Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №7

Мова програмування Java

Тема: Лямбда вирази в Java.

Виконав
студент групи ІА-32:
Костінський М. М.

Перевірив:
Лесик В. О.

2025

Тема: Лямбда вирази в Java.

Мета: набуття навиків у роботі з лямбда виразами та функціональними інтерфейсами, використання потокової обробки даних (Stream API).

Хід роботи

Теорія:

Лямбда-вирази в Java дозволяють компактно описувати анонімні функції, реалізуючи методи функціональних інтерфейсів, які містять єдиний абстрактний метод. Це значно скорочує обсяг шаблонного коду і дозволяє передавати поведінку як параметр методу. Найпоширеніші функціональні інтерфейси, такі як Predicate, Function і Consumer, вже включені в стандартну бібліотеку для вирішення типових алгоритмічних завдань.

Для ефективної обробки даних використовується Stream API, що представляє послідовності елементів як потік, над яким виконується конвеєр операцій. Ці операції поділяються на проміжні, які виконуються ліниво і лише налаштовують трансформацію, та термінальні, що ініціюють виконання і повертають кінцевий результат. Такий підхід дозволяє реалізовувати складні алгоритми фільтрації, перетворення та агрегації даних у декларативному стилі, уникаючи використання явних циклів.

Завдання:

7. Серед простих чисел, які не перевищують заданий n, знайти таке, в двійковій формі якого максимальна кількість нулів. Просте число – це натуральне число, яке ділиться на 1 та на себе.

Код:

```
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введіть число n:");
        int n = scanner.nextInt();

        IntStream.rangeClosed(2, n)
                .filter(Main::isPrime)
                .boxed()
                .collect(Collectors.groupingBy(
                        Main::countZerosInBinary
                ))
                .entrySet().stream()
```

```
                .max(Map.Entry.comparingByKey())
                .ifPresentOrElse(
                        entry -> {
                            int maxZeros = entry.getKey();
                            List<Integer> primes = entry.getValue();
                            System.out.println("Прості числа з кількістю нулів
(" + maxZeros + "):");
                            primes.forEach(prime ->
                                    System.out.println(prime + " (" +
Integer.toBinaryString(prime) + ")")
                            );
                        },
                        () -> System.out.println("Простих чисел не знайдено")
                );
        scanner.close();
    }

    static boolean isPrime(int num) {
        if (num < 2) return false;
        return IntStream.rangeClosed(2, (int) Math.sqrt(num))
                .noneMatch(i -> num % i == 0);
    }

    static int countZerosInBinary(int num) {
        return (int) Integer.toBinaryString(num)
                .chars()
                .filter(ch -> ch == '0')
                .count();
    }
}
```

Приклад виконання коду:

```
Введіть число n:
100
Прості числа з кількістю нулів (4):
67 (1000011)
73 (1001001)
97 (1100001)
```

Висновок: У ході виконання лабораторної роботи, було успішно перетворено початковий код, який шукає прості числа з найбільшою кількістю нулів у двійковому вигляді. Замість звичайних циклів та умов, завдання було повністю реалізовано за допомогою лямбда-виразів та інструментів Stream API. Це дозволило зробити код значно коротшим і більш зрозумілим