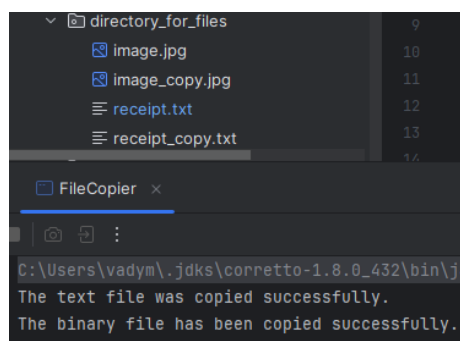
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }
            System.out.println("The binary file has been copied successfully.");

        } catch (IOException e) {
            System.out.println("Error copying a binary file: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        String textSource = "Java-LAB-06/src/directory_for_files/receipt.txt";
        String textDestination = "Java-LAB-06/src/directory_for_files/receipt_copy.txt";
        copyTextFile(textSource, textDestination);

        String imageSource = "Java-LAB-06/src/directory_for_files/image.jpg";
        String imageDestination = "Java-LAB-06/src/directory_for_files/image_copy.jpg";
        copyBinaryFile(imageSource, imageDestination);
    }
}

```



**Рис.6.3. Результат роботи**

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

## 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.

### FileEditor.java:

```
package Task_05;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;

public class FileEditor {

    public static void insertTextAtLinePositions(String filePath, int[] lineNumbers, String[] texts) {
        if (lineNumbers.length != texts.length) {
            System.out.println("Error: The number of items and texts do not match.");
            return;
        }
        try {
            List<String> fileLines = Files.readAllLines(Paths.get(filePath));
            List<Integer> sortedLineNumbers = new ArrayList<>();
            for (int pos : lineNumbers) sortedLineNumbers.add(pos);
            sortedLineNumbers.sort(Integer::compareTo);

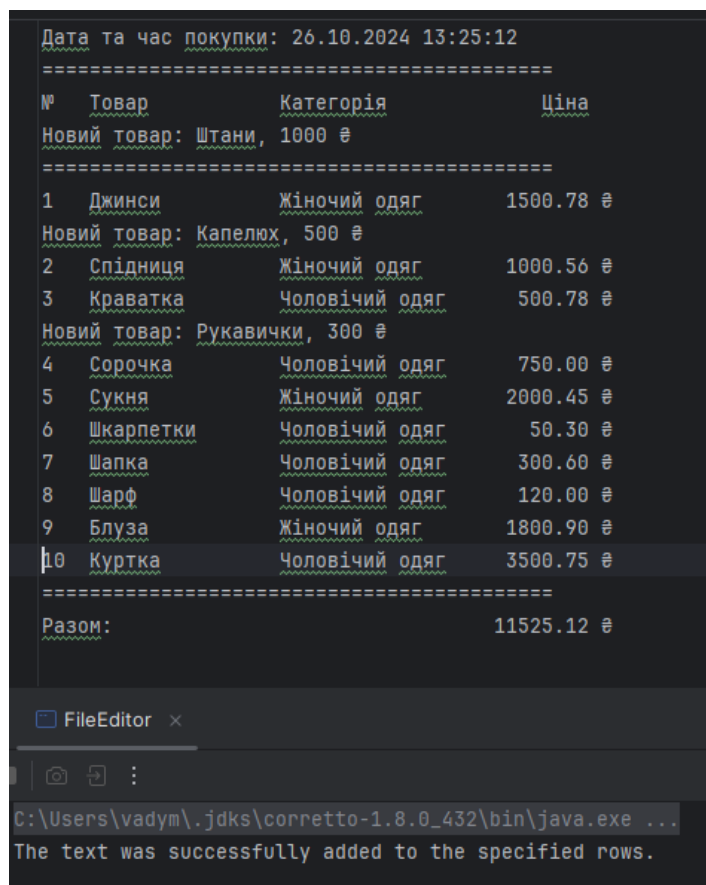
            int offset = 0;
            for (int i = 0; i < texts.length; i++) {
                int insertLine = sortedLineNumbers.get(i) + offset;
                if (insertLine >= 0 && insertLine < fileLines.size()) {
                    fileLines.add(insertLine, texts[i]);
                    offset++;
                } else {
                    System.out.println("Error: string " + insertLine + " doesn't exist in the file.");
                }
            }

            Files.write(Paths.get(filePath), fileLines);
            System.out.println("The text was successfully added to the specified rows.");
        } catch (IOException e) {
            System.out.println("Error while working with the file: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        String filePath = "Java-LAB-06/src/directory_for_files/receipt.txt";
        int[] lineNumbers = {3, 5, 7};
        String[] texts = {
            "Новий товар: Штани, 1000 ₴",
            "Новий товар: Капелюх, 500 ₴",
            "Новий товар: Рукавички, 300 ₴"
        };
        insertTextAtLinePositions(filePath, lineNumbers, texts);
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

```
}
}
```



**Рис.6.4. Результат роботи**

#### 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).

#### FileManager.java:

```
package Task_06;
import java.io.File;
import java.io.IOException;

public class FileManager {

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

```

// 1.
    File innerDirectory = new File("Java-LAB-06/src/inner_directory");
    if (innerDirectory.mkdir()) {
        System.out.println("Folder created: " + innerDirectory.getName());
    } else {
        System.out.println("Folder creation failed.");
    }

// 2.
    System.out.println("Absolute path: " + innerDirectory.getAbsolutePath());

// 3.
    String parent = innerDirectory.getParent();
    System.out.println("The name of the parent directory: " + parent);

// 4.
    File file1 = new File(innerDirectory, "file1.txt");
    File file2 = new File(innerDirectory, "file2.txt");
    if (file1.createNewFile() && file2.createNewFile()) {
        System.out.println("The files are created: " + file1.getName() +
            ", " + file2.getName());
    } else {
        System.out.println("Failed to create files.");
    }

// 5.
    if (file2.delete()) {
        System.out.println("The file has been deleted: " +
            file2.getName());
    } else {
        System.out.println("Couldn't get the file.");
    }

// 6.
    File renamedDirectory = new File("Java-LAB-06/src/renamed_inner_directory");
    if (innerDirectory.renameTo(renamedDirectory)) {
        System.out.println("The folder is renamed to: " + renamedDirectory.getName());
    } else {
        System.out.println("The folder could not be renamed.");
    }

// 7.
    File directoryForFiles = new File("Java-LAB-06/src/directory_for_files");
    if (directoryForFiles.exists() && directoryForFiles.isDirectory()) {
        File[] files = directoryForFiles.listFiles();
        if (files != null) {
            System.out.println("Contents of the folder " + directoryForFiles.getName() + ":");
            for (File f : files) {
                String type = f.isDirectory() ? "Folder" : "File.";
                System.out.printf("%s - %s, Size.: %d byte\n",
                    f.getName(), type, f.length());
            }
        } else {
            System.out.println("The folder is empty.");
        }
    } else {
        System.out.println("The directory_for_files folder does not exist.");
    }

    } catch (IOException e) {
        System.out.println("There was an error:" + e.getMessage());
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}
}

Folder created: inner_directory
Absolute path: C:\Users\vadym\Desktop\Y3S1-Java\Java-LAB-06\Java-LAB-06\src\inner_directory
The name of the parent directory: Java-LAB-06\src
The files are created: file1.txt, file2.txt
The file has been deleted: file2.txt
The folder could not be renamed.
Contents of the folder directory_for_files:
image.jpg - File., Size.: 19790 byte
image_copy.jpg - File., Size.: 19790 byte
receipt.txt - File., Size.: 1168 byte
receipt_copy.txt - File., Size.: 1023 byte

```

**Рис.6.5. Результат роботи**

### 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`.

```

ZipManager.java:
package Task_07;
import java.io.*;
import java.util.zip.*;

public class ZipManager {

    private static final String ZIP_FILE = "Java-LAB-06/src/archive.zip";
    public static void main(String[] args) {
        File directory = new File("Java-LAB-06/src/directory_for_files");
        try {
            createZip(directory);
            System.out.println("The files have been successfully added to the archive.");
            listFilesInZip(ZIP_FILE);
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }

    public static void createZip(File directory) throws IOException {
        try (ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(ZIP_FILE))) {
            if (directory.exists() && directory.isDirectory()) {
                File[] files = directory.listFiles();
                if (files != null) {
                    for (File file : files) {
                        if (file.isFile()) {
                            addFileToZip(file, zos);
                        }
                    }
                }
            } else {
                System.out.println("Folder not found.");
            }
        }
    }
}

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				40
Змн.	Арк.	№ докум.	Підпис	Дата		



```

    }
    private static void addFileToZip(File file, ZipOutputStream zos) throws IOException {
        try (FileInputStream fis = new FileInputStream(file)) {
            ZipEntry zipEntry = new ZipEntry(file.getName());
            zos.putNextEntry(zipEntry);

            byte[] buffer = new byte[1024];
            int length;
            while ((length = fis.read(buffer)) >= 0) {
                zos.write(buffer, 0, length);
            }

            zos.closeEntry();
            System.out.println("Added to the archive: " + file.getName());
        }
    }

    public static void listFilesInZip(String zipFile) throws IOException {
        try (ZipInputStream zis = new ZipInputStream(new FileInputStream(zipFile))) {
            ZipEntry entry;
            System.out.println("The contents of the archive:");
            while ((entry = zis.getNextEntry()) != null) {
                System.out.println(" - " + entry.getName());
                zis.closeEntry();
            }
        }
    }
}

```

```

Added to the archive: image.jpg
Added to the archive: image_copy.jpg
Added to the archive: receipt.txt
Added to the archive: receipt_copy.txt
The files have been successfully added to the archive.
The contents of the archive:
- image.jpg
- image_copy.jpg
- receipt.txt
- receipt_copy.txt

```

**Рис.6.6. Результат роботи**

**Висновок:** я обробив виключних ситуація, створив власні класи винятків, працював з потоками вводу-виводу.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

## Лабораторна робота № 7

**Тема:** Багатопоточне програмування в Java.

**Мета роботи:** Практика роботи з потоками в Java.

### Хід роботи

**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, RUNNUNG, TERMINATED.

#### Main.java:

```
package com.education.ztu.TASK_02;

public class Main {
    public static void main(String[] args) {
        MyThread myThread = new MyThread("MyThread");
        System.out.println("The state of the thread before starting: " +
myThread.getState());

        myThread.start();
        System.out.println("The name of the stream: " + myThread.getName());
        System.out.println("Flow priority: " + myThread.getPriority());
        System.out.println("Whether the stream is live: " + myThread.isAlive());
        System.out.println("Whether the flow is a daemon: " +
myThread.isDaemon());
        System.out.println("The state of the thread after launch: " +
myThread.getState());
        myThread.setName("NewMyThread");
        myThread.setPriority(Thread.MAX_PRIORITY);
        System.out.println("New thread name: " + myThread.getName());
        System.out.println("New thread priority: " + myThread.getPriority());

        try {
            myThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        System.out.println("The state of the thread after completion: " +
myThread.getState());

        Thread mainThread = Thread.currentThread();
        System.out.println("The name of the main thread: " + main-
Thread.getName());
        System.out.println("The priority of the main thread: " + main-
Thread.getPriority());
    }
}

```

### MyThread.java:

```

package com.education.ztu.TASK_02;

class MyThread extends Thread {

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

    @Override
    public void run() {
        for (int i = 0; i < 100; i++) {
            System.out.println("I love programming!!!");
        }
    }
}

```

```

I love programming!!!
I love programming!!!
The state of the thread after completion: TERMINATED
The name of the main thread: main
The priority of the main thread: 5

```

**Рис.7.1. Результат роботи**

**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().

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

### Main.java:

```
package com.education.ztu.TASK_03;

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

        Thread thread = new Thread(task, "Thread-1");
        thread.start();
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        thread.interrupt();
    }
}
```

### MyRunnable.java:

```
package com.education.ztu.TASK_03;
public class MyRunnable implements Runnable {

    @Override
    public void run() {
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            System.out.println(Thread.currentThread().getName() + ": Потік було перервано перед початком.");
            return;
        }

        for (int i = 0; i <= 10000; i++) {
            if (i % 10 == 0) {
                System.out.println(Thread.currentThread().getName() + ": " + i);
            }

            if (i == 9990 || Thread.currentThread().isInterrupted()) {
                System.out.println(Thread.currentThread().getName() + ": Розрахунок завершено!!!");
                return;
            }
        }
    }
}
```

```
Thread-1: 9970
Thread-1: 9980
Thread-1: 9990
Thread-1: Розрахунок завершено!!!
```

Рис.7.2. Результат роботи

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

- Створити клас, який реалізує інтерфейс Runnable.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

- 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.

#### ArithmeticProgressionRunnable.java:

```
package com.education.ztu.TASK_04;

public class ArithmeticProgressionRunnable implements Runnable {
    private static int result = 1;
    private synchronized void printNextProgression() {
        if (result <= 100) {
            System.out.print(result + " ");
            result++;
        }
    }

    @Override
    public void run() {
        while (result <= 100) {
            printNextProgression();
            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                System.out.println(Thread.currentThread().getName() + ": Потік перервано.");
            }
            return;
        }
    }
}
```

#### Main.java:

```
package com.education.ztu.TASK_04;

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

        Thread thread1 = new Thread(task, "Thread-1");
        Thread thread2 = new Thread(task, "Thread-2");
        Thread thread3 = new Thread(task, "Thread-3");

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

3 94 95 96 97 98 99 100

**Рис.7.3. Результат роботи**

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

**Завдання 5.** Переробити 4 завдання використовуючи блок синхронізації.

#### ArithmeticProgressionRunnable.java:

```
package com.education.ztu.TASK_05;

public class ArithmeticProgressionRunnable implements Runnable {
    private static int result = 1;

    @Override
    public void run() {
        while (result <= 100) {
            synchronized (ArithmeticProgressionRunnable.class) {
                if (result <= 101) {
                    System.out.print(result + " ");
                    result++;
                }
            }

            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                System.out.println(Thread.currentThread().getName() + ": The
stream is interrupted.");
                return;
            }
        }
    }
}
```

#### Main.java:

```
package com.education.ztu.TASK_05;

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

        Thread thread1 = new Thread(task, "Thread-1");
        Thread thread2 = new Thread(task, "Thread-2");
        Thread thread3 = new Thread(task, "Thread-3");

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

96 97 98 99 100 101

**Рис.7.4. Результат роботи**

**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.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

- Тут необхідно використати wait() і notify().

### Printer.java:

```
package com.education.ztu.TASK_06;

public class Printer implements Runnable {
    private final SharedData sharedData;

    public Printer(SharedData sharedData) {
        this.sharedData = sharedData;
    }

    @Override
    public void run() {
        while (true) {
            String data = sharedData.getData();

            if ("exit".equalsIgnoreCase(data)) {
                break;
            }

            System.out.println("Data obtained: " + data);
        }
    }
}
```

### Reader.java:

```
package com.education.ztu.TASK_06;
import java.util.Scanner;

public class Reader implements Runnable {
    private final SharedData sharedData;

    public Reader(SharedData sharedData) {
        this.sharedData = sharedData;
    }

    @Override
    public void run() {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.print("Enter a string (or 'exit' to exit): ");
            String input = scanner.nextLine();
            sharedData.setData(input);

            if ("exit".equalsIgnoreCase(input)) {
                break;
            }

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
        scanner.close();
    }
}
```

### ReaderPrinterExample.java:

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```
package com.education.ztu.TASK_06;

public class ReaderPrinterExample {
    public static void main(String[] args) {
        SharedData sharedData = new SharedData();

        Thread readerThread = new Thread(new Reader(sharedData));
        Thread printerThread = new Thread(new Printer(sharedData));

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

### SharedData.java:

```
package com.education.ztu.TASK_06;

public class SharedData {
    private String data;

    public synchronized void setData(String data) {
        this.data = data;
        notify();
    }

    public synchronized String getData() {
        try {
            wait();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        return data;
    }
}
```

```
Enter a string (or 'exit' to exit): 123
Data obtained: 123
Enter a string (or 'exit' to exit): exit
```

**Рис.7.5. Результат роботи**

**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.

### Main.java:

```
package com.education.ztu.TASK_07;
import java.util.Random;
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				48
Змн.	Арк.	№ докум.	Підпис	Дата		



```

import java.util.concurrent.Callable;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.List;
import java.util.ArrayList;

public class Main {
    private static final int ARRAY_SIZE = 1_000_000;
    private static final int THREAD_COUNT = 5;

    public static void main(String[] args) throws Exception {
        int[] array = new int[ARRAY_SIZE];
        Random random = new Random();

        for (int i = 0; i < ARRAY_SIZE; i++) {
            array[i] = random.nextInt(1000);
        }

        long startTime = System.currentTimeMillis();
        int singleThreadSum = calculateSumSingleThread(array);
        long singleThreadDuration = System.currentTimeMillis() - startTime;

        System.out.println("Suma (single stream): " + singleThreadSum);
        System.out.println("Execution time (single-threaded): " + singleThreadDuration + " ms");

        startTime = System.currentTimeMillis();
        int multiThreadSum = calculateSumMultiThread(array);
        long multiThreadDuration = System.currentTimeMillis() - startTime;

        System.out.println("Amount (multi-threaded): " + multiThreadSum);
        System.out.println("Execution time (multi-threaded): " + multiThreadDuration + " ms");
    }

    private static int calculateSumSingleThread(int[] array) {
        int sum = 0;
        for (int num : array) {
            sum += sumOfDigits(num);
        }
        return sum;
    }

    private static int calculateSumMultiThread(int[] array) throws Exception {
        ExecutorService executorService = Executors.newFixedThreadPool(THREAD_COUNT);
        List<Future<Integer>> futures = new ArrayList<>();

        int partSize = ARRAY_SIZE / THREAD_COUNT;
        for (int i = 0; i < THREAD_COUNT; i++) {
            int start = i * partSize;
            int end = (i == THREAD_COUNT - 1) ? ARRAY_SIZE : (i + 1) * partSize;

            Callable<Integer> task = new SumTask(array, start, end);
            futures.add(executorService.submit(task));
        }

        int totalSum = 0;
        for (Future<Integer> future : futures) {
            totalSum += future.get();
        }
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        executorService.shutdown();
        return totalSum;
    }

    private static int sumOfDigits(int number) {
        int sum = 0;
        while (number != 0) {
            sum += number % 10;
            number /= 10;
        }
        return sum;
    }
}

```

### SumTask.java:

```

package com.education.ztu.TASK_07;
import java.util.concurrent.Callable;

public class SumTask implements Callable<Integer> {
    private final int[] array;
    private final int start;
    private final int end;

    public SumTask(int[] array, int start, int end) {
        this.array = array;
        this.start = start;
        this.end = end;
    }

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

    private int sumOfDigits(int number) {
        int sum = 0;
        while (number != 0) {
            sum += number % 10;
            number /= 10;
        }
        return sum;
    }
}

```

```

Suma (single stream): 13504226
Execution time (single-threaded): 6 ms
Amount (multi-threaded): 13504226
Execution time (multi-threaded): 26 ms

```

**Рис.7.6. Результат роботи**

**Висновок:** я попрактикувався з потоками в Java.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

## 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.

### Хід роботи

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

**Завдання 3.** Написати лямбда вирази для вбудованих функціональних інтерфейсів:

a) 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.

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.

#### MainA.java:

```
package com.education.ztu.TASK_02_03;
import java.util.function.Predicate;

public class MainA {
    public static void main(String[] args) {
        Predicate<String> isNumeric = str -> {
            try {
                Double.parseDouble(str);
                return true;
            }
        }
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        } catch (NumberFormatException e) {
            return false;
        }
    };

    System.out.println(isNumeric.test("123")); // true
    System.out.println(isNumeric.test("abc")); // false

    Predicate<String> isEmpty = str -> !str.isEmpty();
    Predicate<String> isValidNumber = isNumeric.and(isEmpty);

    System.out.println(isValidNumber.test("123")); // true
    System.out.println(isValidNumber.test("")); // false
}
}

```

### MainB.java:

```

package com.education.ztu.TASK_02_03;

import java.util.function.Consumer;

public class MainB {
    public static void main(String[] args) {
        Consumer<String> startClass = str -> System.out.println("The pair started at 8:30 a.m.");
        Consumer<String> endClass = str -> System.out.println("The pair ended at 9:50");

        startClass.andThen(endClass).accept("Pair");
    }
}

```

### MainC.java:

```

package com.education.ztu.TASK_02_03;

import java.util.function.Supplier;

public class MainC {
    public static void main(String[] args) {
        Supplier<String> upperCaseSentence = () -> "this sentence is in upper case".toUpperCase();

        System.out.println(upperCaseSentence.get());
    }
}

```

### MainD.java:

```

package com.education.ztu.TASK_02_03;

import java.util.function.Function;
import java.util.Arrays;

public class MainD {
    public static void main(String[] args) {
        Function<String, Integer> multiplyNumbers = str -> Arrays.stream(str.split(" "))
            .map(Integer::parseInt)
            .reduce(1, (a, b) -> a * b);

        System.out.println(multiplyNumbers.apply("2 3 4")); // 24
        System.out.println(multiplyNumbers.apply("1 5 7")); // 35
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

```

    }
}

Printable.java:
package com.education.ztu.TASK_02_03;

@FunctionalInterface
interface Printable {
    void print();
}

```

```

true
false
true
false

```

**Рис.8.1. Результат роботи MainA**

```

The pair started at 8:30 a.m.
The pair ended at 9:50

```

**Рис.8.2. Результат роботи MainB**

```

THIS SENTENCE IS IN UPPER CASE

```

**Рис.8.3. Результат роботи MainC**

```

24
35

```

**Рис.8.4. Результат роботи MainD**

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

```

Main.java:
package com.education.ztu.TASK_04;

import java.util.*;
import java.util.stream.Collectors;

public class Main {

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public static void main(String[] args) {
    List<Product> products = Arrays.asList(
        new Product("Laptop", "BrandA", 1200, 5),
        new Product("Phone", "BrandB", 800, 10),
        new Product("Tablet", "BrandA", 400, 7),
        new Product("Monitor", "BrandC", 300, 15),
        new Product("Headphones", "BrandB", 150, 20)
    );

    // 1. Get all brands and display them in the console
    System.out.println("All brands:");
    products.stream()
        .map(Product::getBrand)
        .distinct()
        .forEach(System.out::println);

    // 2. Get 2 products with a price less than 1000
    System.out.println("\nTwo products with a price of less than 1000:");
    products.stream()
        .filter(p -> p.getPrice() < 1000)
        .limit(2)
        .forEach(System.out::println);

    // 3. Getting the sum of all types of goods
    int totalCount = products.stream()
        .map(Product::getCount)
        .reduce(0, Integer::sum);
    System.out.println("\nThe sum of all types of goods in the warehouse: " +
        totalCount);

    // 4. Group products by brand
    System.out.println("\nProducts are grouped by brand:");
    Map<String, List<Product>> groupedByBrand = products.stream()
        .collect(Collectors.groupingBy(Product::getBrand));
    groupedByBrand.forEach((brand, productList) -> {
        System.out.println(brand + ": " + productList);
    });

    // 5. Sort products by ascending price and return an array
    System.out.println("\nProducts sorted by price:");
    List<Product> sortedByPrice = products.stream()
        .sorted(Comparator.comparing(Product::getPrice))
        .collect(Collectors.toList());
    sortedByPrice.forEach(System.out::println);

    double totalValue = products.stream()
        .mapToDouble(p -> p.getPrice() * p.getCount())
        .sum();
    System.out.println("\nThe total value of all goods in the warehouse: " +
        totalValue);
}

```

## Product.java:

```

package com.education.ztu.TASK_04;

public class Product {
    private String name;
    private String brand;
    private double price;
    private int count;
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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;
}

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

```

#### BrandC

Two products with a price of less than 1000:

Product{name='Phone', brand='BrandB', price=800.0, count=10}

Product{name='Tablet', brand='BrandA', price=400.0, count=7}

The sum of all types of goods in the warehouse: 57

Products are grouped by brand:

BrandC: [Product{name='Monitor', brand='BrandC', price=300.0, count=15}]

BrandA: [Product{name='Laptop', brand='BrandA', price=1200.0, count=5}, Product{name='Tablet', brand='BrandA', price=400.0, count=7}]

BrandB: [Product{name='Phone', brand='BrandB', price=800.0, count=10}, Product{name='Headphones', brand='BrandB', price=150.0, count=20}]

Products sorted by price:

Product{name='Headphones', brand='BrandB', price=150.0, count=20}

Product{name='Monitor', brand='BrandC', price=300.0, count=15}

Product{name='Tablet', brand='BrandA', price=400.0, count=7}

Product{name='Phone', brand='BrandB', price=800.0, count=10}

Product{name='Laptop', brand='BrandA', price=1200.0, count=5}

The total value of all goods in the warehouse: 24300.0

**Рис.8.5. Результат роботи**

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонківський В.І.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

**Завдання 5.** Посилання на методи чи конструктори. В попередньому завданні, де це можливо, виклики переробити на посилання на методи чи конструктори.

### Main.java:

```
package com.education.ztu.TASK_05;

import java.util.*;
import java.util.stream.Collectors;

public class Main {
    public static void main(String[] args) {
        List<Product> products = Arrays.asList(
            new Product("Laptop", "BrandA", 1200, 5),
            new Product("Phone", "BrandB", 800, 10),
            new Product("Tablet", "BrandA", 400, 7),
            new Product("Monitor", "BrandC", 300, 15),
            new Product("Headphones", "BrandB", 150, 20)
        );
        // 1. Get all brands and display them in the console
        System.out.println("All brands:");
        products.stream()
            .map(Product::getBrand) // Using a method reference to get a brand
            .distinct()
            .forEach(System.out::println); // Reference to a method for out-
putting to the console
        // 2. Get 2 products with a price less than 1000
        System.out.println("\nTwo products with a price of less than 1000:");
        products.stream()
            .filter(p -> p.getPrice() < 1000)
            .limit(2)
            .forEach(System.out::println);
        // 3. Getting the sum of all types of goods
        int totalCount = products.stream()
            .map(Product::getCount) // Reference to the method for getting the
quantity
            .reduce(0, Integer::sum); // Reference to the method to summarise
        System.out.println("\nThe sum of all types of goods in the warehouse: " +
totalCount);
        // 4. Group products by brand
        System.out.println("\nProducts are grouped by brand:");
        Map<String, List<Product>> groupedByBrand = products.stream()
            .collect(Collectors.groupingBy(Product::getBrand)); // Reference
to the method for grouping by brand
        groupedByBrand.forEach((brand, productList) -> System.out.println(brand +
": " + productList));

        // 5. Sort products by ascending price and return an array
        System.out.println("\nProducts sorted by price:");
        List<Product> sortedByPrice = products.stream()
            .sorted(Comparator.comparing(Product::getPrice)) // Reference to
the method for sorting by price
            .collect(Collectors.toList());
        sortedByPrice.forEach(System.out::println); // Reference to the method to
output

        double totalValue = products.stream()
            .mapToDouble(p -> p.getPrice() * p.getCount()) // Using lambda to
calculate the total cost of goods
            .sum();
        System.out.println("\nThe total value of all goods in the warehouse: " +
totalValue);
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				56
Змн.	Арк.	№ докум.	Підпис	Дата		



```

    }
}

Product.java:
package com.education.ztu.TASK_05;

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

```

```

All brands:
BrandA
BrandB
BrandC

Two products with a price of less than 1000:
Product{name='Phone', brand='BrandB', price=800.0, count=10}
Product{name='Tablet', brand='BrandA', price=400.0, count=7}

The sum of all types of goods in the warehouse: 57

Products are grouped by brand:
BrandC: [Product{name='Monitor', brand='BrandC', price=300.0, count=15}]
BrandA: [Product{name='Laptop', brand='BrandA', price=1200.0, count=5}, Product{name='Tablet', brand='BrandA', price=400.0, count=7}]
BrandB: [Product{name='Phone', brand='BrandB', price=800.0, count=10}, Product{name='Headphones', brand='BrandB', price=150.0, count=20}]

Products sorted by price:
Product{name='Headphones', brand='BrandB', price=150.0, count=20}
Product{name='Monitor', brand='BrandC', price=300.0, count=15}
Product{name='Tablet', brand='BrandA', price=400.0, count=7}
Product{name='Phone', brand='BrandB', price=800.0, count=10}
Product{name='Laptop', brand='BrandA', price=1200.0, count=5}

The total value of all goods in the warehouse: 24300.0

```

**Рис.8.6. Результат роботи**

**Завдання 6.** Використання Optional та його методів. Знайти максимальне значення з масиву чисел, в іншому випадку повернути рядок «Числа відсутні».

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

## Main.java:

```
package com.education.ztu.TASK_06;
import java.util.*;
import java.util.stream.Collectors;
public class Main {
    public static void main(String[] args) {
        List<Product> products = Arrays.asList(
            new Product("Laptop", "BrandA", 1200, 5),
            new Product("Phone", "BrandB", 800, 10),
            new Product("Tablet", "BrandA", 400, 7),
            new Product("Monitor", "BrandC", 300, 15),
            new Product("Headphones", "BrandB", 150, 20)
        );
        // 1. Retrieve all brands and display them in the console
        System.out.println("All brands:");
        products.stream()
            .map(Product::getBrand) // Method reference to get the brand
            .distinct()
            .forEach(System.out::println); // Method reference for console
        output
        // 2. Retrieve 2 products with a price less than 1000
        System.out.println("\nTwo products with a price less than 1000:");
        products.stream()
            .filter(p -> p.getPrice() < 1000)
            .limit(2)
            .forEach(System.out::println);
        // 3. Calculate the total count of all products
        int totalCount = products.stream()
            .map(Product::getCount) // Method reference to get the count
            .reduce(0, Integer::sum); // Method reference to calculate the sum
        System.out.println("\nTotal count of all products in stock: " + total-
        Count);
        // 4. Group products by brand
        System.out.println("\nProducts grouped by brand:");
        Map<String, List<Product>> groupedByBrand = products.stream()
            .collect(Collectors.groupingBy(Product::getBrand)); // Method ref-
        erence for grouping by brand
        groupedByBrand.forEach((brand, productList) -> System.out.println(brand +
        ": " + productList));
        // 5. Sort products by ascending price and convert to a list
        System.out.println("\nProducts sorted by price:");
        List<Product> sortedByPrice = products.stream()
            .sorted(Comparator.comparing(Product::getPrice)) // Method refer-
        ence to sort by price
            .collect(Collectors.toList());
        sortedByPrice.forEach(System.out::println); // Method reference for con-
        sole output
        // Calculate the total value of all products
        double totalValue = products.stream()
            .mapToDouble(p -> p.getPrice() * p.getCount()) // Lambda to calcu-
        late the total value of products
            .sum();
        System.out.println("\nTotal value of all products in stock: " + to-
        talValue);
    }
}
```

## Product.java:

```
package com.education.ztu.TASK_06;
public class Product {
    private String name;
```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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;
}

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

```

```

All brands:
BrandA
BrandB
BrandC

Two products with a price less than 1000:
Product{name='Phone', brand='BrandB', price=800.0, count=10}
Product{name='Tablet', brand='BrandA', price=400.0, count=7}

Total count of all products in stock: 57

Products grouped by brand:
BrandC: [Product{name='Monitor', brand='BrandC', price=300.0, count=15}]
BrandA: [Product{name='Laptop', brand='BrandA', price=1200.0, count=5}, Product{name='Tablet', brand='BrandA', price=400.0, count=7}]
BrandB: [Product{name='Phone', brand='BrandB', price=800.0, count=10}, Product{name='Headphones', brand='BrandB', price=150.0, count=20}]

Products sorted by price:
Product{name='Headphones', brand='BrandB', price=150.0, count=20}
Product{name='Monitor', brand='BrandC', price=300.0, count=15}
Product{name='Tablet', brand='BrandA', price=400.0, count=7}
Product{name='Phone', brand='BrandB', price=800.0, count=10}
Product{name='Laptop', brand='BrandA', price=1200.0, count=5}

Total value of all products in stock: 24300.0

```

**Рис.8.7. Результат роботи**

**Висновок:** я попрактикувався з лямбда виразами, функціональними інтерфейсами, використав посилання на методи та Stream API при розробці програм на Java.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				59
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.

### Хід роботи

**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.

#### Employee.java:

```
package com.education.ztu.TASK_02;
public class Employee {
    private String fullName;
    private String birthDate;
    private String position;
    private String experience;
    private String address;
    private String email;
    private String phone;

    public Employee(String fullName, String birthDate, String position, String experience, String address, String email, String phone) {
        this.fullName = fullName;
        this.birthDate = birthDate;
        this.position = position;
        this.experience = experience;
        this.address = address;
        this.email = email;
        this.phone = phone;
    }

    public String getFullName() {
        return fullName;
    }
    public String getBirthDate() {
        return birthDate;
    }
    public void setBirthDate(String birthDate) {
        this.birthDate = birthDate;
    }
    public String getPosition() {
        return position;
    }
    public void setPosition(String position) {
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        this.position = position;
    }
    public String getEmail() {
        return email;
    }
    public String getPhone() {
        return phone;
    }

    @Override
    public String toString() {
        return "Employee{" +
            "fullName='" + fullName + '\'' +
            ", birthDate='" + birthDate + '\'' +
            ", position='" + position + '\'' +
            ", experience='" + experience + '\'' +
            ", address='" + address + '\'' +
            ", email='" + email + '\'' +
            ", phone='" + phone + '\'' +
            '}';
    }
}

```

### Main.java:

```

package com.education.ztu.TASK_02;

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

public class Main {
    public static void main(String[] args) {
        // List of employees
        List<Employee> employees = Arrays.asList(
            new Employee("Ivan Ivanenko", "20.05.1995", "Manager", "5 years",
                "Kyiv, Khreshchatyk St., 10", "ivan.ivanenko@example.com", "+380501234567"),
            new Employee("Petro Petrenko", "15.11.1987", "Developer", "10
years", "Lviv, Stryiska St., 5", "petro.petrenko@example.com", "+380931234567"),
            new Employee("Olena Olenko", "03.07.1992", "Analyst", "8 years",
                "Odesa, Deribasivska St., 1", "olena.olenko@example.com", "+380671234567"),
            new Employee("Sergiy Sergienko", "01.01.1980", "HR Manager", "12
years", "Kharkiv, Sumskaya St., 15", "sergiy.sergienko@example.com",
                "+380661234567"),
            new Employee("Natalia Natalenko", "30.09.1990", "Designer", "6
years", "Dnipro, Polya St., 8", "natalia.natalenko@example.com", "+380991234567"),
            new Employee("Andriy Andrienko", "25.03.1985", "Project Manager",
                "15 years", "Zaporizhzhia, Miru St., 20", "andriy.andrienko@example.com",
                "+380501111111"),
            new Employee("Maria Marienko", "10.12.1993", "Marketer", "4
years", "Kyiv, Hrushevskoho St., 25", "maria.marienko@example.com",
                "+380671000000")
        );

        // 1. Find phones and emails
        System.out.println("Found phones:");
        employees.forEach(emp -> System.out.println(emp.getPhone()));

        System.out.println("\nFound emails:");
        employees.forEach(emp -> System.out.println(emp.getEmail()));

        // 2. Change the date format for birth dates
        Pattern datePattern = Pattern.compile("(\\d{2})\\.\\. (\\d{2})\\.\\. (\\d{4})");
        employees.forEach(emp -> {

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Matcher matcher = datePattern.matcher(emp.getBirthDate());
        if (matcher.matches()) {
            String newDate = matcher.group(3) + "-" + matcher.group(2) + "-" +
matcher.group(1);
            emp.setBirthDate(newDate);
        }
    });

    // 3. Update positions
    employees.get(0).setPosition("Senior Manager");
    employees.get(3).setPosition("HR Director");
    employees.get(6).setPosition("Head of Marketing Department");

    // 4. Display updated employee data
    System.out.println("\nUpdated employee data:");
    employees.forEach(System.out::println);
}
}

```

```

Found phones:
+380501234567
+380931234567
+380671234567
+380661234567
+380991234567
+380501111111
+380671000000

Found emails:
ivan.ivanenko@example.com
petro.petrenko@example.com
olena.olenko@example.com
sergiy.sergienko@example.com
natalia.natalenko@example.com
andriy.andrienko@example.com
maria.marienko@example.com

Updated employee data:
Employee{fullName='Ivan Ivanenko', birthDate='1995-05-20', position='Senior Manager', experience='5 years', address=''}
Employee{fullName='Petro Petrenko', birthDate='1987-11-15', position='Developer', experience='10 years', address=''}
Employee{fullName='Olena Olenko', birthDate='1992-07-03', position='Analyst', experience='8 years', address='Odesa'}
Employee{fullName='Sergiy Sergienko', birthDate='1980-01-01', position='HR Director', experience='12 years', address=''}
Employee{fullName='Natalia Natalenko', birthDate='1990-09-30', position='Designer', experience='6 years', address=''}
Employee{fullName='Andriy Andrienko', birthDate='1985-03-25', position='Project Manager', experience='15 years', address=''}
Employee{fullName='Maria Marienko', birthDate='1993-12-10', position='Head of Marketing Department', experience='4 years', address=''}

```

**Рис.9.1. Результат роботи**

### **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.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

- Попрацювати з приватним полем (встановити та отримати значення).
- Результати роботи відобразити в консолі.

### CustomClass.java:

```
package com.education.ztu.TASK_03;

public class CustomClass {
    public String publicField;
    private int privateField;

    public CustomClass() {
        this.publicField = "Default";
        this.privateField = 0;
    }

    public CustomClass(String publicField, int privateField) {
        this.publicField = publicField;
        this.privateField = privateField;
    }

    public void publicMethod() {
        System.out.println("Public method called");
    }

    private String privateMethod(String message) {
        return "Private method says: " + message;
    }

    public String getPublicField() {
        return publicField;
    }

    public void setPublicField(String publicField) {
        this.publicField = publicField;
    }
}
```

### Main.java:

```
package com.education.ztu.TASK_03;

public class Main {
    public static void main(String[] args) {
        ReflectionDemo.main(args);
    }
}
```

### ReflectionDemo.java:

```
package com.education.ztu.TASK_03;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Method;

public class ReflectionDemo {

    public static void main(String[] args) {
        try {
            // Getting a Class 3 object in the following ways
            Class<?> cls1 = CustomClass.class;
            Class<?> cls2 = new CustomClass().getClass();
        }
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Class<?> cls3 =
Class.forName("com.education.ztu.TASK_03.CustomClass");

        System.out.println("Class obtained by three different ways:");
        System.out.println(cls1);
        System.out.println(cls2);
        System.out.println(cls3);
        System.out.println();

        // Withdrawing everything
        System.out.println("Fields:");
        for (Field field : cls1.getDeclaredFields()) {
            System.out.println("Name: " + field.getName() + ", Type: " +
field.getType().getName());
        }

        System.out.println("\nMethods:");
        for (Method method : cls1.getDeclaredMethods()) {
            System.out.print("Name: " + method.getName());
            System.out.print(", Return Type: " + meth-
od.getReturnType().getName());
            System.out.print(", Parameter Types: ");
            Class<?>[] paramTypes = method.getParameterTypes();
            for (Class<?> paramType : paramTypes) {
                System.out.print(paramType.getName() + " ");
            }
            System.out.println();
        }

        System.out.println("\nConstructors:");
        for (Constructor<?> constructor : cls1.getDeclaredConstructors()) {
            System.out.print("Name: " + constructor.getName());
            System.out.print(", Parameter Types: ");
            Class<?>[] paramTypes = constructor.getParameterTypes();
            for (Class<?> paramType : paramTypes) {
                System.out.print(paramType.getName() + " ");
            }
            System.out.println();
        }
        System.out.println();

        // An instance of a class
        CustomClass customInstance = (CustomClass)
cls1.getConstructor().newInstance();
        System.out.println("Instance created: " + customInstance);

        // The challenge of the public method
        Method publicMethod = cls1.getMethod("publicMethod");
        publicMethod.invoke(customInstance);

        // Accessing a private field (getting and setting a value)
        Field privateField = cls1.getDeclaredField("privateField");
        privateField.setAccessible(true);
        privateField.setInt(customInstance, 42);
        System.out.println("Private field value set to: " + private-
Field.getInt(customInstance));

        // Calling a private method
        Method privateMethod = cls1.getDeclaredMethod("privateMethod",
String.class);
        privateMethod.setAccessible(true);
        String result = (String) privateMethod.invoke(customInstance, "Hello
from Reflection!");

```



```

        System.out.println("Private method result: " + result);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

```

Class obtained by three different ways:
class com.education.ztu.TASK_03.CustomClass
class com.education.ztu.TASK_03.CustomClass
class com.education.ztu.TASK_03.CustomClass

Fields:
Name: publicField, Type: java.lang.String
Name: privateField, Type: int

Methods:
Name: getPublicField, Return Type: java.lang.String, Parameter Types:
Name: publicMethod, Return Type: void, Parameter Types:
Name: setPublicField, Return Type: void, Parameter Types: java.lang.String
Name: privateMethod, Return Type: java.lang.String, Parameter Types: java.lang.String

Constructors:
Name: com.education.ztu.TASK_03.CustomClass, Parameter Types:
Name: com.education.ztu.TASK_03.CustomClass, Parameter Types: java.lang.String int

Instance created: com.education.ztu.TASK_03.CustomClass@4554617c
Public method called
Private field value set to: 42
Private method result: Private method says: Hello from Reflection!

```

**Рис.9.2. Результат роботи**

#### **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.

#### **AnnotatedClass.java:**

```

package com.education.ztu.TASK_04;

@MyAnnotation(
    author = "John Doe",
    version = "2.0",
    description = "This is a sample class with MyAnnotation"
)
public class AnnotatedClass {
    public void display() {
        System.out.println("AnnotatedClass method is called.");
    }
}

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

### Main.java:

```
package com.education.ztu.TASK_04;

public class Main {
    public static void main(String[] args) {
        // Getting the AnnotatedClass class
        Class<AnnotatedClass> clazz = AnnotatedClass.class;

        // Checking the availability of an annotation
        if (clazz.isAnnotationPresent(MyAnnotation.class)) {
            // Retrieving data from an annotation
            MyAnnotation annotation = clazz.getAnnotation(MyAnnotation.class);

            // Data output to the console
            System.out.println("Author: " + annotation.author());
            System.out.println("Version: " + annotation.version());
            System.out.println("Description: " + annotation.description());
        } else {
            System.out.println("No annotation found on class AnnotatedClass.");
        }

        AnnotatedClass obj = new AnnotatedClass();
        obj.display();
    }
}
```

### MyAnnotated.java:

```
package com.education.ztu.TASK_04;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

// Creating an annotation
@Target(ElementType.TYPE) // Can be applied to classes or interfaces
@Retention(RetentionPolicy.RUNTIME) // Available at runtime
public @interface MyAnnotation {
    String author() default "Anonymous"; // The default value is
    String version() default "1.0"; // The default value is
    String description() default "No description provided"; // Extra field
}
```

```
Author: John Doe
Version: 2.0
Description: This is a sample class with MyAnnotation
AnnotatedClass method is called.
```

**Рис.9.3. Результат роботи**

**Висновок:** я попрактикувався з регулярними виразами, використав рефлексії, створив власні анотації.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

## 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.

### Хід роботи

#### 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.

#### 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)

#### 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)

#### Завдання 5. XML парсери:

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

#### 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.

**AgeComparator.java:**

```
package game;
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```
import java.util.Comparator;

public class AgeComparator implements Comparator<Participant> {
    @Override
    public int compare(Participant p1, Participant p2) {
        return Integer.compare(p1.getAge(), p2.getAge());
    }
}
```

## Employee.java:

```
package game;

public class Employee extends Participant {
    private static final long serialVersionUID = 1L;

    public Employee(String name, int age) {
        super(name, age);
    }
}
```

## Game.java:

```
package game;

public class Game {
    public static void main(String[] args) {
        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);

        Team<Schoolar> schoolarTeam = new Team<>("Dragon");
        schoolarTeam.addNewParticipant(schoolar1);
        schoolarTeam.addNewParticipant(schoolar2);

        Team<Student> studentTeam = new Team<>("Vpered");
        studentTeam.addNewParticipant(student1);
        studentTeam.addNewParticipant(student2);

        Team<Employee> employeeTeam = new Team<>("Robotyagi");
        employeeTeam.addNewParticipant(employee1);
        employeeTeam.addNewParticipant(employee2);

        schoolarTeam.playWith(new Team<>("Rozumnyky"));
        studentTeam.playWith(new Team<>("StudentPower"));
    }
}
```

## JSONParser.java:

```
package game;

import com.google.gson.Gson;
import game.Participant;
import game.Schoolar;
import game.Team;

public class JSONParser {

    public static void main(String[] args) {
        Gson gson = new Gson();
    }
}
```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        // Creating a Java object
        Team<Participant> team = new Team<>("Dragon");
        team.addNewParticipant(new Scholar("Ivan", 13));
        team.addNewParticipant(new Scholar("Mariya", 15));

        // Converting a Java object to JSON
        String json = gson.toJson(team);
        System.out.println("JSON Representation:");
        System.out.println(json);

        // Convert JSON to a Java object
        Team deserializedTeam = gson.fromJson(json, Team.class);
        System.out.println("\nDeserialized Team:");
        System.out.println(deserializedTeam);
    }
}

```

### Main.java:

```

package game;

public class Main {
    public static void main(String[] args) {
        Scholar scholar1 = new Scholar("Ivan", 13);
        Scholar scholar2 = new Scholar("Mariya", 15);

        Team<Scholar> scholarTeam = new Team<>("Dragon");
        scholarTeam.addNewParticipant(scholar1);
        scholarTeam.addNewParticipant(scholar2);

        System.out.println("Original team: " + scholarTeam);

        Team<Scholar> clonedTeam = scholarTeam.clone();
        System.out.println("Cloned team: " + clonedTeam);

        System.out.println("Equals: " + scholar1.equals(scholar2));
        System.out.println("HashCode of scholar1: " + scholar1.hashCode());
        System.out.println("HashCode of scholar2: " + scholar2.hashCode());
    }
}

```

### Participant.java:

```

package game;

import java.io.Serializable;

/**
 * Abstract class {@code Participant} represents a team participant.
 * Implements interfaces {@link Cloneable}, {@link Comparable}, and {@link Serializable}.
 */
public abstract class Participant implements Cloneable, Comparable<Participant>,
    Serializable {
    private static final long serialVersionUID = 1L;

    /**
     * The name of the participant.
     */
    private String name;

    /**
     * The age of the participant.
     */
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    */
    private int age;

    /**
     * Creates a participant with the specified name and age.
     *
     * @param name The name of the participant.
     * @param age The age of the participant.
     */
    public Participant(String name, int age) {
        this.name = name;
        this.age = age;
    }

    /**
     * Returns the name of the participant.
     *
     * @return The name of the participant.
     */
    public String getName() {
        return name;
    }

    /**
     * Sets the name of the participant.
     *
     * @param name The new name of the participant.
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * Returns the age of the participant.
     *
     * @return The age of the participant.
     */
    public int getAge() {
        return age;
    }

    /**
     * Sets the age of the participant.
     *
     * @param age The new age of the participant.
     */
    public void setAge(int age) {
        this.age = age;
    }

    /**
     * Creates a deep copy of the {@code Participant} object.
     *
     * @return A cloned {@code Participant} object.
     */
    @Override
    public Participant clone() {
        try {
            return (Participant) super.clone();
        } catch (CloneNotSupportedException e) {
            throw new RuntimeException("Clone not supported for Participant", e);
        }
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/**
 * Compares {@code Participant} objects by name.
 *
 * @param other Another {@code Participant} object to compare with.
 * @return The result of string comparison (participant name).
 */
@Override
public int compareTo(Participant other) {
    return this.name.compareTo(other.name);
}

/**
 * Returns a string representation of the {@code Participant} object.
 *
 * @return A string representation of the {@code Participant} object.
 */
@Override
public String toString() {
    return "Participant{" +
        "name='" + name + '\'' +
        ", age=" + age +
        '}';
}

/**
 * Checks equality between this and another object.
 *
 * @param obj The object to compare with.
 * @return {@code true} if the objects are equal; {@code false} otherwise.
 */
@Override
public boolean equals(Object obj) {
    if (this == obj) return true;
    if (obj == null || getClass() != obj.getClass()) return false;
    Participant that = (Participant) obj;
    return age == that.age && name.equals(that.name);
}

/**
 * Returns the hash code for the {@code Participant} object.
 *
 * @return The hash code of the object.
 */
@Override
public int hashCode() {
    return 31 * name.hashCode() + age;
}
}

```

### Schoolar.java:

```

package game;

public class Schoolar extends Participant {
    private static final long serialVersionUID = 1L;

    public Schoolar(String name, int age) {
        super(name, age);
    }
}

```

### SerializationDemo.java:

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

package game;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import java.io.*;

public class SerializationDemo {
    private static final Logger logger = LogManager
        .getLogger(SerializationDemo.class);

    public static void main(String[] args) {
        logger.info("The beginning of serialisation...");
        Scholar scholar = new Scholar("Ivan", 13);

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOut-
            putStream("scholar.ser"))) {
            oos.writeObject(scholar);
            logger.debug("Object {} is serialised", scholar.getName());
        } catch (IOException e) {
            logger.error("Serialisation error", e);
        }

        logger.info("The beginning of deserialisation...");
        try (ObjectInputStream ois = new ObjectInputStream(new FileIn-
            putStream("scholar.ser"))) {
            Scholar deserializedScholar = (Scholar) ois.readObject();
            logger.debug("Object {} deserialised", deserial-
                izedScholar.getName());
        } catch (IOException | ClassNotFoundException e) {
            logger.error("Deserialisation error", e);
        }
    }
}

```

### Student.java:

```

package game;

public class Student extends Participant {
    private static final long serialVersionUID = 1L;

    public Student(String name, int age) {
        super(name, age);
    }
}

```

### Team.java:

```

package game;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

/**
 * The Team class represents a team of participants of a specific type.
 *
 * @param <T> The type of participants in the team, which extends {@link Partici-

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				72
Змн.	Арк.	№ докум.	Підпис	Дата		



```

    participant}.
    */
    public class Team<T extends Participant> implements Serializable, Cloneable {
        private static final Logger logger = LogManager.getLogger(Team.class);
        private static final long serialVersionUID = 1L;

        /**
         * The name of the team.
         */
        private String name;

        /**
         * The list of participants in the team.
         */
        private List<T> participants = new ArrayList<>();

        /**
         * Creates a team with the specified name.
         *
         * @param name The name of the team.
         */
        public Team(String name) {
            this.name = name;
            logger.info("Team {} created", name);
        }

        /**
         * Adds a new participant to the team.
         *
         * @param participant The participant to be added.
         */
        public void addNewParticipant(T participant) {
            participants.add(participant);
            logger.debug("Added participant {} to team {}", participant.getName(),
name);
        }

        /**
         * Conducts a competition between two teams.
         *
         * @param team The other team to compete with.
         */
        public void playWith(Team<T> team) {
            logger.info("Team {} is playing against team {}", this.name, team.name);
        }

        /**
         * Creates a clone of the team with a deep copy of the participants list.
         *
         * @return The cloned team.
         */
        @Override
        public Team<T> clone() {
            try {
                @SuppressWarnings("unchecked")
                Team<T> clonedTeam = (Team<T>) super.clone();
                clonedTeam.participants = new ArrayList<>();
                for (T participant : this.participants) {
                    clonedTeam.participants.add((T) participant.clone());
                }
                logger.info("Cloned team {}", this.name);
                return clonedTeam;
            } catch (CloneNotSupportedException e) {

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        logger.error("Error cloning team {}", this.name, e);
        throw new RuntimeException("Clone not supported for Team", e);
    }
}
/**
 * Returns the string representation of the team.
 *
 * @return The string representation of the team.
 */
@Override
public String toString() {
    return "Team{" + "name='" + name + '\'' + ", participants=" + participants
+ '}' + ' ';
}
}

```

### XMLParser.java:

```

package TASK_05;
import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.File;

public class XMLParser {

    public static void readXML(String filePath) {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(new File(filePath));

            document.getDocumentElement().normalize();
            NodeList teamList = document.getElementsByTagName("team");

            for (int i = 0; i < teamList.getLength(); i++) {
                Node teamNode = teamList.item(i);
                if (teamNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element teamElement = (Element) teamNode;
                    System.out.println("Team: " + teamElement.getAttribute("name"));

                    NodeList participants = teamElement.getElementsByTagName("participant");
                    for (int j = 0; j < participants.getLength(); j++) {
                        Node participantNode = participants.item(j);
                        if (participantNode.getNodeType() == Node.ELEMENT_NODE) {
                            Element participantElement = (Element) participantNode;

                            String name = participantElement.getElementsByTagName("name").item(0).getTextContent();
                            String age = participantElement.getElementsByTagName("age").item(0).getTextContent();
                            System.out.println("Participant: " + name + ", Age: " + age);
                        }
                    }
                }
            }
        } catch (Exception e) {

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        e.printStackTrace();
    }
}

public static void saveXML(String filePath) {
    try {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.newDocument();

        Element root = document.createElement("teams");
        document.appendChild(root);

        Element team = document.createElement("team");
        team.setAttribute("name", "Dragon");
        root.appendChild(team);

        Element participant = document.createElement("participant");
        team.appendChild(participant);

        Element name = document.createElement("name");
        name.setTextContent("Ivan");
        participant.appendChild(name);

        Element age = document.createElement("age");
        age.setTextContent("13");
        participant.appendChild(age);

        TransformerFactory transformerFactory = TransformerFactory
ry.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        DOMSource source = new DOMSource(document);
        StreamResult result = new StreamResult(new File(filePath));
        transformer.transform(source, result);

        System.out.println("XML saved to " + filePath);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static void main(String[] args) {
    String filePath = "teams.xml";
    saveXML(filePath);
    readXML(filePath);
}
}

```

### log4j2.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
    <Appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} [%t] %-5level %log-
ger{36} - %msg%n" />
        </Console>
        <File name="File" fileName="logs/game.log">
            <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} [%t] %-5level %log-
ger{36} - %msg%n" />
        </File>
    </Appenders>
    <Loggers>

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    <Root level="info">
        <AppenderRef ref="Console" />
        <AppenderRef ref="File" />
    </Root>
</Loggers>
</Configuration>
pom.xml:
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>game-project</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
    </properties>

    <dependencies>
        <!-- Log4j Core -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.20.0</version>
        </dependency>
        <!-- Log4j SLF4J Implementation -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j-impl</artifactId>
            <version>2.20.0</version>
        </dependency>

        <!-- Gson -->
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.8</version>
        </dependency>
    </dependencies>
</project>

```

```

XML saved to teams.xml
Team: Dragon
Participant: Ivan, Age: 13

```

**Рис.10.1. Результат роботи XMLParser**

**Висновок:** Попрактикувався з XML та JSON парсерами, використав серіалізацію, логування та документування коду.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

## 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.

### Хід роботи

**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 проєкт.
- Створити базу даних store.
- Налаштувати з'єднання (параметри з'єднання повинні бути збережені в properties файлі та зчитуватись з ResourceBundle).

#### db.properties:

```
db.url=jdbc:mysql://localhost:3306/store
db.user=root
db.password=root
```

#### DatabaseConnector.java:

```
package TASK_01_02;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.ResourceBundle;

public class DatabaseConnector {
    private Connection connection;

    public void connect() {
        ResourceBundle bundle = ResourceBundle.getBundle("db");
        String url = bundle.getString("db.url");
        String username = bundle.getString("db.user");
        String password = bundle.getString("db.password");

        try {
            connection = DriverManager.getConnection(url, username, password);
            System.out.println("Connection successful.");
        } catch (SQLException e) {
            System.err.println("Connection failed: " + e.getMessage());
        }
    }

    public Connection getConnection() {
        return connection;
    }
}
```

#### Main.java:

```
package TASK_01_02;
public class Main {
    public static void main(String[] args) {
        DatabaseConnector dbConnector = new DatabaseConnector();
        dbConnector.connect();
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонківський В.І.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```
}  
}
```

Connection successful.

Рис.11.1. Результат роботи

**Завдання 3.** Робота з базою даних використовуючи клас Statement:

- Створити необхідні таблицю чи таблиці в базі даних
- 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.

#### DatabaseReader.java:

```
package TASK_03;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.Statement;  
import java.util.Properties;  
import java.io.FileInputStream;  
  
public class DatabaseReader {  
  
    private static Connection getConnection() throws Exception {  
        Properties props = new Properties();  
        try (FileInputStream fis = new FileInputStream("src/db.properties")) {  
            props.load(fis);  
        }  
        String url = props.getProperty("db.url");  
        String user = props.getProperty("db.user");  
        String password = props.getProperty("db.password");  
  
        return DriverManager.getConnection(url, user, password);  
    }  
  
    public static void displayProducts() {  
        String query = "SELECT * FROM products";  
        try (Connection connection = getConnection();  
            Statement statement = connection.createStatement();  
            ResultSet resultSet = statement.executeQuery(query)) {  
  
            System.out.println("Grocery List:");  
            System.out.println("-----");  
            System.out.printf("%-5s %-20s %-30s %-10s %-15s\n", "ID", "Name", "De-  
scription", "Price", "Category");  
            System.out.println("-----");  
            System.out.println("-----");  
  
            while (resultSet.next()) {  
                int id = resultSet.getInt("id");  
                String name = resultSet.getString("name");  
                String description = resultSet.getString("description");  
                double price = resultSet.getDouble("price");  
                String category = resultSet.getString("category");  
  
                System.out.printf("%-5d %-20s %-30s %-10.2f %-15s\n", id, name,  
description, price, category);  
            }  
        }  
    }  
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } catch (Exception e) {
        System.out.println("Error in data output: " + e.getMessage());
        e.printStackTrace();
    }
}

```

### Main.java:

```

package TASK_03;
public class Main {
    public static void main(String[] args) {
        DatabaseReader.displayProducts();
    }
}

```

Grocery List::				
ID	Name	Description	Price	Category
101	Product 1	Description of Product 1	10,99	Electronics
102	Product 2	Description of Product 2	12,99	Electronics
103	Product 3	Description of Product 3	5,49	Furniture
104	Product 4	Description of Product 4	8,75	Furniture
105	Product 5	Description of Product 5	3,99	Clothing
106	Product 6	Description of Product 6	7,49	Clothing
107	Product 7	Description of Product 7	11,25	Books
108	Product 8	Description of Product 8	6,89	Books
109	Product 9	Description of Product 9	4,99	Electronics
110	Product 10	Description of Product 10	9,99	Furniture

**Рис.11.2. Результат роботи**

**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.

### Main.java:

```

package TASK_04;
public class Main {
    public static void main(String[] args) {
        ProductManager.addProducts();
        ProductManager.displayProductsByCategory("Clothing");
        ProductManager.deleteAllProducts();
    }
}

```

### ProductManager.java:

```

package TASK_04;
import java.sql.Connection;
import java.sql.DriverManager;

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Properties;
import java.io.FileInputStream;
public class ProductManager {
    // Method for obtaining a compound
    private static Connection getConnection() throws Exception {
        Properties props = new Properties();
        try (FileInputStream fis = new FileInputStream("src/db.properties")) {
            props.load(fis);
        }
        String url = props.getProperty("db.url");
        String user = props.getProperty("db.user");
        String password = props.getProperty("db.password");

        return DriverManager.getConnection(url, user, password);
    }
    // Add products
    public static void addProducts() {
        String query = "INSERT INTO products (name, description, price, category)
VALUES (?, ?, ?, ?)";
        try (Connection connection = getConnection();
            PreparedStatement preparedStatement = connection.prepareStatement(query)) {
            // 5 new product
            preparedStatement.setString(1, "Product 11");
            preparedStatement.setString(2, "Description of Product 11");
            preparedStatement.setDouble(3, 15.99);
            preparedStatement.setString(4, "Clothing");
            preparedStatement.addBatch();
            preparedStatement.setString(1, "Product 12");
            preparedStatement.setString(2, "Description of Product 12");
            preparedStatement.setDouble(3, 19.99);
            preparedStatement.setString(4, "Clothing");
            preparedStatement.addBatch();
            preparedStatement.setString(1, "Product 13");
            preparedStatement.setString(2, "Description of Product 13");
            preparedStatement.setDouble(3, 25.49);
            preparedStatement.setString(4, "Books");
            preparedStatement.addBatch();
            preparedStatement.setString(1, "Product 14");
            preparedStatement.setString(2, "Description of Product 14");
            preparedStatement.setDouble(3, 9.99);
            preparedStatement.setString(4, "Books");
            preparedStatement.addBatch();
            preparedStatement.setString(1, "Product 15");
            preparedStatement.setString(2, "Description of Product 15");
            preparedStatement.setDouble(3, 29.99);
            preparedStatement.setString(4, "Electronics");
            preparedStatement.addBatch();
            preparedStatement.executeBatch();
            System.out.println("Goods successfully added.");
        } catch (Exception e) {
            System.out.println("Error when adding products: " + e.getMessage());
            e.printStackTrace();
        }
    }
    // Get products by category
    public static void displayProductsByCategory(String category) {
        String query = "SELECT * FROM products WHERE category = ?";
        try (Connection connection = getConnection();
            PreparedStatement preparedStatement = connection.prepareStatement(query)) {

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				80
Змн.	Арк.	№ докум.	Підпис	Дата		



```

        preparedStatement.setString(1, category);
        try (ResultSet resultSet = preparedStatement.executeQuery()) {
            System.out.println("Products in category \"" + category + "\":");
            System.out.println("-----");
            System.out.printf("%-5s %-20s %-30s %-10s %-15s\n", "ID", "Name",
"Description", "Price", "Category");
            System.out.println("-----");
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                String description = resultSet.getString("description");
                double price = resultSet.getDouble("price");
                String prodCategory = resultSet.getString("category");
                System.out.printf("%-5d %-20s %-30s %-10.2f %-15s\n", id,
name, description, price, prodCategory);
            }
        } catch (Exception e) {
            System.out.println("Error when receiving goods: " + e.getMessage());
            e.printStackTrace();
        }
    }
    // Delete all records from the table
    public static void deleteAllProducts() {
        String query = "DELETE FROM products";
        try (Connection connection = getConnection();
            PreparedStatement preparedStatement = connection.prepareStatement(query)) {

            int rowsDeleted = preparedStatement.executeUpdate();
            System.out.println("Delete all records from the table " + rowsDeleted);

        } catch (Exception e) {
            System.out.println("Error when deleting products: " + e.getMessage());
            e.printStackTrace();
        }
    }
}

```

Goods successfully added.

Products in category "Clothing":

```

-----
ID      Name          Description          Price      Category
-----
116     Product 11        Description of Product 11    15,99      Clothing
117     Product 12        Description of Product 12    19,99      Clothing
Delete all records from the table 5

```

**Рис.11.3. Результат роботи**

**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.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

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

### Main.java:

```
package TASK_05;
public class Main {
    public static void main(String[] args) {
        TransactionHandler.handleTransactions();
    }
}
```

### TransactionHandler.java:

```
package TASK_05;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.Savepoint;
import java.sql.Statement;
import java.sql.ResultSet;
import java.util.Properties;
import java.io.FileInputStream;

public class TransactionHandler {

    private static Connection getConnection() throws Exception {
        Properties props = new Properties();
        try (FileInputStream fis = new FileInputStream("src/db.properties")) {
            props.load(fis);
        }
        String url = props.getProperty("db.url");
        String user = props.getProperty("db.user");
        String password = props.getProperty("db.password");

        return DriverManager.getConnection(url, user, password);
    }

    public static void handleTransactions() {
        try (Connection connection = getConnection()) {
            connection.setAutoCommit(false); // Disable autocommit to keep them
            out of the way

            String insertQuery1 = "INSERT INTO products (name, description, price,
            category) VALUES (?, ?, ?, ?)";
            String insertQuery2 = "INSERT INTO products (name, description, price,
            category VALUES (?, ?, ?, ?)"; // Syntax error

            Savepoint savepoint = null;

            try (PreparedStatement statement1 = connec-
            tion.prepareStatement(insertQuery1);
                PreparedStatement statement2 = connec-
            tion.prepareStatement(insertQuery2)) {

                // Adding the first item
                statement1.setString(1, "Transaction Product 1");
                statement1.setString(2, "Description of Transaction Product 1");
                statement1.setDouble(3, 20.00);
                statement1.setString(4, "Category A");
                statement1.executeUpdate();

                System.out.println("First item added successfully");
                savepoint = connection.setSavepoint("AfterFirstProduct"); // Cre-
                ating a savepoint
            }
        }
    }
}
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        // Trying to add a second product with an error
        statement2.setString(1, "Transaction Product 2");
        statement2.setString(2, "Description of Transaction Product 2");
        statement2.setDouble(3, 25.00);
        statement2.setString(4, "Category B");
        statement2.executeUpdate();

        // If both products are added successfully (NO)
        connection.commit();
        System.out.println("Transaction completed successfully.");

    } catch (Exception e) {
        System.out.println("Error when adding a second product: " +
e.getMessage());
        if (savepoint != null) {
            connection.rollback(savepoint); // Back to savepoint
            System.out.println("Rollback to save point: only the first
item is saved.");
        } else {
            connection.rollback(); // Full back
            System.out.println("Full back: not a single item added.");
        }
    }

    // Output data from the products table
    displayProducts(connection);

} catch (Exception e) {
    System.out.println("Error when working with transactions: " +
e.getMessage());
    e.printStackTrace();
}

}

private static void displayProducts(Connection connection) {
    String query = "SELECT * FROM products";
    try (Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query)) {

        System.out.println("A list of items after a transaction:");
        System.out.println("-----");
        System.out.printf("%-5s %-20s %-30s %-10s %-15s\n", "ID", "Name", "De-
scription", "Price", "Category");
        System.out.println("-----");

        while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String name = resultSet.getString("name");
            String description = resultSet.getString("description");
            double price = resultSet.getDouble("price");
            String category = resultSet.getString("category");

            System.out.printf("%-5d %-20s %-30s %-10.2f %-15s\n", id, name,
description, price, category);
        }

    } catch (Exception e) {
        System.out.println("Error in data output: " + e.getMessage());
    }
}

```

```

}
}

First item added successfully
Error when adding a second product: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax
Rollback to save point: only the first item is saved.
A list of items after a transaction:
-----
ID      Name      Description      Price      Category
-----
122     Product 1    Description of Product 1    10,99     Electronics
123     Product 2    Description of Product 2    12,99     Electronics
124     Product 3    Description of Product 3    5,49      Furniture
125     Product 4    Description of Product 4    8,75      Furniture
126     Product 5    Description of Product 5    3,99      Clothing
127     Product 6    Description of Product 6    7,49      Clothing
128     Product 7    Description of Product 7    11,25     Books
129     Product 8    Description of Product 8    6,89      Books
130     Product 9    Description of Product 9    4,99      Electronics
131     Product 10   Description of Product 10   9,99      Furniture
132     Transaction Product 1 Description of Transaction Product 1 20,00     Category A

```

**Рис.11.4. Результат роботи**

**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.

#### AbstractDAO.java:

```

package TASK_06;
import java.sql.Connection;
import java.util.List;

public abstract class AbstractDAO<T> {
    protected Connection connection;

    public AbstractDAO(Connection connection) {
        this.connection = connection;
    }

    public abstract void create(T entity);

    public abstract T read(int id);

    public abstract List<T> readAll();

    public abstract void update(T entity);

    public abstract void delete(int id);
}

```

#### Main.java:

```

package TASK_06;

```

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.util.List;
import java.util.Properties;
import java.io.FileInputStream;

public class Main {
    private static Connection getConnection() throws Exception {
        Properties props = new Properties();
        try (FileInputStream fis = new FileInputStream("src/db.properties")) {
            props.load(fis);
        }
        String url = props.getProperty("db.url");
        String user = props.getProperty("db.user");
        String password = props.getProperty("db.password");

        return DriverManager.getConnection(url, user, password);
    }

    public static void main(String[] args) {
        try (Connection connection = getConnection()) {
            ProductDAO productDAO = new ProductDAO(connection);

            // 1. Creating a new product
            Product newProduct = new Product("DAO Product", "Description of DAO
Product", 50.00, "Category DAO");
            productDAO.create(newProduct);
            System.out.println("Add product: " + newProduct);

            // 2. Reading the product
            Product fetchedProduct = productDAO.read(newProduct.getId());
            System.out.println("Read product: " + fetchedProduct);

            // 3. Updating the product
            fetchedProduct.setPrice(55.00);
            fetchedProduct.setCategory("Updated Category");
            productDAO.update(fetchedProduct);
            System.out.println("Updated product: " + fetchedProduct);

            // 4. Reading all products
            List<Product> products = productDAO.readAll();
            System.out.println("All product:");
            for (Product product : products) {
                System.out.println(product);
            }

            // 5. Deleting a product
            productDAO.delete(fetchedProduct.getId());
            System.out.println("Product with ID " + fetchedProduct.getId() + " was
deleted.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## Product.java:

```

package TASK_06;

public class Product {
    private int id;

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

private String name;
private String description;
private double price;
private String category;

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

// Builder without ID to create a new product
public Product(String name, String description, double price, String category)
{
    this(0, name, description, price, category);
}

// Getters и 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 getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public double getPrice() {
    return price;
}

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

public String getCategory() {
    return category;
}

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

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

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

## ProductDAO.java:

```
package TASK_06;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductDAO extends AbstractDAO<Product> {

    public ProductDAO(Connection connection) {
        super(connection);
    }
    @Override
    public void create(Product product) {
        String query = "INSERT INTO products (name, description, price, category)
VALUES (?, ?, ?, ?)";
        try (PreparedStatement statement = connection.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS)) {
            statement.setString(1, product.getName());
            statement.setString(2, product.getDescription());
            statement.setDouble(3, product.getPrice());
            statement.setString(4, product.getCategory());
            statement.executeUpdate();

            ResultSet keys = statement.getGeneratedKeys();
            if (keys.next()) {
                product.setId(keys.getInt(1));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    @Override
    public Product read(int id) {
        String query = "SELECT * FROM products WHERE id = ?";
        try (PreparedStatement statement = connection.prepareStatement(query)) {
            statement.setInt(1, id);
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                return new Product(
                    resultSet.getInt("id"),
                    resultSet.getString("name"),
                    resultSet.getString("description"),
                    resultSet.getDouble("price"),
                    resultSet.getString("category")
                );
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }
    @Override
    public List<Product> readAll() {
        List<Product> products = new ArrayList<>();
        String query = "SELECT * FROM products";
        try (Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(query)) {
            while (resultSet.next()) {
                products.add(new Product(
                    resultSet.getInt("id"),
                    resultSet.getString("name"),
                    resultSet.getString("description"),
                    resultSet.getDouble("price"),
                    resultSet.getString("category")
                ));
            }
        }
    }
}
```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        resultSet.getString("category")
    ));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
return products;
}
@Override
public void update(Product product) {
    String query = "UPDATE products SET name = ?, description = ?, price = ?,
category = ? WHERE id = ?";
    try (PreparedStatement statement = connection.prepareStatement(query)) {
        statement.setString(1, product.getName());
        statement.setString(2, product.getDescription());
        statement.setDouble(3, product.getPrice());
        statement.setString(4, product.getCategory());
        statement.setInt(5, product.getId());
        statement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
@Override
public void delete(int id) {
    String query = "DELETE FROM products WHERE id = ?";
    try (PreparedStatement statement = connection.prepareStatement(query)) {
        statement.setInt(1, id);
        statement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}
}

```

```

Add product: Product{id=133, name='DAO Product', description='Description of DAO Product', price=50,00, category='Category DAO'}
Read product: Product{id=133, name='DAO Product', description='Description of DAO Product', price=50,00, category='Category DAO'}
Updated product: Product{id=133, name='DAO Product', description='Description of DAO Product', price=55,00, category='Updated Category'}
All product:
Product{id=122, name='Product 1', description='Description of Product 1', price=10,99, category='Electronics'}
Product{id=123, name='Product 2', description='Description of Product 2', price=12,99, category='Electronics'}
Product{id=124, name='Product 3', description='Description of Product 3', price=5,49, category='Furniture'}
Product{id=125, name='Product 4', description='Description of Product 4', price=8,75, category='Furniture'}
Product{id=126, name='Product 5', description='Description of Product 5', price=3,99, category='Clothing'}
Product{id=127, name='Product 6', description='Description of Product 6', price=7,49, category='Clothing'}
Product{id=128, name='Product 7', description='Description of Product 7', price=11,25, category='Books'}
Product{id=129, name='Product 8', description='Description of Product 8', price=6,89, category='Books'}
Product{id=130, name='Product 9', description='Description of Product 9', price=4,99, category='Electronics'}
Product{id=131, name='Product 10', description='Description of Product 10', price=9,99, category='Furniture'}
Product{id=133, name='DAO Product', description='Description of DAO Product', price=55,00, category='Updated Category'}
Product with ID 133 was deleted.

```

**Рис.11.5. Результат роботи**

**Висновок:** я набув навички створення зв'язку Java програми з базою даних та взаємодії з нею в процесі роботи програми.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				88
Змн.	Арк.	№ докум.	Підпис	Дата		



## Лабораторна робота № 12

**Тема:** Інструменти тестування Junit та Mockito.

**Мета роботи:** Набути навичок модульних та інтеграційних тестів використовуючи Junit та Mockito.

### Хід роботи

**Завдання 2.** Додати до створеного проекту код, який треба покрити тестами. Можете використати код з лабораторних робіт 2 чи 3 або код якогось іншого вашого java проекту.

**Завдання 3.** Тестування за допомогою бібліотеки Junit.

Створіть відповідні тестові класи та напишіть тест-кейси.

Ваші тест-кейси повинні використовувати такі анотації: @BeforeClass, @AfterClass, @Before, @After, @Test, @Ignore.

Ваші тест-кейси повинні використовувати такі методи: fail, assertTrue, assertsEquals, assertNull, assertNotNull, assertSame, assertNotSame.

Перевірте позитивні, негативні випадки та випадки, які генерують виключні ситуації.

Використайте Rules (TemporaryFolder, ExpectedException, TestName, Timeout, ErrorCollector, Verifier).

Використайте групування тестів @Suite.SuiteClasses та параметризацію @Parameters.

**Завдання 4.** Тестування за допомогою фреймворка Mockito.

Створіть відповідні тестові класи та проанотуйте їх анотацією @RunWith(MockitoJUnitRunner.class).

Напишіть тест-кейси. Ваші тест-кейси повинні використовувати такі методи та анотації:

- @Mock, @Spy, @Captor, @InjectMocks або аналогічні їм статичні методи класу Mockito
- when(mock).thenReturn(value), when(mock).thenThrow(),doReturn(value).when(mock).methodName(params) .
- verify() в комбінації з atLeast, atLeastOnce, atMost, times, never

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

### Task.java:

```
public class Task {
    public int[] generateFibonacci(int n) {
        if (n <= 0) throw new IllegalArgumentException("n має бути більше 0");
        int[] fibonacci = new int[n];
        if (n >= 1) fibonacci[0] = 1;
        if (n >= 2) fibonacci[1] = 1;
        for (int i = 2; i < n; i++) {
            fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];
        }
        return fibonacci;
    }
}
```

### Task\_Test.java:

```
import org.junit.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class Task_Test {
    private final Task task = new Task();

    @Test
    public void testGenerateFibonacciWithSmallNumber() {
        int[] result = task.generateFibonacci(5);
        assertThat(result).containsExactly(1, 1, 2, 3, 5);
    }

    @Test
    public void testGenerateFibonacciWithOneElement() {
        int[] result = task.generateFibonacci(1);
        assertThat(result).containsExactly(1);
    }

    @Test
    public void testGenerateFibonacciWithTwoElements() {
        int[] result = task.generateFibonacci(2);
        assertThat(result).containsExactly(1, 1);
    }

    @Test(expected = IllegalArgumentException.class)
    public void testGenerateFibonacciWithInvalidInput() {
        task.generateFibonacci(0);
    }
}
```

### Task\_Test\_Junit.java:

```
import org.junit.*;
import org.junit.rules.*;
import static org.junit.Assert.*;

public class Task_Test_Junit {

    private Task task;

    @Rule
    public TemporaryFolder temporaryFolder = new TemporaryFolder();

    @Rule
    public ExpectedException expectedException = ExpectedException.none();

    @Rule
```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public TestName testName = new TestName();

@Rule
public Timeout timeout = Timeout.millis(1000);

@BeforeClass
public static void setUpBeforeClass() {
    System.out.println("BeforeClass: Performed before running all tests.");
}

@AfterClass
public static void tearDownAfterClass() {
    System.out.println("AfterClass: Performed after running all tests.");
}

@Before
public void setUp() {
    System.out.println("Before: Initialisation before each test.");
    task = new Task();
}

@After
public void tearDown() {
    System.out.println("After: Cleaning after each test.");
}

@Test
public void testGenerateFibonacciPositive() {
    System.out.println("Тест: " + testName.getMethodName());
    int[] result = task.generateFibonacci(5);
    assertEquals(new int[]{1, 1, 2, 3, 5}, result);
}

@Test(expected = IllegalArgumentException.class)
public void testGenerateFibonacciWithZero() {
    System.out.println("Тест: " + testName.getMethodName());
    task.generateFibonacci(0);
}

@Test
public void testGenerateFibonacciWithNull() {
    System.out.println("Тест: " + testName.getMethodName());
    int[] result = task.generateFibonacci(1);
    assertNotNull(result);
    assertEquals(new int[]{1}, result);
}

@Ignore("This test is skipped because it is not complete.")
@Test
public void ignoredTest() {
    fail("This test should not be performed.");
}
}

```

### Task\_Test\_Junit\_Parameterized.java:

```

import org.junit.*;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import java.util.Arrays;
import java.util.Collection;

import static org.junit.Assert.assertEquals;

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@RunWith(Parameterized.class)
public class Task_Test_Junit_Parameterized {

    private final int input;
    private final int[] expected;
    private Task task;

    public Task_Test_Junit_Parameterized(int input, int[] expected) {
        this.input = input;
        this.expected = expected;
    }

    @Parameterized.Parameters
    public static Collection<Object[]> data() {
        return Arrays.asList(new Object[][]{
            {1, new int[]{1}},
            {2, new int[]{1, 1}},
            {5, new int[]{1, 1, 2, 3, 5}}
        });
    }

    @Before
    public void setUp() {
        task = new Task();
    }

    @Test
    public void testGenerateFibonacci() {
        assertEquals(expected, task.generateFibonacci(input));
    }
}

```

### Task\_Test\_Mockito.java:

```

import org.junit.*;
import org.junit.runner.RunWith;
import org.mockito.*;
import org.mockito.junit.MockitoJUnitRunner;

import static org.mockito.Mockito.*;
import static org.junit.Assert.*;

@RunWith(MockitoJUnitRunner.class)
public class Task_Test_Mockito {

    @Mock
    private Task task;

    @InjectMocks
    private Task_Test_Mockito_Service taskService;

    @Captor
    private ArgumentCaptor<Integer> argumentCaptor;

    @Before
    public void setUp() {
        MockitoAnnotations.initMocks(this);
    }

    @Test
    public void testGenerateFibonacciWithMock() {
        when(task.generateFibonacci(5)).thenReturn(new int[]{1, 1, 2, 3, 5});
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        int[] result = task.generateFibonacci(5);

        assertEquals(new int[]{1, 1, 2, 3, 5}, result);

        verify(task, times(1)).generateFibonacci(5);
    }

    @Test
    public void testGenerateFibonacciWithException() {
        when(task.generateFibonacci(0)).thenThrow(new IllegalArgumentException("n
must be greater than 0"));
        IllegalArgumentException exception = assertThrows(
            IllegalArgumentException.class,
            () -> task.generateFibonacci(0)
        );

        assertEquals("n must be greater than 0", exception.getMessage());
        verify(task, times(1)).generateFibonacci(0);
    }

    @Test
    public void testDoReturnMock() {
        doReturn(new int[]{1, 1, 2, 3, 5}).when(task).generateFibonacci(5);

        int[] result = task.generateFibonacci(5);

        assertEquals(new int[]{1, 1, 2, 3, 5}, result);

        verify(task, times(1)).generateFibonacci(5);
    }

    @Test
    public void testVerifyNeverCalled() {
        verify(task, never()).generateFibonacci(10);
    }

    @Test
    public void testCaptor() {
        when(task.generateFibonacci(anyInt())).thenReturn(new int[]{1, 1, 2, 3,
5});
        task.generateFibonacci(5);
        verify(task).generateFibonacci(argumentCaptor.capture());
        assertEquals(Integer.valueOf(5), argumentCaptor.getValue());
    }
}

```

### Task\_Test\_Mockito\_Service.java:

```

public class Task_Test_Mockito_Service {
    private final Task task;

    public Task_Test_Mockito_Service(Task task) {
        this.task = task;
    }

    public int[] processFibonacci(int n) {
        return task.generateFibonacci(n);
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

### AllTests.java:

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({
    Task_Test.class,
    Task_Test_Junit.class,
    Task_Test_Junit_Parameterized.class,
    Task_Test_Mockito.class
})
public class AllTests {
}
```

✓ AllTests	234 ms	✓ Tests passed: 15, ignored: 1 of 16 tests – 234 ms
> ✓ Task_Test	39 ms	"C:\Program Files\Java\jdk-23\bin\java.exe" ...
> ✓ Task_Test_Junit	38 ms	BeforeClass: Performed before running all tests.
✓ Task_Test_Junit_Parameterized	3 ms	Before: Initialisation before each test.
> ✓ [0]	3 ms	Тест: testGenerateFibonacciPositive
> ✓ [1]	0 ms	After: Cleaning after each test.
✓ [2]	0 ms	
✓ testGenerateFibonacci[2]	0 ms	This test is skipped because it is not complete.
✓ Task_Test_Mockito	154 ms	Before: Initialisation before each test.
✓ testDoReturnMock	148 ms	Тест: testGenerateFibonacciWithNull
✓ testVerifyNeverCalled	1 ms	After: Cleaning after each test.
✓ testGenerateFibonacciWithException	3 ms	Before: Initialisation before each test.
✓ testCaptor	2 ms	Тест: testGenerateFibonacciWithZero
✓ testGenerateFibonacciWithMock	0 ms	After: Cleaning after each test.
		AfterClass: Performed after running all tests.

Рис.12.1. Результат тестування

**Висновок:** я набув навичок модульних та інтеграційних тестів, використовуючи Junit та Mockito.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

## Лабораторна робота № 13

**Тема:** Інструменти збірки.

**Мета роботи:** Набути навичок роботи з Maven та Gradle.

### Хід роботи

**Завдання 2.** Створити maven Java проект maven\_java\_lab\_13:

- Додайте до створеного проекту код, тести до нього на власний вибір (можете використати з лабораторної роботи №12) та необхідні залежності в pom.xml.
- Прописати в pom.xml groupId - ваше прізвище, artifactId - com.education.ztu, version – вашу версію, packaging – jar (якщо звичайний проект) або war (якщо веб проект, наприклад як в лабораторній №14)
- Додати такі плагіни як CheckStyle – відповідність правилам написання коду, FindBugs – пошук багів, PMD – для аналізу коду, JaCoCo – перевірка покриття тестами. Конфігурувати плагіни CheckStyle, FindBugs, PMD на виконання в фазі verify, а JaCoCo в фазі test.
- Для генерації звітів у html додати та налаштувати плагіни maven-site plugin та maven-project-info-reports-plugin.
- Виконати команди mvn test та mvn verify.
- Запустити на виконання mvn clean site. Переглянути звіти плагінів. У разі генерації помилок плагінами виправити їх та знову запустити mvn clean site.
- Запустити mvn clean install. Перевірити чи створився jar або war файл (в залежності від вашого проекту) в директорії target.

#### Task.java:

```
public class Task {  
    public int[] generateFibonacci(int n) {  
        if (n <= 0) throw new IllegalArgumentException("n має бути більше 0");  
        int[] fibonacci = new int[n];  
        if (n >= 1) fibonacci[0] = 1;  
        if (n >= 2) fibonacci[1] = 1;  
        for (int i = 2; i < n; i++) {  
            fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];  
        }  
        return fibonacci;  
    }  
}
```

#### Task\_Test.java:

```
import org.junit.Test;  
import static org.assertj.core.api.Assertions.assertThat;
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public class Task_Test {
    private final Task task = new Task();

    @Test
    public void testGenerateFibonacciWithSmallNumber() {
        int[] result = task.generateFibonacci(5);
        assertThat(result).containsExactly(1, 1, 2, 3, 5);
    }

    @Test
    public void testGenerateFibonacciWithOneElement() {
        int[] result = task.generateFibonacci(1);
        assertThat(result).containsExactly(1);
    }

    @Test
    public void testGenerateFibonacciWithTwoElements() {
        int[] result = task.generateFibonacci(2);
        assertThat(result).containsExactly(1, 1);
    }

    @Test(expected = IllegalArgumentException.class)
    public void testGenerateFibonacciWithInvalidInput() {
        task.generateFibonacci(0);
    }
}

```

### Task\_Test\_Junit.java:

```

import org.junit.*;
import org.junit.rules.*;
import static org.junit.Assert.*;

public class Task_Test_Junit {

    private Task task;

    @Rule
    public TemporaryFolder temporaryFolder = new TemporaryFolder();

    @Rule
    public ExpectedException expectedException = ExpectedException.none();

    @Rule
    public TestName testName = new TestName();

    @Rule
    public Timeout timeout = Timeout.millis(1000);

    @BeforeClass
    public static void setUpBeforeClass() {
        System.out.println("BeforeClass: Performed before running all tests.");
    }

    @AfterClass
    public static void tearDownAfterClass() {
        System.out.println("AfterClass: Performed after running all tests.");
    }

    @Before
    public void setUp() {
        System.out.println("Before: Initialisation before each test.");
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				96
Змн.	Арк.	№ докум.	Підпис	Дата		



```

        task = new Task();
    }

    @After
    public void tearDown() {
        System.out.println("After: Cleaning after each test.");
    }

    @Test
    public void testGenerateFibonacciPositive() {
        System.out.println("Тест: " + testName.getMethodName());
        int[] result = task.generateFibonacci(5);
        assertEquals(new int[]{1, 1, 2, 3, 5}, result);
    }

    @Test(expected = IllegalArgumentException.class)
    public void testGenerateFibonacciWithZero() {
        System.out.println("Тест: " + testName.getMethodName());
        task.generateFibonacci(0);
    }

    @Test
    public void testGenerateFibonacciWithNull() {
        System.out.println("Тест: " + testName.getMethodName());
        int[] result = task.generateFibonacci(1);
        assertNotNull(result);
        assertEquals(new int[]{1}, result);
    }

    @Ignore("This test is skipped because it is not complete.")
    @Test
    public void ignoredTest() {
        fail("This test should not be performed.");
    }
}

```

### Task\_Test\_Junit\_Parameterized.java:

```

import org.junit.*;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import java.util.Arrays;
import java.util.Collection;

import static org.junit.Assert.assertEquals;

@RunWith(Parameterized.class)
public class Task_Test_Junit_Parameterized {

    private final int input;
    private final int[] expected;
    private Task task;

    public Task_Test_Junit_Parameterized(int input, int[] expected) {
        this.input = input;
        this.expected = expected;
    }

    @Parameterized.Parameters
    public static Collection<Object[]> data() {
        return Arrays.asList(new Object[][]{
            {1, new int[]{1}},
            {2, new int[]{1, 1}},
        });
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {5, new int[]{1, 1, 2, 3, 5}}
    });
}

@Before
public void setUp() {
    task = new Task();
}

@Test
public void testGenerateFibonacci() {
    assertEquals(expected, task.generateFibonacci(input));
}
}

```

### Task\_Test\_Mockito.java:

```

import org.junit.*;
import org.junit.runner.RunWith;
import org.mockito.*;
import org.mockito.junit.MockitoJUnitRunner;

import static org.mockito.Mockito.*;
import static org.junit.Assert.*;

@RunWith(MockitoJUnitRunner.class)
public class Task_Test_Mockito {

    @Mock
    private Task task;

    @InjectMocks
    private Task_Test_Mockito_Service taskService;

    @Captor
    private ArgumentCaptor<Integer> argumentCaptor;

    @Before
    public void setUp() {
        MockitoAnnotations.initMocks(this);
    }

    @Test
    public void testGenerateFibonacciWithMock() {
        when(task.generateFibonacci(5)).thenReturn(new int[]{1, 1, 2, 3, 5});

        int[] result = task.generateFibonacci(5);

        assertEquals(new int[]{1, 1, 2, 3, 5}, result);

        verify(task, times(1)).generateFibonacci(5);
    }

    @Test
    public void testGenerateFibonacciWithException() {
        when(task.generateFibonacci(0)).thenThrow(new IllegalArgumentException("n
must be greater than 0"));
        IllegalArgumentException exception = assertThrows(
            IllegalArgumentException.class,
            () -> task.generateFibonacci(0)
        );

        assertEquals("n must be greater than 0", exception.getMessage());
    }
}

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        verify(task, times(1)).generateFibonacci(0);
    }

    @Test
    public void testDoReturnMock() {
        doReturn(new int[]{1, 1, 2, 3, 5}).when(task).generateFibonacci(5);

        int[] result = task.generateFibonacci(5);

        assertEquals(new int[]{1, 1, 2, 3, 5}, result);

        verify(task, times(1)).generateFibonacci(5);
    }

    @Test
    public void testVerifyNeverCalled() {
        verify(task, never()).generateFibonacci(10);
    }

    @Test
    public void testCaptor() {
        when(task.generateFibonacci(anyInt())).thenReturn(new int[]{1, 1, 2, 3,
5});
        task.generateFibonacci(5);
        verify(task).generateFibonacci(argumentCaptor.capture());
        assertEquals(Integer.valueOf(5), argumentCaptor.getValue());
    }
}

```

### Task\_Test\_Mockito\_Service.java:

```

public class Task_Test_Mockito_Service {
    private final Task task;

    public Task_Test_Mockito_Service(Task task) {
        this.task = task;
    }

    public int[] processFibonacci(int n) {
        return task.generateFibonacci(n);
    }
}

```

### AllTests.java:

```

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({
    Task_Test.class,
    Task_Test_Junit.class,
    Task_Test_Junit_Parameterized.class,
    Task_Test_Mockito.class
})
public class AllTests {
}

```

### pom.xml:

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>Leus</groupId>
    <artifactId>Java-LAB-13</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>

    <name>Java-LAB-13</name>
    <description>
        This project demonstrates a Java-based application for generating Fibonacci sequences
        with integrated testing and quality assurance tools. Includes Checkstyle,
        PMD, FindBugs,
        and JaCoCo for comprehensive code analysis and reporting.
    </description>
    <url>https://github.com/VadymLeus/Y3S1-Java</url>

    <licenses>
        <license>
            <name>Apache License, Version 2.0</name>
            <url>http://www.apache.org/licenses/LICENSE-2.0</url>
            <distribution>repo</distribution>
        </license>
    </licenses>

    <developers>
        <developer>
            <id>Leus</id>
            <name>Vadym Leus</name>
            <email>vadym.leus@example.com</email>
            <roles>
                <role>developer</role>
            </roles>
        </developer>
    </developers>

    <dependencies>
        <!-- JUnit 4 -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>

        <!-- Mockito -->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>4.6.1</version>
            <scope>test</scope>
        </dependency>

        <!-- AssertJ -->
        <dependency>
            <groupId>org.assertj</groupId>
            <artifactId>assertj-core</artifactId>
            <version>3.11.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<build>
  <plugins>
    <!-- Checkstyle -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-checkstyle-plugin</artifactId>
      <version>3.2.2</version>
      <executions>
        <execution>
          <phase>verify</phase>
          <goals>
            <goal>check</goal>
          </goals>
        </execution>
      </executions>
    </plugin>

    <!-- FindBugs -->
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>findbugs-maven-plugin</artifactId>
      <version>3.0.5</version>
      <executions>
        <execution>
          <phase>verify</phase>
          <goals>
            <goal>check</goal>
          </goals>
        </execution>
      </executions>
    </plugin>

    <!-- PMD -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-pmd-plugin</artifactId>
      <version>3.15.0</version>
      <executions>
        <execution>
          <phase>verify</phase>
          <goals>
            <goal>check</goal>
          </goals>
        </execution>
      </executions>
    </plugin>

    <!-- JaCoCo -->
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>0.8.10</version>
      <executions>
        <execution>
          <id>prepare-agent</id>
          <phase>test</phase>
          <goals>
            <goal>prepare-agent</goal>
          </goals>
        </execution>
        <execution>
          <id>generate-report</id>
          <phase>verify</phase>

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <goals>
            <goal>report</goal>
        </goals>
    </execution>
</executions>
</plugin>

<!-- Maven Site Plugin -->
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-site-plugin</artifactId>
    <version>3.12.0</version>
    <executions>
        <execution>
            <phase>site</phase>
            <goals>
                <goal>site</goal>
            </goals>
        </execution>
    </executions>
</plugin>

<!-- Maven Project Info Reports Plugin -->
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-project-info-reports-plugin</artifactId>
    <version>3.4.1</version>
</plugin>
</plugins>
</build>
</project>

```

✓ AllTests	264 ms	✓ Tests passed: 15, ignored: 1 of 16 tests – 264 ms
✓ Task_Test	62 ms	C:\Users\vadym\jdk\corretto-1.8.0_432\bin\java.exe ...
✓ testGenerateFibonacciWithOneElem	61 ms	BeforeClass: Performed before running all tests.
✓ testGenerateFibonacciWithInvalidInput	1 ms	Before: Initialisation before each test.
✓ testGenerateFibonacciWithTwoElements	0 ms	Test: testGenerateFibonacciPositive
✓ testGenerateFibonacciWithSmallNumbers	0 ms	After: Cleaning after each test.
> ✓ Task_Test_Junit	44 ms	This test is skipped because it is not complete.
✓ Task_Test_Junit_Parameterized	0 ms	Before: Initialisation before each test.
> ✓ [0]	0 ms	Test: testGenerateFibonacciWithNull
> ✓ [1]	0 ms	After: Cleaning after each test.
✓ [2]	0 ms	Before: Initialisation before each test.
✓ testGenerateFibonacci[2]	0 ms	Test: testGenerateFibonacciWithZero
✓ Task_Test_Mockito	158 ms	After: Cleaning after each test.
✓ testDoReturnMock	151 ms	AfterClass: Performed after running all tests.
✓ testVerifyNeverCalled	1 ms	Process finished with exit code 0
✓ testGenerateFibonacciWithException	3 ms	
✓ testCaptor	2 ms	
✓ testGenerateFibonacciWithMock	1 ms	

Рис.13.1. Результат тестування

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

[INFO] Generating "Licenses" report --- maven-project-info-reports-plugin:3.4.1:licenses
[INFO] Generating "Plugin Management" report --- maven-project-info-reports-plugin:3.4.1:plugin-management
[INFO] Generating "Plugins" report --- maven-project-info-reports-plugin:3.4.1:plugins
[INFO] Generating "Summary" report --- maven-project-info-reports-plugin:3.4.1:summary
[INFO] Generating "Team" report --- maven-project-info-reports-plugin:3.4.1:team
[INFO] --- site:3.12.0:site (default) @ Java-LAB-13 ---
[WARNING] Input file encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] configuring report plugin org.apache.maven.plugins:maven-project-info-reports-plugin:3.4.1
[INFO] 15 reports detected for maven-project-info-reports-plugin:3.4.1: ci-management, dependencies, dependency-info,
    plugin-management, plugins, scm, summary, team
[INFO] Rendering site with default locale English (en)
[INFO] Relativizing decoration links with respect to localized project URL: https://github.com/VadymLeus/Y3S1-Java
[INFO] Rendering content with org.apache.maven.skins:maven-default-skin:jar:1.3 skin.
[INFO] Generating "Dependencies" report --- maven-project-info-reports-plugin:3.4.1:dependencies
[INFO] Generating "Dependency Information" report --- maven-project-info-reports-plugin:3.4.1:dependency-info
[INFO] Generating "About" report --- maven-project-info-reports-plugin:3.4.1:index
[INFO] Generating "Licenses" report --- maven-project-info-reports-plugin:3.4.1:licenses
[INFO] Generating "Plugin Management" report --- maven-project-info-reports-plugin:3.4.1:plugin-management
[INFO] Generating "Plugins" report --- maven-project-info-reports-plugin:3.4.1:plugins
[INFO] Generating "Summary" report --- maven-project-info-reports-plugin:3.4.1:summary
[INFO] Generating "Team" report --- maven-project-info-reports-plugin:3.4.1:team
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.845 s
[INFO] Finished at: 2024-12-10T14:35:38+02:00
[INFO] -----

```

**Рис.13.2. Виконання команди mvn clean site**

**Завдання 3.** Створити gradle Java проект gradle\_java\_lab\_13 з пакетом com.education.ztu

- Створити конфігурацію аналогічну до завдання 2 у файлі build.gradle.
- Для перегляду всіх доступних тасок виконати команду gradle tasks --all.
- Запустити таски доданих плагінів, виправити помилки.
- Для створення jar або war файлу виконати команду gradle build. Перевірити чи він був створений в gradle/wrapper.
- Написати власний task використовуючи groovy (логіка за власним бажанням).

Запустити його для перевірки.

```

Build.gradle:
plugins {
    id 'java'
    id 'jacoco'
    id 'checkstyle'
}

group = 'Leus'
version = '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    testImplementation 'junit:junit:4.13.2'
    testImplementation 'org.mockito:mockito-core:4.6.1'
    testImplementation 'org.assertj:assertj-core:3.11.1'
}

checkstyle {
    toolVersion = '8.45'
}

```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				103
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

jar {
    manifest {
        attributes 'Main-Class': 'com.education.ztu.Task'
    }
}

task printHello {
    doLast {
        println 'Hello, Gradle!'
    }
}

checkstyle {
    toolVersion = "10.12.0"
    configFile = file('gradle/config/checkstyle/checkstyle.xml')
}

tasks.withType(Checkstyle).configureEach {
    enabled = false
}

settings.gradle:
rootProject.name = 'gradle-Java-LAB-13'

```

```

~\Desktop\Y3S1-Java\Java-LAB-13\Gradle git:[main]
gradle build
Starting a Gradle Daemon, 1 incompatible and 1 stopped Daemons could not be reu

> Task :compileTestJava
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
[Incubating] Problems report is available at: file:///C:/Users/vadym/Desktop/Y3S1-Java\Java-LAB-13\Gradle\reports\problems-report.html

BUILD SUCCESSFUL in 14s
4 actionable tasks: 4 executed

```

**Рис.13.3. Виконання команди gradle build**

```

~\Desktop\Y3S1-Java\Java-LAB-13\Gradle git:[main]
gradle test

BUILD SUCCESSFUL in 535ms
3 actionable tasks: 3 up-to-date

```

**Рис.13.4. Виконання команди gradle test**

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				104
Змн.	Арк.	№ докум.	Підпис	Дата		



```
processTestResources - Processes test resources.

Rules
-----
Pattern: clean<TaskName>: Cleans the output files of a task.
Pattern: build<ConfigurationName>: Assembles the artifacts of a configuration.

BUILD SUCCESSFUL in 603ms
1 actionable task: 1 executed
```

**Рис.13.5. Виконання команди gradle tasks --all**

**Висновок:** я набув навичок роботи з Maven та Gradle.

## Лабораторна робота № 14

**Тема:** Spring.

**Мета роботи:** Практика роботи з Spring Framework.

### Хід роботи

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

**Завдання 3.** Робота з Spring WebMVC:

- Створити новий контролер для можливості видалити та редагувати існуючі об'єкти класу TodoItem.
- Для мапінга використати @Controller, @PostMapping, @DeleteMapping анотації.
- Отримати параметри запиту з використання анотацій @PathParam та @ModelAttribute
- Реалізувати переадресацію та перенаправлення запиту.

- Реалізувати обробку помилок за допомогою @ExceptionHandler.

**Завдання 4.** Робота з Thymeleaf:

- Створити сторінку для редагування об'єкту класу TodoItem.
- Редагувати існуючу сторінку index.html та додати посилання на сторінки редагування та видалення.

**Завдання 5.** Робота з Spring Data JDBC:

- Реалізувати логіку видалення та оновлення статусу для об'єкту класу TodoItem.

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

- Оновити існуючий TodoService та додати до нього логіку для видалення та оновлення TodoItems та використати ці методи в створеному раніше контролері.

#### pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.13</version>
    <relativePath/> <!-- lookup parent from repository -->
```

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				106
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</parent>
<groupId>com.education.ztu</groupId>
<artifactId>spring</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>spring</name>
<description>Demo project for Spring Boot</description>
<properties>
    <java.version>17</java.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jdbc</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>jakarta.validation</groupId>
        <artifactId>jakarta.validation-api</artifactId>
        <version>3.0.2</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.flywaydb</groupId>
        <artifactId>flyway-core</artifactId>
    </dependency>
    <dependency>
        <groupId>org.flywaydb</groupId>
        <artifactId>flyway-mysql</artifactId>
    </dependency>

    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>

```

**TodoConfiguration.java:**

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				107
Змн.	Арк.	№ докум.	Підпис	Дата		

```

package com.education.ztu.spring.configuration;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;

@Configuration
@ConfigurationProperties
public class TodoConfiguration {

    private String title;

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
}

```

### MainPageController.java:

```

package com.education.ztu.spring.controller;

import com.education.ztu.spring.configuration.TodoConfiguration;
import com.education.ztu.spring.entity.TODOItem;
import com.education.ztu.spring.respository.TODOItemRepository;
import com.education.ztu.spring.service.TODOService;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

@Controller("/")
public class MainPageController {

    private final TODOService todoService;
    private final TodoConfiguration todoConfiguration;

    public MainPageController(TODOService todoService,
                             TodoConfiguration todoConfiguration) {
        this.todoService = todoService;
        this.todoConfiguration = todoConfiguration;
    }

    @GetMapping
    public String getMainPage(Model model) {
        model.addAttribute("todo", new TODOItem());
        model.addAttribute("title", todoConfiguration.getTitle());
        model.addAttribute("todos", todoService.getAllTODOItems());
        return "index.html";
    }

    @PostMapping
    public String todoItemSubmit(@ModelAttribute TODOItem item, Model model) {
        todoService.saveTODOItem(item);
        return "redirect:/";
    }
}

```

		Левченко В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				108
Змн.	Арк.	№ докум.	Підпис	Дата		

## TodoItemController.java:

```
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.*;

import jakarta.validation.constraints.NotNull;

@Controller
@RequestMapping("/todo")
public class TodoItemController {

    private final TODOService todoService;

    public TodoItemController(TODOService todoService) {
        this.todoService = todoService;
    }

    @GetMapping("/edit/{id}")
    public String showEditForm(@PathVariable("id") Long id, Model model) {
        TODOItem todoItem = todoService.get_TODOItemById(id);
        model.addAttribute("todoItem", todoItem);
        return "edit";
    }

    @PostMapping("/edit")
    public String edit_TODOItem(@ModelAttribute TODOItem item) {
        todoService.save_TODOItem(item);
        return "redirect:/";
    }

    @PostMapping("/delete/{id}")
    public String delete_TODOItem(@PathVariable("id") @NotNull Long id) {
        todoService.delete_TODOItemById(id);
        return "redirect:/";
    }
}
```

## TODOItem.java:

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

import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapping.Column;
import org.springframework.data.relational.core.mapping.Table;

@Table("todo_items")
public class TODOItem {

    @Id
    private Long id;

    @Column(value = "name")
    private String data;

    @Column(value = "is_completed")
    private Boolean isCompleted;

    public Long getId() {
        return id;
    }
}
```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				109
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getData() {
        return data;
    }

    public void setData(String data) {
        this.data = data;
    }

    public Boolean getCompleted() {
        return isCompleted;
    }

    public void setCompleted(Boolean completed) {
        isCompleted = completed;
    }
}

```

### TodoItemRepository.java:

```

package com.education.ztu.spring.respository;

import com.education.ztu.spring.entity.TODOItem;
import org.springframework.data.jdbc.repository.query.Modifying;
import org.springframework.data.jdbc.repository.query.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;

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

    @Query("SELECT * FROM todo_items WHERE is_completed = :is_completed")
    List<TODOItem> findAllWhereCompleted(@Param("is_completed") boolean isCompleted);

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

```

### TODOService.java:

```

package com.education.ztu.spring.service;

import com.education.ztu.spring.entity.TODOItem;
import com.education.ztu.spring.respository.TODOItemRepository;
import org.springframework.stereotype.Component;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.StreamSupport;

@Component
public class TODOService {

    private final TODOItemRepository todoItemRepository;

    public TODOService(TODOItemRepository todoItemRepository) {

```

		Левс В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				110
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        this.todoItemRepository = todoItemRepository;
    }
    public TodoItem getTodoItemById(Long id) {
        return todoItemRepository.findById(id).orElseThrow(() -> new RuntimeException("Todo item not found"));
    }
    public List<TodoItem> getAllTodoItems() {
        Iterable<TodoItem> itemsIterator = todoItemRepository.findAll();
        return StreamSupport.stream(itemsIterator.spliterator(), false)
            .collect(Collectors.toList());
    }
    public void saveTodoItem(TodoItem todoItem) {
        todoItemRepository.save(todoItem);
    }
    public void deleteTodoItemById(Long id) {
        todoItemRepository.deleteById(id);
    }
}

```

### Application.java:

```

package com.education.ztu.spring;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.context.properties.EnableConfigurationProperties;

@EnableConfigurationProperties
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}

```

### ApplicationTests.java:

```

package com.education.ztu.spring;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class ApplicationTests {

    @Test
    void contextLoads() {
    }

}

```

```

2024-12-10T14:55:48.717+02:00 INFO 8372 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.16]
2024-12-10T14:55:48.769+02:00 INFO 8372 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-12-10T14:55:48.770+02:00 INFO 8372 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 619 ms
2024-12-10T14:55:48.865+02:00 INFO 8372 --- [main] o.f.c.internal.license.VersionPrinter : Flyway Community Edition 9.5.1 by Redgate
2024-12-10T14:55:48.865+02:00 INFO 8372 --- [main] o.f.c.internal.license.VersionPrinter : See what's new here: https://flywaydb.org/documentation/learnmore/re
2024-12-10T14:55:48.870+02:00 INFO 8372 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-12-10T14:55:49.087+02:00 INFO 8372 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@15a
2024-12-10T14:55:49.088+02:00 INFO 8372 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-12-10T14:55:49.106+02:00 INFO 8372 --- [main] o.f.c.i.database.base.BaseDatabaseType : Database: jdbc:mysql://localhost:3306/todo (MySQL 8.0)
2024-12-10T14:55:49.133+02:00 INFO 8372 --- [main] o.f.core.internal.command.DbValidate : Successfully validated 1 migration (execution time 00:00.012s)
2024-12-10T14:55:49.140+02:00 INFO 8372 --- [main] o.f.core.internal.command.DbMigrate : Current version of schema 'todo': 1
2024-12-10T14:55:49.141+02:00 INFO 8372 --- [main] o.f.core.internal.command.DbMigrate : Schema 'todo' is up to date. No migration necessary.
2024-12-10T14:55:49.381+02:00 INFO 8372 --- [main] o.s.b.a.w.s.WelcomePageHandlerMapping : Adding welcome page template: index
2024-12-10T14:55:49.490+02:00 INFO 8372 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2024-12-10T14:55:49.493+02:00 INFO 8372 --- [main] com.education.ztu.spring.Application : Started Application in 1.573 seconds (process running for 1.769)

```

Рис.14.1. Виконання команди `mvn spring-boot:run`

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				111
Змн.	Арк.	№ докум.	Підпис	Дата		

**TODO List**

☐ Do somethings Edit Delete

☐ Dont do somethings Edit Delete

Add

**Рис.14.2. Додавання нових завдань**

### Edit Task

Task

Completed ☒

Save Cancel

**Рис.14.3. Зміна завдань**

**TODO List**

☒ Do somethings Edit Delete

☐ Dont do somethings Edit Delete

Add

**Рис.14.4. Результати змін**

**Висновок:** я попрактикувався роботи з Spring Framework.

		Леус В.О.			ДУ «Житомирська політехніка».24.121.15.000 – Звіт	Арк.
		Піонтківський В.І.				112
Змн.	Арк.	№ докум.	Підпис	Дата		