Лабораторна робота №1

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

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

Завдання на лабораторну роботу

Завдання 4. Написання простих програм:

Програма 1

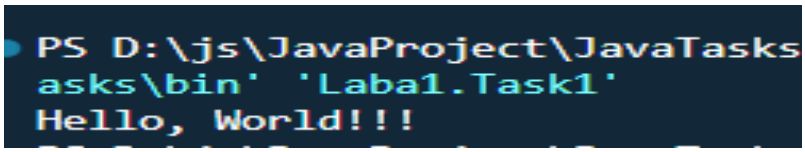
Ім'я класу: com.education.ztu.Task1

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

Вигляд коду:

```
package Laba1;

public class Task1 {
    public static void main(String[] args) {
        System.out.println("Hello, World!!!");
    }
}
```

Результат виконання програми:



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.

Вигляд коду:

```java
package Laba1;
import java.util.Scanner;

public class Task2 {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print(s:"Enter first number:");
        int Number1 = scanner.nextInt();

        System.out.print(s:"Enter second number:");
        int Number2 = scanner.nextInt();

        int sum = Number1 + Number2;

        System.out.println("Total: " + sum);

        scanner.close();
    }
}
```

```
PROBLEMS 84  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

PS D:\js\JavaProject\JavaTasks>  & 'C:\Program Files\Microsoft\jdk-11.0.16
asks\bin' 'Laba1.Task2'
Enter first number:45
Enter second number:78
Total: 123
PS D:\js\JavaProject\JavaTasks>
```

Програма 4

Ім'я класу: com.education.ztu.Task4

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

Вигляд коду:

```java
public class Task4 {
    private int num1;
    private int num2;
```

```
    public Task4() {}

    public void readInput() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first positive integer: ");
        num1 = scanner.nextInt();
        System.out.print("Enter the second positive integer: ");
        num2 = scanner.nextInt();
        scanner.close();
    }

    public int calculateGCD() {
        if (num1 == 0) {
            return num2;
        } else if (num2 == 0) {
            return num1;
        } else {
            while (num2 != 0) {
                int temp = num2;
                num2 = num1 % num2;
                num1 = temp;
            }
            return num1;
        }
    }

    public static void main(String[] args) {
        Task4 gcd = new Task4();
        gcd.readInput();
        int result = gcd.calculateGCD();
        System.out.println("The greatest common divisor is: " + result);
    }
}
```

Результат виконання програми:

```
Enter the first positive integer: 56
Enter the second positive integer: 34
The greatest common divisor is: 2
```

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.

Вигляд коду:

```java
public class Task5 {
  private int number;

    public Task5() {}

    public void readInput() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        number = scanner.nextInt();
        scanner.close();
    }

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

    public static void main(String[] args) {
        Task5 sumOfDigits = new Task5();
        sumOfDigits.readInput();
        int result = sumOfDigits.calculateSumOfDigits();
        System.out.println("The sum of digits is: " + result);
    }
}
```

Результат виконання програми:

```
PS D:\js\JavaProject\JavaTasks>
Enter a positive integer: 56
The sum of digits is: 11
```

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.

Вигляд коду:

```
public class Task6 {
  public int[] generateFibonacci(int n) {
        int[] fibonacciArray = new int[n];
        if (n > 0) fibonacciArray[0] = 1;
        if (n > 1) fibonacciArray[1] = 1;


        for (int i = 2; i < n; i++) {
            fibonacciArray[i] = fibonacciArray[i - 1] + fibonacciArray[i - 2];
        }

        return fibonacciArray;
    }

    public int[] reverseArray(int[] array) {
        int n = array.length;
        int[] reversedArray = new int[n];

        for (int i = 0; i < n; i++) {
            reversedArray[i] = array[n - 1 - i];
        }

        return reversedArray;
    }

    public void printArray(int[] array) {
        for (int i : array) {
            System.out.print(i + " ");
        }
        System.out.println();
    }




    public void run() {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Введіть кількість елементів у послідовності
Фібоначчі: ");
            int n = scanner.nextInt();

            if (n <= 0) {
```

```
                System.out.println("Кількість елементів має бути додатною!");
                return;
            }

            int[] fibonacciArray = generateFibonacci(n);

            int[] reversedFibonacciArray = reverseArray(fibonacciArray);

            System.out.println("Послідовність Фібоначчі:");
            printArray(fibonacciArray);

            System.out.println("Зворотна послідовність Фібоначчі:");
            printArray(reversedFibonacciArray);
        } catch (Exception e) {
            System.out.println("Помилка вводу! Введіть коректне ціле число.");
        } finally {
            scanner.close();
        }
    }

    public static void main(String[] args) {
        Task6 fibonacciArray = new Task6();
        fibonacciArray.run();
    }
}
```

Результат виконання програми:

```
Введ?ть к?льк?сть елемент?в у посл?довност? Ф?боначч?: 7
Посл?довн?сть Ф?боначч?:
1 1 2 3 5 8 13
Зворотна посл?довн?сть Ф?боначч?:
13 8 5 3 2 1 1
```

Програма 7

Ім'я класу: com.education.ztu.Task7

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

A ==> 65

B ==> 66

C ==> 67

Вигляд коду:

```
public class Task7 {
  public void displayAlphabetWithCodes() {
    char[] alphabet = new char[26];

    for (int i = 0; i < 26; i++) {
        alphabet[i] = (char) ('A' + i);
    }

    for (char letter : alphabet) {
        int code = (int) letter;
        System.out.println(letter + " ==> " + code);
    }
}

public static void main(String[] args) {
    Task7 task = new Task7();
    task.displayAlphabetWithCodes();
}
}
```

Результат виконання програми:

```
A ==> 65
B ==> 66
C ==> 67
D ==> 68
E ==> 69
F ==> 70
G ==> 71
H ==> 72
I ==> 73
J ==> 74
K ==> 75
L ==> 76
M ==> 77
N ==> 78
O ==> 79
P ==> 80
Q ==> 81
R ==> 82
S ==> 83
T ==> 84
U ==> 85
V ==> 86
W ==> 87
X ==> 88
Y ==> 89
Z ==> 90
```

Висновок: встановив IDE Intellij IDEA; створив репозиторій на GitLab; вивчив реалізацію базових алгоритмічних конструкцій у мові програмування Java; знайомство з правилами оформлення програмного коду.

Лабораторна робота №2

Тема: Створення структури класу заданої предметної області.

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

Завдання на лабораторну роботу

Завдання 2. Створити ієрархію класів відповідно до UML діаграми:

- поля класів повинні бути приховані модифікаторами доступу private, protected;

- створити конструктор без аргументів та з агрументами;

- 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 enam 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.

- продемонтрувати роботу оператору instanceof.

Вигляд коду:

```
public class Main {
  public static void main(String[] args) {
      Student student = new Student("FisrtName: Vladyslav", "lastName: Leos",
20, Location.CITY, Gender.MALE, "Web-developer");
```

```
        Car carForTeacher = new Car("Toyota", "Lanzer");
        Teacher teacher = new Teacher("FisrtName: Maruna", "lastName: Markovna",
35, Location.TOWN, Gender.FEMALE, "Ukr-language, History", carForTeacher);
        Car carForEmployee = new Car("Ford", "Kuga");
        Employee employee = new Employee("FisrtName: Oleg", "lastName: Ivanov",
40, Location.VILLAGE, Gender.MALE, "Pr-manandger", carForEmployee);


        System.out.println(student.getFullInfo());
        System.out.println(teacher.getFullInfo());
        System.out.println(employee.getFullInfo());

        Car.Engine teacherEngine = teacher.getCar().new Engine();
        teacherEngine.startEngine();
        System.out.println("Teacher's car engine running: " +
teacherEngine.isEngineWorks());

        Person.showCounter();

        if (teacher instanceof Person) {
            System.out.println("Teacher is an instance of Person.");
        }
    }
}
```

Результат виконання програми:

```
FisrtName: Vladyslav lastName: Leos, 20 age, MALE, location: CITY
FisrtName: Maruna lastName: Markovna, 35 age, FEMALE, location: TOWN
FisrtName: Oleg lastName: Ivanov, 40 age, MALE, location: VILLAGE
Engine started.
Teacher's car engine running: true
Value create objects: 6
Teacher is an instance of Person.
```

Завдання 3. Створити клас Operation з статичними методами addition, subtraction, multiplication, division, average, maximum, minimum, що приймають необмежену кількість аргументів через varargs. в методі main класу Main2 продемонструвати роботу методів класу Operation - вивести всі значення enam Location.

Вигляд коду:

```
public class Main2 {
    public static void main(String[] args) {
        int[] numbers = {6,2,4,5,6,7,11,15};

        System.out.println("Addition: " + Operation.addition(numbers));
        System.out.println("Subtraction: " + Operation.subtraction(numbers));
```

```
        System.out.println("Multiplication: " +
Operation.multiplication(numbers));
        System.out.println("Division: " + Operation.division(numbers));
        System.out.println("Average: " + Operation.average(numbers));
        System.out.println("Maximum: " + Operation.maximum(numbers));
        System.out.println("Minimum: " + Operation.minimum(numbers));

        // System.out.println("All Locations:");
        // for (Location loc : Location.values()) {
        //     System.out.println(loc);
        // }
    }
}
```

Результат виконання програми:

```
Addition: 56
Subtraction: -44
Multiplication: 1663200
Division: 2.164502164502164E-5
Average: 7.0
Maximum: 15
Minimum: 2
All Locations:
CITY
VILLAGE
TOWN
```

Висновок: створив ієрархію класів заданої предметної області, робота з статичними методами.

Лабораторна робота №3

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

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

Завдання на лабораторну роботу

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

- до команди можна додавати тільки учасників, що відносяться до одної ліги (Schoolar, 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.

Вигляд коду:

```
import Laba3.game.Schoolar;
import Laba3.game.Student;
import Laba3.game.Team;


public class Main {
    public static void main(String[] args) throws Exception {
      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 scollarTeam = new Team ("Dragon");
        scollarTeam.addNewParticipant(schoolar1);
        scollarTeam.addNewParticipant(schoolar2);
        scollarTeam.addNewParticipant(student1);
        scollarTeam.addNewParticipant(employee1);

        Team studentTeam = new Team("Vpered");
        studentTeam.addNewParticipant(student1);
        studentTeam.addNewParticipant(student2);
        Team employeeTeam = new Team("Robotyagi");
        employeeTeam.addNewParticipant(employee1);
        employeeTeam.addNewParticipant(employee2);
        Team anotherTeam = new Team("Fantaziya");
        anotherTeam.addNewParticipant("hello");

        Team scollarTeam2 = new Team("Rozumnyky");
        Schoolar schoolar3 = new Schoolar("Sergey", 12);
        Schoolar schoolar4 = new Schoolar("Olga", 14);
        scollarTeam2.addNewParticipant(schoolar3);
        scollarTeam2.addNewParticipant(schoolar4);
        scollarTeam.playWith(scollarTeam2);
        scollarTeam.playWith(employeeTeam);

        Team clonedScholarTeam = scollarTeam.cloneTeam();
        System.out.println("Original team: " + scollarTeam);
        System.out.println("Cloned team: " + clonedScholarTeam);

        System.out.println("Original team hashCode: " +
scollarTeam.hashCode());
        System.out.println("Cloned team hashCode: " +
clonedScholarTeam.hashCode());

        System.out.println("Teams are equal: " +
scollarTeam.equals(clonedScholarTeam));

        System.out.println("Participants comparison:");
        System.out.println(schoolar1 + " hashCode: " + schoolar1.hashCode());
        System.out.println(schoolar2 + " hashCode: " + schoolar2.hashCode());
        System.out.println("scholar1 equals scholar2: " +
schoolar1.equals(schoolar2));
    }
}
```

Результат виконання програми:

```
To the team Dragon was added participant Andriy
To the team Vpered was added participant Mykola
To the team Vpered was added participant Viktoria
To the team Robotyagi was added participant Andriy
To the team Dragon was added participant Mariya
To the team Dragon was added participant Mykola
To the team Dragon was added participant Andriy
To the team Vpered was added participant Mykola
To the team Vpered was added participant Viktoria
To the team Dragon was added participant Andriy
To the team Vpered was added participant Mykola
To the team Vpered was added participant Viktoria
To the team Robotyagi was added participant Andriy
To the team Robotyagi was added participant Oksana
To the team Fantaziya was added participant hello
To the team Vpered was added participant Viktoria
To the team Robotyagi was added participant Andriy
To the team Robotyagi was added participant Oksana
To the team Fantaziya was added participant hello
To the team Robotyagi was added participant Andriy
To the team Robotyagi was added participant Oksana
To the team Fantaziya was added participant hello
To the team Rozumnyky was added participant Sergey
To the team Robotyagi was added participant Oksana
To the team Fantaziya was added participant hello
To the team Rozumnyky was added participant Sergey
To the team Rozumnyky was added participant Sergey
To the team Rozumnyky was added participant Olga
The team Rozumnyky is winner!
The team Dragon is winner!
To the team Rozumnyky was added participant Olga
The team Rozumnyky is winner!
The team Dragon is winner!
The team Rozumnyky is winner!
The team Dragon is winner!
Original team: Team{teamName='Dragon'}
Cloned team: Team{teamName='Dragon'}
Original team hashCode: 314337396
Cloned team hashCode: 736709391
Teams are equal: false
Participants comparison:
Scholar{name='Ivan', age=13} hashCode: 71045348
Scholar{name='Mariya', age=15} hashCode: -1791087859
scholar1 equals scholar2: false
```

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

- продемонструвати роботу порівнянь на прикладі сортування учасників команд.

Вигляд коду:

```
mport Laba3.game.Age;
import Laba3.game.Participant;
import Laba3.game.Priority;
import Laba3.game.Student;

public class Main2 {
  public static void main(String[] args) {
        Student p1 = new Student("Mariya", 18);
        Student p2 = new Student("Mykola", 20);
        Student p3 = new Student("Viktoria", 21);
        Student p4 = new Student("Andriy", 28);
        Student p5 = new Student("Oksana", 25);
        Student p6 = new Student("Vladyslav", 25);

        List<Student> participants = new ArrayList<>();
        participants.add(p1);
        participants.add(p2);
        participants.add(p3);
        participants.add(p4);
        participants.add(p5);
        participants.add(p6);

        Collections.sort(participants);
        System.out.println("Sort on name:");
        participants.forEach(System.out::println);

        participants.sort(new Age());
        System.out.println("\nSort of age:");
        participants.forEach(System.out::println);

        participants.sort(Priority.byNameThenAge);
        System.out.println("\nSort of name , then of age");
        participants.forEach(System.out::println);
   }
}
```

Результат виконання програми:

```
Sort on name:
Scholar{name='Andriy', age=28}
Scholar{name='Mariya', age=18}
Scholar{name='Mykola', age=20}
Scholar{name='Oksana', age=25}
Scholar{name='Viktoria', age=21}
Scholar{name='Vladyslav', age=25}

Sort of age:
Scholar{name='Mariya', age=18}
Scholar{name='Mykola', age=20}
Scholar{name='Viktoria', age=21}
Scholar{name='Oksana', age=25}
Scholar{name='Vladyslav', age=25}
Scholar{name='Andriy', age=28}

Sort of name , then of age
Scholar{name='Andriy', age=28}
Scholar{name='Mariya', age=18}
Scholar{name='Mykola', age=20}
Scholar{name='Oksana', age=25}
Scholar{name='Viktoria', age=21}
Scholar{name='Vladyslav', age=25}
```

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

Лабораторна робота №4

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

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

Завдання на лабораторну роботу

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 підрядок "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.

Вигляд коду:

```
public class Task1 {
  public static void processString(String input) {
    System.out.println("Останній символ рядка: " + input.charAt(input.length()
- 1));
    System.out.println("Чи закінчується на \"!!!\": " +
input.endsWith("!!!"));
    System.out.println("Чи починається з \"I learn \": " + input.startsWith("I
learn "));
    System.out.println("Чи містить \"Java\": " + input.contains("Java"));

    int javaIndex = input.indexOf("Java");
    System.out.println("Позиція \"Java\" у рядку: " + javaIndex);

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

```
    System.out.println("У верхньому регістрі: " + input.toUpperCase());
    System.out.println("У нижньому регістрі: " + input.toLowerCase());

    String withoutJava = input.replace("Java", "");
    System.out.println("Рядок без \"Java\": " + withoutJava);
}

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

Результат виконання програми:

```
Останній символ рядка: !
Чи закінчується на "!!!": true
Чи починається з "I learn ": true
Чи містить "Java": true
Позиція "Java" у рядку: 8
Замінити "a" на "o": I leorn Jovo!!!
У верхньому регістрі: I LEARN JAVA!!!
У нижньому регістрі: i learn java!!!
Рядок без "Java": I learn !!!
```

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

- Дано два числа, наприклад, 4 і 36, необхідно скласти наступні рядки: 4 + 36 = 40 4 - 36 = -32 4 * 36 = 144

- Використати метод StringBuilder.append().

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

- Замініть символ “=” на слово “рівно”. Використати метод StringBuilder.replace(). - Змінити послідовність розташування символів в рядку на протилежну. Використати метод StringBuilder.reverse().

- Визначити довжину та capacity.

Вигляд коду:

```
public class Task2 {
  public static void main(String[] args) {
    int num1 = 50;
    int num2 = 23;

    StringBuilder addition = new StringBuilder();
```

```
    addition.append(num1).append(" + ").append(num2).append(" = ").append(num1
+ num2);
    System.out.println("Додавання: " + addition);

    StringBuilder subtraction = new StringBuilder();
    subtraction.append(num1).append(" - ").append(num2).append(" =
").append(num1 - num2);
    System.out.println("Віднімання: " + subtraction);

    StringBuilder multiplication = new StringBuilder();
    multiplication.append(num1).append(" * ").append(num2).append(" =
").append(num1 * num2);
    System.out.println("Множення: " + multiplication);

    StringBuilder additionModified = new StringBuilder(addition);
    additionModified.deleteCharAt(additionModified.indexOf("="));
    additionModified.insert(additionModified.indexOf(" ") + 3, "рівно");
    System.out.println("Додавання з \"рівно\" " + additionModified);

    StringBuilder subtractionModified = new StringBuilder(subtraction);
    int equalsIndex = subtractionModified.indexOf("=");
    subtractionModified.replace(equalsIndex, equalsIndex + 1, "рівно");
    System.out.println("Віднімання з \"рівно\" " + subtractionModified);

    StringBuilder reversedAddition = new StringBuilder(addition).reverse();
    System.out.println("Додавання у зворотньому порядку: " +
reversedAddition);

    System.out.println("Довжина рядка додавання: " + addition.length());
    System.out.println("Ємність рядка додавання: " + addition.capacity());
}
}
```

Результат виконання програми:

```
Додавання: 50 + 23 = 73
Віднімання: 50 - 23 = 27
Множення: 50 * 23 = 1150
Додавання з "рівно" 50 + р?вно23  73
Віднімання з "рівно" 50 - 23 р?вно 27
Додавання у зворотньому порядку: 37 = 32 + 05
Довжина рядка додавання: 12
Ємність рядка додавання: 16
PS D:\js\JavaProject\JavaTasks>
```

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.

Вигляд коду:

```
import java.util.Formatter;
import java.util.List;
import java.util.ArrayList;

class Tovar {
    private final String brand;
    private final String size;
    private final String category;
    private final double price;

    public Tovar(String brand, String size, String category, double price) {
      this.brand = brand;
      this.size = size;
      this.category = category;
      this.price = price;
    }

    public String getBrand() {
        return brand;
    }

    public String getSize() {
        return size;
    }

    public String getCategory() {
```

```
        return category;
    }

    public double getPrice() {
        return price;
    }
}

public class Task3 {
    public static void main(String[] args) {
        List<Tovar> products = new ArrayList<>();
        products.add(new Tovar("Michelin","195/55 R15", "Зимня", 1500.78));
        products.add(new Tovar("Continental", "195/75 R16C", "Літня",
1000.56));
        products.add(new Tovar("BridgeStone", "215/55 R16","Зимня", 500.78));
        products.add(new Tovar("Matador", "215/55 R17","Літня", 300.45));
        products.add(new Tovar("Firelli", "215/60 R17","ВсіСезона", 2200.99));
        products.add(new Tovar("Good&Year", "215/50 R17","Зимня", 1200.30));
        products.add(new Tovar("Hankook", "215/60 R17","Зимня", 1800.10));
        products.add(new Tovar("Dunlop", "205/65 R15","ВсіСезона", 999.99));

        String dateTime = "28.03.2019 13:25:12";
        double totalAmount = 0;

        Formatter formatter = new Formatter();
        formatter.format("Дата та час покупки: %s\n", dateTime);
        formatter.format("=========================================\n");
        formatter.format("%-3s %-10s %-15s %-10s %10s\n", "№", "Товар",
"Розмір", "Категорія", "Ціна");
        formatter.format("=========================================\n");

        for (int i = 0; i < products.size(); i++) {
            Tovar product = products.get(i);
            formatter.format("%-3d %-10s %-15s %-10s %10.2f \n", i + 1,
product.getBrand(), product.getSize(), product.getCategory(),
product.getPrice());
            totalAmount += product.getPrice();
        }

        formatter.format("=========================================\n");
        formatter.format("Разом: %34.2f \n", totalAmount);

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

Результат виконання програми:

```
Дата та час покупки: 28.03.2019 13:25:12
=========================================
№   Товар       Розмір          Категорія       Ціна
=========================================
1   Michelin    195/55 R15      Зимня         1500,78
2   Continental 195/75 R16C      Літня         1000,56
3   BridgeStone 215/55 R16       Зимня          500,78
4   Matador     215/55 R17      Літня          300,45
5   Firelli     215/60 R17      ВсіСезона     2200,99
6   Good&Year   215/50 R17      Зимня         1200,30
7   Hankook     215/60 R17      Зимня         1800,10
8   Dunlop      205/65 R15      ВсіСезона      999,99
=========================================
Разом:                                9503,95
```

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.

Вигляд коду:

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

public class Task5 {
    public static void main(String[] args) {
        LocalDateTime labStart = LocalDateTime.of(2024, 11, 11, 10, 0, 0);
        System.out.println("Дата та час початку лабораторної: " + labStart);

        System.out.println("День тижня: " + labStart.getDayOfWeek());
        System.out.println("День у році: " + labStart.getDayOfYear());
        System.out.println("Місяць: " + labStart.getMonth());
```

```
        System.out.println("Рік: " + labStart.getYear());
        System.out.println("Години: " + labStart.getHour());
        System.out.println("Хвилини: " + labStart.getMinute());
        System.out.println("Секунди: " + labStart.getSecond());

        boolean isLeapYear = Year.of(labStart.getYear()).isLeap();
        System.out.println("Чи рік високосний: " + isLeapYear);

        LocalDateTime currentDateTime = LocalDateTime.now();
        System.out.println("\nПоточний час: " + currentDateTime);

        if (currentDateTime.isAfter(labStart)) {
            System.out.println("Поточний час після початку лабораторної.");
        } else if (currentDateTime.isBefore(labStart)) {
            System.out.println("Поточний час до початку лабораторної.");
        } else {
            System.out.println("Поточний час співпадає з початком
лабораторної.");
        }

        LocalDateTime modifiedDateTime = labStart
                .plusDays(1)
                .minusHours(2)
                .withMinute(45);

        System.out.println("\nЗмінена дата та час: " + modifiedDateTime);

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy
HH:mm:ss");
        System.out.println("Форматована змінена дата та час: " +
modifiedDateTime.format(formatter));
    }
}
```

Результат виконання програми:

```
Дата та час початку лабораторної: 2024-11-11T10:0
День тижня: MONDAY
День у році: 316
Місяць: NOVEMBER
Рік: 2024
Години: 10
Хвилини: 0
Секунди: 0
Чи рік високосний: true

Поточний час: 2024-12-16T11:36:16.366733800
Поточний час п?сля початку лабораторної.
```

Висновок: опрацював з класами String, StringBuffer, StringBuilder та їх методами; використав локалізації та інтернаціоналізації; робота з датами.

Лабораторна робота №5

Тема: Java Collections Framework

Мета роботи: робота з Java Collections Framework

Завдання на лабораторну роботу

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)

Вигляд коду:

```
public class ProductArray {
  public static void main(String[] args) {
        List<Product> products = new ArrayList<>();

        products.add(new Product("Laptop", 50, 150));
        products.add(new Product("Smartphone", 60, 70));
        products.add(new Product("Tablet", 100.50, 50));
        products.add(new Product("PC", 230, 70));

        List<Product> moreProducts = new ArrayList<>();
        moreProducts.add(new Product("Colunm", 150, 30));
        moreProducts.add(new Product("TV", 450, 120));
        products.addAll(moreProducts);

        System.out.println("Продукт за індексом 2: " + products.get(2));

        int index = products.indexOf(new Product("Mouse", 150, 70));
        System.out.println("Індекс 'Mouse': " + index);

        products.remove(1);
        System.out.println("Після видалення продукту за індексом 1: " +
products);

        products.set(0, new Product("Rollet", 240, 2000));
        System.out.println("Після заміни першого продукту: " + products);


        boolean contains = products.contains(new Product("Tablet", 456, 300));
        System.out.println("Містить 'Tablet': " + contains);
```

```
        products.sort(Comparator.comparingDouble(Product::getPrice));
        System.out.println("Список після сортування за ціною: " + products);

        List<Product> subList = products.subList(0, 2);
        System.out.println("Підсписок: " + subList);
        Object[] productArray = products.toArray();
        System.out.println("Масив продуктів: " +
Arrays.toString(productArray));

        products.clear();
        System.out.println("Список після очищення: " + products);

        System.out.println("Список порожній: " + products.isEmpty());
    }
```

Результат виконання програми:

```
Продукт за індексом 2: Продукт {Назва ='Tablet',Ціна ='100.5',Кількість =50}
Індекс 'Mouse': -1
Після видалення продукту за індексом 1: [Продукт {Назва ='Laptop',Ціна ='50.0',Кількість =150}, Продукт {Назва ='Tablet',Ціна ='100.5',Кількість =50}, Продукт {Назва ='PC'
на ='230.0',Кількість =70}, Продукт {Назва ='Column',Ціна ='150.0',Кількість =30}, Продукт {Назва ='TV',Ціна ='450.0',Кількість =120}]
Після заміни першого продукту: [Продукт {Назва ='Rollet',Ціна ='240.0',Кількість =2000}, Продукт {Назва ='Tablet',Ціна ='100.5',Кількість =50}, Продукт {Назва ='PC',Ціна =
0.0',Кількість =70}, Продукт {Назва ='Column',Ціна ='150.0',Кількість =30}, Продукт {Назва ='TV',Ціна ='450.0',Кількість =120}]
Містить 'Tablet': false
Список після сортування за ціною: [Продукт {Назва ='Tablet',Ціна ='100.5',Кількість =50}, Продукт {Назва ='Column',Ціна ='150.0',Кількість =30}, Продукт {Назва ='PC',Ціна
30.0',Кількість =70}, Продукт {Назва ='Rollet',Ціна ='240.0',Кількість =2000}, Продукт {Назва ='TV',Ціна ='450.0',Кількість =120}]
Підсписок: [Продукт {Назва ='Tablet',Ціна ='100.5',Кількість =50}, Продукт {Назва ='Column',Ціна ='150.0',Кількість =30}]
Масив продуктів: [Продукт {Назва ='Tablet',Ціна ='100.5',Кількість =50}, Продукт {Назва ='Column',Ціна ='150.0',Кількість =30}, Продукт {Назва ='PC',Ціна ='230.0',Кількіст
70}, Продукт {Назва ='Rollet',Ціна ='240.0',Кількість =2000}, Продукт {Назва ='TV',Ціна ='450.0',Кількість =120}]
Список після очищення: []
Список порожній: true
```

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)

Вигляд коду:

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


        Queue.push(new Product("Laptop", 1500, 130));
        Queue.offerLast(new Product("Smartphone", 800, 40));
        Queue.offerLast(new Product("Tablet", 500, 50));
        Queue.push(new Product("Smartwatch", 200.50,50 ));

        System.out.println("Перший продукту:" + Queue.getFirst());
        System.out.println("Останій продукт:" + Queue.peekLast());

        System.out.println("Видалення першого продукту:" + Queue.pop());
        System.out.println("Видалення останій продукту: " + Queue.pollLast());
```

```
        System.out.println("Видалення останій продукту: " +
Queue.removeLast());
    }
}
```

Результат виконання програми:

```
Перший продукту:Продукт {Назва ='Smartwatch',Ціна ='200.5',Кількість =50}
Останій продукт:Продукт {Назва ='Tablet',Ціна ='500.0',Кількість =50}
Видалення першого продукту:Продукт {Назва ='Smartwatch',Ціна ='200.5',Кількість =50}
Видалення останій продукту: Продукт {Назва ='Tablet',Ціна ='500.0',Кількість =50}
Видалення останій продукту: Продукт {Назва ='Smartphone',Ціна ='800.0',Кількість =40}
```

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

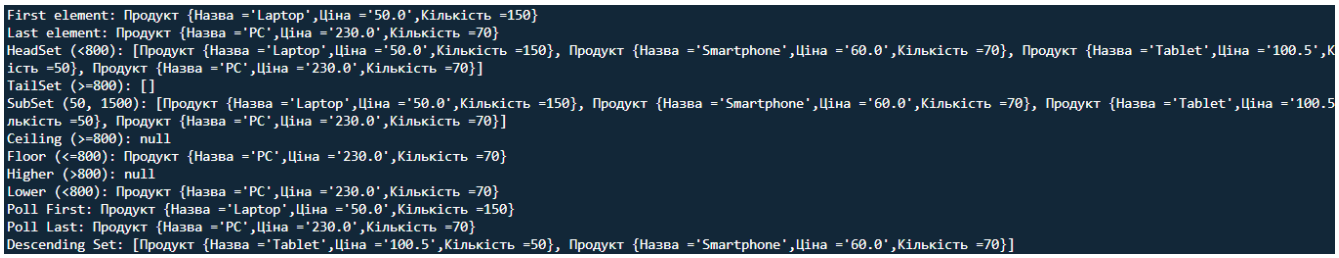
• Використовуємо клас TreeSet.

• Продемонструвати роботу з множиною використовуючи методи (add, first, last, headSet, subSet, tailSet, ceiling, floor, higher, lower, pollFirst, pollLast, descendingSet)

Вигляд коду:

```
public class TreeSetArray {
  public static void main(String[] args) {
    TreeSet <Product> products = new TreeSet<>();

    products.add(new Product("Laptop", 50, 150));
    products.add(new Product("Smartphone", 60, 70));
    products.add(new Product("Tablet", 100.50, 50));
    products.add(new Product("PC", 230, 70));
        System.out.println("First element: " + products.first());
        System.out.println("Last element: " + products.last());
        System.out.println("HeadSet (<800): " + products.headSet(new
Product("", 800, 45)));
        System.out.println("TailSet (>=800): " + products.tailSet(new
Product("", 800, 89)));
        System.out.println("SubSet (50, 1500): " + products.subSet(new
Product("", 50 , 56), new Product("", 1500, 56)));
        System.out.println("Ceiling (>=800): " + products.ceiling(new
Product("", 800,56)));
        System.out.println("Floor (<=800): " + products.floor(new Product("",
800, 50)));
        System.out.println("Higher (>800): " + products.higher(new Product("",
800, 58)));
        System.out.println("Lower (<800): " + products.lower(new Product("",
800, 89)));
        System.out.println("Poll First: " + products.pollFirst());
        System.out.println("Poll Last: " + products.pollLast());
        System.out.println("Descending Set: " + products.descendingSet());
  }}
```

Результат виконання програми:



Завдання 5. 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)

Вигляд коду:

```
public class Map {
  public static void main (String args[]){
    HashMap<String, Product> map = new HashMap<>();
    map.put("Laptop", new Product("Laptop", 50, 150));
    map.put("Smartphone", new Product("Smartphone", 60, 70));
    map.put("Tablet", new Product("Tablet", 100.50, 50));
    map.put("PC", new Product("PC", 230, 70));

    System.out.println("Laptop: " + map.get("Laptop"));

    System.out.println("Contains key 'Jeans': " + map.containsKey("Jeans"));
    System.out.println("Contains value Product('Shoes', 70.00): " +
map.containsValue(new Product("Shoes", 70.00, 67)));

    System.out.println("Keys: " + map.keySet());
    System.out.println("Values: " + map.values());

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

    map.remove("Jeans");
    map.putIfAbsent("Tablet", new Product("Tablet", 300.99 , 50));
    System.out.println("Map size: " + map.size());
```

```
    map.clear();
    System.out.println("Map is empty: " + map.isEmpty());
  }
}
```

Результат виконання програми:

```
Laptop: Продукт {Назва ='Laptop',Ціна ='50.0',Кількість =150}
Contains key 'Jeans': false
Contains value Product('Shoes', 70.00): false
Keys: [Laptop, PC, Tablet, Smartphone]
Values: [Продукт {Назва ='Laptop',Ціна ='50.0',Кількість =150}, Продукт {Назва ='PC',Ціна ='230.0',Кількість =70}, Продукт
азва ='Tablet',Ціна ='100.5',Кількість =50}, Продукт {Назва ='Smartphone',Ціна ='60.0',Кількість =70}]
Key: Laptop, Value: Продукт {Назва ='Laptop',Ціна ='50.0',Кількість =150}
Key: PC, Value: Продукт {Назва ='PC',Ціна ='230.0',Кількість =70}
Key: Tablet, Value: Продукт {Назва ='Tablet',Ціна ='100.5',Кількість =50}
Key: Smartphone, Value: Продукт {Назва ='Smartphone',Ціна ='60.0',Кількість =70}
Map size: 4
Map is empty: true
```

Завдання 6. Продемонструвати роботу з класом Collections:

• Для роботи використати масив створений через Arrays.asList

• Метод Collections.sort()

• 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()

Вигляд коду:

```
public class ArrayCollections {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(3, 7, 2, 8, 5, 1);

        Collections.sort(numbers);
        System.out.println("Sorted: " + numbers);

        int index = Collections.binarySearch(numbers, 3);
        System.out.println("Index of 3: " + index);

        Collections.reverse(numbers);
        System.out.println("Reversed: " + numbers);
```

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

        Collections.fill(numbers, 5);
        System.out.println("Filled with 5: " + numbers);

        numbers = Arrays.asList(3, 7, 2, 8, 5, 1);
        System.out.println("Max: " + Collections.max(numbers));
        System.out.println("Min: " + Collections.min(numbers));


        List<Integer> copy = new ArrayList<>(Arrays.asList(new
Integer[numbers.size()]));
        Collections.copy(copy, numbers);
        System.out.println("Copied: " + copy);

        Collections.rotate(numbers, 10);
        System.out.println("Rotated: " + numbers);

        List<String> checkedList = Collections.checkedList(new ArrayList<>(),
String.class);
        checkedList.add("Hello");
        System.out.println("Checked List: " + checkedList);

        numbers = Arrays.asList(3, 7, 2, 8, 5, 1, 3, 3);
        System.out.println("Frequency of 3: " + Collections.frequency(numbers,
3));
    }
}
```

Результат виконання програми:

```
Sorted: [1, 2, 3, 5, 7, 8]
Index of 3: 2
Reversed: [8, 7, 5, 3, 2, 1]
Shuffled: [5, 2, 7, 3, 8, 1]
Filled with 5: [5, 5, 5, 5, 5, 5]
Max: 8
Min: 1
Copied: [3, 7, 2, 8, 5, 1]
Rotated: [2, 8, 5, 1, 3, 7]
Checked List: [Hello]
Frequency of 3: 3
```

Висновок: опрацював з Java Collections Framework

Лабораторна робота №7

Тема: Багатопоточне програмування в Java

Мета роботи: практика роботи з потоками в Java

Завдання на лабораторну роботу

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 пріорітет, превірити чи він живий, чи є потоком демоном.

• Змінити ім'я, пріорітет створеного потоку та вивести в консоль оновлені значення.
• Після завершення роботи створеного потоку (використати метод join()) вивести ім’я головного потоку, та його пріорітет.

• Відобразити в консолі, коли ваш потік буде в стані NEW, RUNNUNG, TERMINATED.

Вигляд коду:

```
class TaskThread extends Thread {
  public void run () {
      for(int i = 0 ; i < 100 ; i++){
        System.out.println("Я люблю програмувати)");
      }
  }
}

public class Task2 {
  public static void main(String[] args) throws InterruptedException {
    TaskThread thread = new TaskThread();

    System.out.println("Ім'я потоку: " + thread.getName());
    System.out.println("Пріоритет: " + thread.getPriority());
    System.out.println("Живий: " + thread.isAlive());
    System.out.println("Демон: " + thread.isDaemon());


    thread.start();
    System.out.println("Потік в стані RUNNING.");
```

```
    thread.setName("ProgrammingThread");
    thread.setPriority(Thread.MAX_PRIORITY);
    System.out.println("Оновлене ім'я: " + thread.getName());
    System.out.println("Оновлений пріоритет: " + thread.getPriority());

    thread.join();
    System.out.println("Потік завершився. Стан TERMINATED.");

    Thread mainThread = Thread.currentThread();
    System.out.println("Ім'я головного потоку: " + mainThread.getName());
    System.out.println("Пріоритет головного потоку: " +
mainThread.getPriority());
}
}
```

Результат виконання програми:

```
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Я люблю програмувати)
Оновлений пріоритет: 10
Потік завершився. Стан TERMINATED.
Ім'я головного потоку: main
Пріоритет головного потоку: 5
```

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!!!»

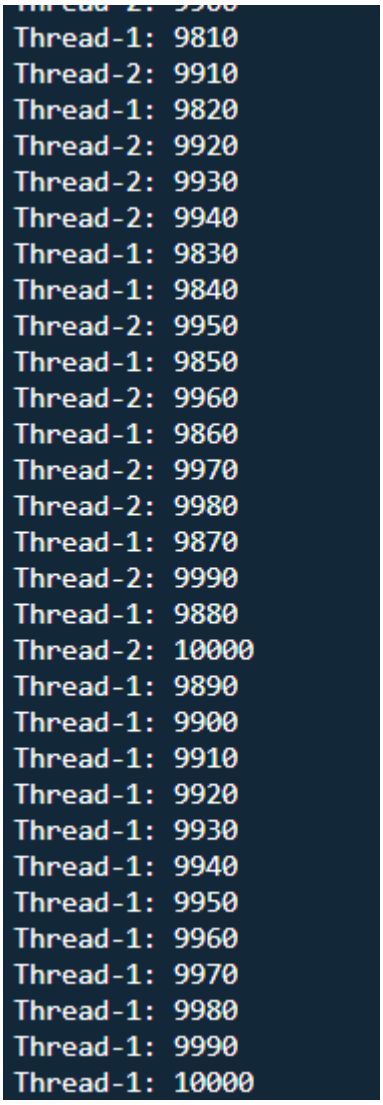
• Створити три потоки, які виконують завдання друку значень.

• Використовуємо статичний метод Thread.sleep(), щоб зробити паузу на 2 секунди для головного потоку, а після цього викликати для створених потоків метод interrupt().

Вигляд коду:

```
class MyRunnable implements Runnable {
  @Override
  public void run() {
      for (int i = 0; i <= 10000; i += 10) {
          if (Thread.currentThread().isInterrupted()) {
              System.out.println("Розрахунок завершено!!!");
              return;
          }
          System.out.println(Thread.currentThread().getName() + ": " + i);
      }
  }
}

public class Task3 {
  public static void main(String[] args) throws InterruptedException {
    Thread thread1 = new Thread(new MyRunnable());
    Thread thread2 = new Thread(new MyRunnable());
    Thread thread3 = new Thread(new MyRunnable());

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

    Thread.sleep(2000);
    thread1.interrupt();
    thread2.interrupt();
    thread3.interrupt();
}
}
```

Результат виконання програми:



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.

• 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 корректну роботу використовуючи синхронізований метод.

• Створити три потоки, які виконують завдання друку значень.

Вигляд коду:

```
class ArithmeticProgression implements Runnable {
  private static int result = 1;

  private synchronized void printProgression() {
      for (int i = 0; i < 100; i++) {
          System.out.print(result + " ");
          result++;
          try {
              Thread.sleep(200);
          } catch (InterruptedException e) {
              Thread.currentThread().interrupt();
          }
      }
      System.out.println();
  }

  @Override
  public void run() {
      printProgression();
  }
}

public class Task4 {
  public static void main(String[] args) {
      Thread thread1 = new Thread(new ArithmeticProgression());
      Thread thread2 = new Thread(new ArithmeticProgression());
      Thread thread3 = new Thread(new ArithmeticProgression());

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

Результат виконання програми:

```
 1 1 1 4 4 4 7 7 7 10 10 10 13 13 13 16 16 16 19 19 19 22 22 22 25 25 27 28 28 30 31 31 33 34 34 36 37 37 39 40 40 42 43 43 4546 46 48 49 49 51 52 52
 0 61 61 63 64 64 66 67 68 69 70 70 72 73 73 75 76 76 78 79 79 81 82 82 84 85 85 87 88 88 90 91 91 93 94 94 96 97 97 99 100 100 102 103 103 105 106 1
 12 112 114 115 115 117 118 118 120 121 121 123 124 124 126 127 127 129 130 130 132 133 133 135 136 136 138 139 139 141 142 142 144 145 145 147 148 1
 54 154 156 157 157 159 160 160 162 163 163 165 166 166 168 169 169 169 172 172 172 175 175 177 178 178 180 181 181 183 184 184 186 187 187 189 190 1
 96 196 198 199 199 201 202 202 204 205 205 207 208 208 210 211 211 213 214 214 216 217 217 219 220 220 222 223 223 225 226 226 228 229 229 231 232 2
 38 238 240 241 241 243 244 244 246 247 247 249 250 250 252 253 253 255 256 256 258 259 259 261 262 262 264 265 265 267 268 268 270 271 271 273 274 2
 80 280 282 283 283 285 286 286 288 289 289 291 292 292 294 295 295 297 298 299 300
```

Завдання 5. Переробити 4 завдання використовуючи блок синхронізації.

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

• Змінну треба використати як об'єкт для синхронізації.

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

Вигляд коду:

```
class SharedData {
  private String message = "";

  public synchronized void write(String msg) throws InterruptedException {
      message = msg;
      notify();
      wait();
  }

  public synchronized void read() throws InterruptedException {
      System.out.println("Printer: " + message);
      notify();
      wait();
  }
}

public class Task6 {
  public static void main(String[] args) {
      SharedData data = new SharedData();

      Thread reader = new Thread(() -> {
          java.util.Scanner scanner = new java.util.Scanner(System.in);
          try {
              while (true) {
                  System.out.print("Reader: Введіть повідомлення: ");
                  String input = scanner.nextLine();
                  if (input.equalsIgnoreCase("exit")) break;
                  data.write(input);
              }
          } catch (InterruptedException e) {
              Thread.currentThread().interrupt();
          }
      });

      Thread printer = new Thread(() -> {
          try {
              while (!Thread.currentThread().isInterrupted()) {
                  data.read();
              }
          } catch (InterruptedException e) {
              Thread.currentThread().interrupt();
          }
```

```
        });

        reader.start();
        printer.start();
    }
```

Результат виконання програми:

```
Reader: Введ?ть пов?домлення: Hello World
Printer: Hello World
```

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.

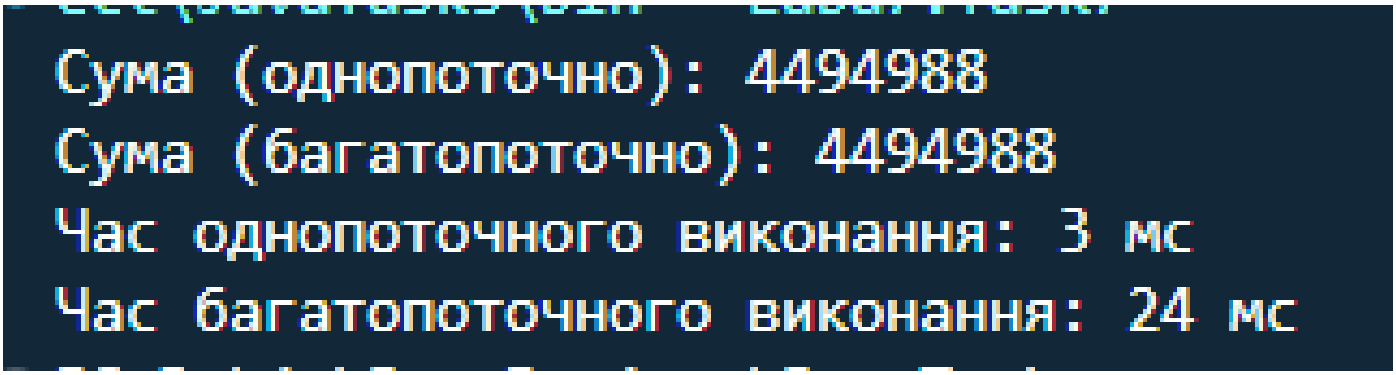
• Заміряти час виконання обох варіантів завдання використовуючи System.currentTimeMillis() та вивести результати в консоль.

Вигляд коду:

```
public class Task7 {
    private static final int ARRAY_SIZE = 1_000_000;
    private static final int[] array = new int[ARRAY_SIZE];

    public static void main(String[] args) throws InterruptedException {
        fillArray();

        long singleThreadTime = calculateSingleThread();
        long multiThreadTime = calculateMultiThread();

        System.out.println("Час однопоточного виконання: " + singleThreadTime
+ " мс");
        System.out.println("Час багатопоточного виконання: " + multiThreadTime
+ " мс");
    }

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

    private static long calculateSingleThread() {
```

```
        long startTime = System.currentTimeMillis();
        long sum = 0;
        for (int value : array) {
            sum += value;
        }
        long endTime = System.currentTimeMillis();
        System.out.println("Сума (однопоточно): " + sum);
        return endTime - startTime;
    }

    private static long calculateMultiThread() throws InterruptedException {
        long startTime = System.currentTimeMillis();
        ExecutorService executor = Executors.newFixedThreadPool(5);

        int chunkSize = ARRAY_SIZE / 5;
        Callable<Long>[] tasks = new Callable[5];
        for (int i = 0; i < 5; i++) {
            final int start = i * chunkSize;
            final int end = (i == 4) ? ARRAY_SIZE : start + chunkSize;
            tasks[i] = () -> {
                long sum = 0;
                for (int j = start; j < end; j++) {
                    sum += array[j];
                }
                return sum;
            };
        }

        long sum = executor.invokeAll(java.util.Arrays.asList(tasks))
                .stream()
                .mapToLong(f -> {
                    try {
                        return f.get();
                    } catch (Exception e) {
                        throw new RuntimeException(e);
                    }
                }).sum();

        executor.shutdown();
        long endTime = System.currentTimeMillis();
        System.out.println("Сума (багатопоточно): " + sum);
        return endTime - startTime;
    }
}
```

Результат виконання програми:



Висновок:опрацював з потоками в Java

Лабораторна робота №8

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

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

Завдання на лабораторну роботу

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.

Вигляд коду:

```
class Product {
  private String name;
  private String brand;
  private double price;
  private int quantity;

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

  public String getName() {
    return name;
  }

  public String getBrand() {
    return brand;
  }

  public Double getPrice() {
    return price;
```

```
  }

  public int getQuntity() {
    return quantity;
  }

  public String toString() {
    return "Продукт {" +
        "Назва ='" + name + '\'' +
        "Бренд ='" + brand + '\'' +
        ",Ціна ='" + price + '\'' +
        ",Кількість =" + quantity +
        "}";
  }
}

public class Task4 {
  public static void main(String[] args) {
    List<Product> products = Arrays.asList(
        new Product("Ноутбук", "BrandA", 1500, 10),
        new Product("Сматрофон", "BrandB", 800, 20),
        new Product("Навушники", "BrandC", 150, 50),
        new Product("Монітор", "BrandA", 300, 15),
        new Product("Клавіатура", "BrandB", 80, 25));

    System.out.println("Brands:");
    products.stream()
        .map(Product::getBrand)
        .distinct()
        .forEach(System.out::println);

    System.out.println("\nПродуктів з ціною більше ніж 1000:");
    products.stream()
        .filter(p -> p.getPrice() < 1000)
        .limit(2)
        .forEach(System.out::println);


    int totalCount = products.stream()
        .map(Product::getQuntity)
        .reduce(0, Integer::sum);
    System.out.println("\nЗагальна кількість всіх продуктів: " + totalCount);


    System.out.println("\nGrouped by brand:");
    Map<String, List<Product>> groupedBrand = products.stream()
        .collect(groupingBy(Product::getBrand));
```

```
    groupedBrand.forEach((brand, productList) -> {
      System.out.println(brand + ": " + productList);
    });

    System.out.println("\nСортування за ціною:");
    List<Product> sortedByPrice = products.stream()
        .sorted(Comparator.comparingDouble(Product::getPrice))
        .collect(toList());
    sortedByPrice.forEach(System.out::println);
  }
}
```

Результат виконання програми:

```
Brands:
BrandA
BrandB
BrandC

Продуктів з ціною більше ніж 1000:
Продукт {Назва ='Сматрофон'Бренд ='BrandB',Ціна ='800.0',Кількість =20}
Продукт {Назва ='Навушники'Бренд ='BrandC',Ціна ='150.0',Кількість =50}

Загальна кількість всіх продуктів: 120

Grouped by brand:
BrandC: [Продукт {Назва ='Навушники'Бренд ='BrandC',Ціна ='150.0',Кількість =50}]
BrandA: [Продукт {Назва ='Ноутбук'Бренд ='BrandA',Ціна ='1500.0',Кількість =10}, Продукт {Назва ='Монітор'Бренд ='BrandA',Ціна ='300.0',Кількість =15}]
BrandB: [Продукт {Назва ='Сматрофон'Бренд ='BrandB',Ціна ='800.0',Кількість =20}, Продукт {Назва ='Клавіатура'Бренд ='BrandB',Ціна ='80.0',Кількість =25}]

Сортування за ціною:
Продукт {Назва ='Клавіатура'Бренд ='BrandB',Ціна ='80.0',Кількість =25}
Продукт {Назва ='Навушники'Бренд ='BrandC',Ціна ='150.0',Кількість =50}
Продукт {Назва ='Монітор'Бренд ='BrandA',Ціна ='300.0',Кількість =15}
Продукт {Назва ='Сматрофон'Бренд ='BrandB',Ціна ='800.0',Кількість =20}
Продукт {Назва ='Ноутбук'Бренд ='BrandA',Ціна ='1500.0',Кількість =10}
```

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

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

Вигляд коду:

```
public class Task6 {
  public static void main(String[] args) {
        Integer[] numbers = {12, 45, 78, 34, 89, 123};

        String result = Arrays.stream(numbers)
                              .max(Integer::compareTo)
                              .map(String::valueOf)
                              .orElse("Числа відсутні");

        System.out.println("Максимальне значення: " + result);

        Integer[] emptyArray = {};
        String emptyResult = Arrays.stream(emptyArray)
                                   .max(Integer::compareTo)
                                   .map(String::valueOf)
```

```
                                    .orElse("Числа відсутні");

        System.out.println("Результат для порожнього масиву: " + emptyResult);
    }
}
```

Результат виконання програми:

```
Максимальне значення: 123
Результат для порожнього масиву: Числа відсутні
```

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

Лабораторна робота №9

Тема: Регулярні вирази. Рефлексія. Анотації.

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

Завдання на лабораторну роботу

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.

Вигляд коду:

```
public class Task1 {
    public static void main(String[] args) {
        String text = "Іваненко Іван Іванович, 25 років, посада: Інженер,
досвід: 3 роки, адреса: вул. Центральна 1, Київ, емайл:
ivan.ivanenko@gmail.com, телефон: +380(67)123-45-67.\n"+
        "Петрова Марія Петрівна, 30 років, посада: Бухгалтер, досвід: 5 років,
адреса: вул. Шевченка 10, Львів, емайл: maria.petrova@ukr.net, телефон:
+380(50)987-65-43.\n";

        System.out.println("Телефони:");
        Pattern phonePattern = Pattern.compile("\\+380\\(\\d{2}\\)\\d{3}-
\\d{2}-\\d{2}");
        Matcher phoneMatcher = phonePattern.matcher(text);
        while (phoneMatcher.find()) {
            System.out.println(phoneMatcher.group());
        }

        System.out.println("\nЕмайли:");
        Pattern emailPattern = Pattern.compile("\\b[\\w.%+-]+@[\\w.-]+\\.[a-
zA-Z]{2,}\\b");
        Matcher emailMatcher = emailPattern.matcher(text);
        while (emailMatcher.find()) {
            System.out.println(emailMatcher.group());
        }
```

```
        System.out.println("\nОновлені дати народження:");
        String textWithDates =
        "Іваненко Іван Іванович, дата народження: 20.05.1995.\n"+
        "Петрова Марія Петрівна, дата народження: 15.08.1992.\n"+
        "Сидоренко Андрій Олексійович, дата народження: 01.01.1988";

        String updatedDates =
textWithDates.replaceAll("(\\d{2})\\.(\\d{2})\\.(\\d{4})", "$3-$2-$1");
        System.out.println(updatedDates);

        System.out.println("Оновлені посади:");
        String updatedText = text.replaceAll("Інженер", "Старший інженер")
                                 .replaceAll("Бухгалтер", "Головний
бухгалтер");
        System.out.println(updatedText);
    }
}
```

Результат виконання програми:

```
Телефони:
+380(67)123-45-67
+380(50)987-65-43

Емайли:
ivan.ivanenko@gmail.com
maria.petrova@ukr.net

Оновлен? дати народження:
?ваненко ?ван ?ванович, дата народження: 1995-05-20.
Петрова Мар?я Петр?вна, дата народження: 1992-08-15.
Сидоренко Андр?й Олекс?йович, дата народження: 1988-01-01
Оновлен? посади:
?ваненко ?ван ?ванович, 25 рок?в, посада: Старший ?нженер, досв?д: 3 роки, адреса: вул. Центральна 1, Київ,
anenko@gmail.com, телефон: +380(67)123-45-67.
Петрова Мар?я Петр?вна, 30 рок?в, посада: Головний бухгалтер, досв?д: 5 рок?в, адреса: вул. Шевченка 10, Льв
ia.petrova@ukr.net, телефон: +380(50)987-65-43.
```

Завдання 3. Робота з користувацьким класом методами Reflection API:

• Створити власний клас в якому міститимуться публічні та приватні поля, коструктори і методи з аргументами та без.

• Отримати об'єкт класу Class для користувацького класу трьома способами.

• Отримати всі поля, методи, конструктори, що визначені тільки в цьому класі (не враховувати наслідування) та вивести ці значення в консоль (назву, типи параметрів та значення, що повертається).

• Створити екземпляр класу.

• Викликати метод класу.

• Попрацювати з приватним полем (встановити та отримати значення)

• Результати роботи відобразити в консолі.

Вигляд коду:

```
class Employee {
  public String name;
  private int age;
  private String position;

  public Employee() {
  }

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

  public void work() {
      System.out.println(name + " is working as " + position);
  }

  private void setPosition(String position) {
      this.position = position;
  }

  private String getPosition() {
      return position;
  }
}

public class Task2 {
    public static void main(String[] args) throws Exception {
        Class<Employee> employeeClass = Employee.class;
        System.out.println("Class: " + employeeClass.getName());


        System.out.println("\nFields:");
        for (Field field : employeeClass.getDeclaredFields()) {
            System.out.println(field);
        }

        System.out.println("\nMethods:");
        for (Method method : employeeClass.getDeclaredMethods()) {
            System.out.println(method);
        }
```

```
        System.out.println("\nConstructors:");
        for (Constructor<?> constructor :
employeeClass.getDeclaredConstructors()) {
            System.out.println(constructor);
        }


        Employee employee =
employeeClass.getDeclaredConstructor().newInstance();

        employee.name = "Іваненко Іван";
        Method workMethod = employeeClass.getMethod("work");
        workMethod.invoke(employee);

        Field positionField = employeeClass.getDeclaredField("position");
        positionField.setAccessible(true);
        positionField.set(employee, "Інженер");

        Method getPositionMethod =
employeeClass.getDeclaredMethod("getPosition");
        getPositionMethod.setAccessible(true);
        String position = (String) getPositionMethod.invoke(employee);
        System.out.println("Посада: " + position);
    }
}
```

Результат виконання програми:

```
Fields:
public java.lang.String Laba9.Employee.name
private int Laba9.Employee.age
private java.lang.String Laba9.Employee.position

Methods:
private java.lang.String Laba9.Employee.getPosition()
public void Laba9.Employee.work()
private void Laba9.Employee.setPosition(java.lang.String)

Constructors:
public Laba9.Employee()
public Laba9.Employee(java.lang.String,int,java.lang.String)
?аненко ?ван is working as null
Посада: ?нженер
```

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

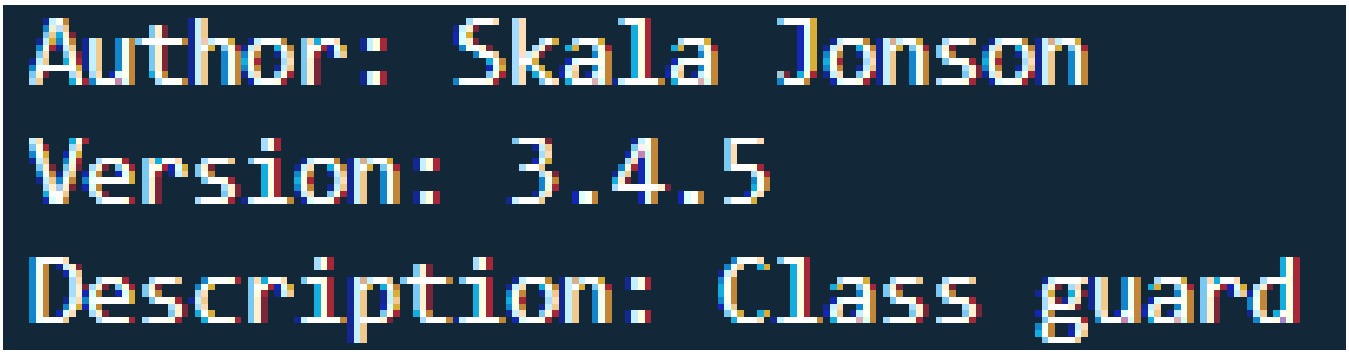
• Встановити їй політику утримання через анотацію @Retention

• Додати анотацію до відповідного об'єкту в коді. • Отримати дані анотації з об'єкту та вивести в консоль.

Вигляд коду:

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@interface Documentation {
    String author() default "Anonymous";
    String version() default "1.0";
    String description() default "No description";
}

@Documentation(
        author = "Skala Jonson",
        version = "3.4.5",
        description = "Class guard"
)
class EmployeeAnnotated {
    private String name;
    private int age;

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

public class Task3 {
  public static void main(String[] args) {
    Class<EmployeeAnnotated> employeeClass = EmployeeAnnotated.class;

    if (employeeClass.isAnnotationPresent(Documentation.class)) {
        Documentation doc = employeeClass.getAnnotation(Documentation.class);

        System.out.println("Author: " + doc.author());
        System.out.println("Version: " + doc.version());
        System.out.println("Description: " + doc.description());
    }
}
}
```

Результат виконання програми:



Висновок: зробов з регулярними виразами, використання рефлексії, створення власних анотацій.