Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

ЛАБОРАТОРНА РОБОТА №12
з дисципліни «Основи програмування - 2»

Тема: «Stream API»

Виконали:
студенти групи ІА-31
Клим’юк В.Л, Самелюк А.С,
Дук М.Д, Сакун Д.С

Перевірив:
асистент кафедри ІСТ
Степанов А. С.

Київ-2024

Тема: Stream API

Мета: Розробити програму, що використовує Stream API для вирішення задач з використанням функціонального стилю програмування.

Хід роботи

1. Пригадати як використовувати Stream API.

2. Виконати завдання 1-3 відповідно до свого варіанту у таблиці 1. Для цього:
   - проаналізувати завдання;
   - створити зазначенні класи та тестові дані;
   - усі списки мають бути типізованими (наприклад, ArrayList<Student>, а не просто ArrayList);
   - завдання слід виконувати використовуючи функціональний стиль програмування (дозволяється використовувати Stream API, лямбда вирази та посилання на методи; заборонено використовувати такі конструкції як if, switch, for, while).

Варіант	Завдання 1	Завдання 2	Завдання 3
1	1.1	2.1	3.1

Завдання №1.
Є список абітурієнтів, які поступають у ВНЗ: (прізвище, кількість балів 0..100). Для простоти вважати, що немає абітурієнтів з однаковим прізвищем, та напівпрохідного балу (ситуація, коли два або більше абітурієнтів мають однакові бали, але зарахований може бути лише один з них через брак вільних місць). Також відома загальна кількість місць на «бюджет» та на «контракт».
Вивести на екран списки прізвищ і балів, відсортовані за абеткою:
1.1. Абітурієнтів, які поступили на «бюджет» (N студентів з найвищими балами, де N – кількість місць на «бюджет», кількість балів >= 60)

Завдання №2.
Є наступні класи:
Інститут (назва, список факультетів)
Факультет (назва, список студентів)
Студент (ім'я, прізвище, номер залікової книжки, середній бал)
Вивести на екран:
2.1. Список усіх студентів інституту (відсортований за абеткою по прізвищам, у разі ідентичності прізвищ – по іменам, а у разі ідентичності імен – за номером залікової книжки)

Завдання №3.
Для абітурієнтів із завдання №1, які не можуть бути зараховані в інститут (кількість балів < 60) отримати список:
3.1. Список абітурієнтів **List<Enrollee>**;

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

class Enrollee {
    String surname;
    int points;

    public Enrollee(String surname, int points) {
        this.surname = surname;
        this.points = points;
    }
}

public class Task_1 {
    public static void main(String[] args) {
        List<Enrollee> enrollees = Arrays.asList(
                new Enrollee("Smith", 80),
                new Enrollee("Johnson", 90),
                new Enrollee("Williams", 70),
                new Enrollee("Brown", 50),
                new Enrollee("Jones", 65)
        );


        List<Enrollee> budgetEnrollees = enrollees.stream()
                .filter(e -> e.points >= 60)
                .sorted(Comparator.comparing(e -> e.surname))
                .toList();

        budgetEnrollees.forEach(e -> System.out.println(e.surname + ": " + e.points));
    }
}
```

Приклад коду 1.1

```java
import java.util.*;
import java.util.stream.Stream;

class Student {
    String firstName;
    String lastName;
    int studentId;
    double averageScore;

    public Student(String firstName, String lastName, int studentId, double
averageScore) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.studentId = studentId;
        this.averageScore = averageScore;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
```

```
        return firstName;
    }

    public int getStudentId() {
        return studentId;
    }
}

class Faculty {
    String name;
    List<Student> students;

    public Faculty(String name, List<Student> students) {
        this.name = name;
        this.students = students;
    }

    public Stream<Student> studentStream() {
        return students.stream();
    }
}

class Institute {
    String name;
    List<Faculty> faculties;

    public Institute(String name, List<Faculty> faculties) {
        this.name = name;
        this.faculties = faculties;
    }

    public Stream<Student> getAllStudents() {
        return faculties.stream()
                .flatMap(Faculty::studentStream);
    }
}

public class Task_2 {
    public static void main(String[] args) {
        List<Student> students = Arrays.asList(
                new Student("John", "Smith", 12345, 75.5),
                new Student("Alice", "Johnson", 23456, 85.0),
                new Student("Bob", "Williams", 34567, 70.25),
                new Student("Emily", "Brown", 45678, 60.75),
                new Student("David", "Jones", 56789, 92.3)
        );

        List<Faculty> faculties = Arrays.asList(
                new Faculty("FICT", Arrays.asList(students.get(0), students.get(1))),
                new Faculty("FAM", Arrays.asList(students.get(2), students.get(3),
students.get(4)))
        );

        Institute institute = new Institute("KPI", faculties);

        institute.getAllStudents()
                .sorted(Comparator.comparing(Student::getLastName)
                        .thenComparing(Student::getFirstName)
                        .thenComparingInt(Student::getStudentId))
                .forEach(s -> System.out.println(s.lastName + " " + s.firstName + " -
ID: " + s.studentId));
    }
}
```

Приклад коду 1.2

```java
import java.util.*;

public class Task_3 {
    public static void main(String[] args) {
        List<Enrollee> enrollees = Arrays.asList(
                new Enrollee("Smith", 80),
                new Enrollee("Johnson", 90),
                new Enrollee("Williams", 70),
                new Enrollee("Brown", 50),
                new Enrollee("Jones", 65)
        );

        List<Enrollee> failedEnrollees = enrollees.stream()
                .filter(e -> e.points < 60)
                .toList();

        failedEnrollees.forEach(e -> System.out.println(e.surname + ": " + e.points));
    }
}
```

Приклад коду 1.3

```
Johnson: 90
Jones: 65
Smith: 80
Williams: 70

Process finished with exit code 0
```

```
Brown Emily - ID: 45678
Johnson Alice - ID: 23456
Jones David - ID: 56789
Smith John - ID: 12345
Williams Bob - ID: 34567

Process finished with exit code 0
```

```
Brown: 50

Process finished with exit code 0
```

Результат виконання програми.

Висновки: Під час виконання лабораторної роботи наша група отримала практичний досвід використання Stream API та функціонального стилю програмування, можна зробити висновок, що цей підхід дозволяє зробити код більш зрозумілим, компактним та ефективним. Stream API надає потужні можливості для маніпулювання потоками даних, таким чином допомагаючи уникнути складних ітераційних конструкцій. Загалом, використання Stream API сприяє покращенню читабельності, підвищенню продуктивності та зниженню кількості помилок в програмному коді.