**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

**ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи №1

**на тему:** “*Робота з колекціями”*

# з дисципліни: *“* Кросплатформне програмування*”*

**Лектор:**

доц. кафедри ПЗ

Дяконюк Л. М.

**Виконав:**

студент групи ПЗ-33

Шлом’як Д. В.

**Прийняв:**

доц. кафедри ПЗ

Баштовий А. В.

Львів – 2024

**Тема.** Робота з колекціями.

**Мета.** Навчитись користуватись колекціями y Java із використанням StreamApi та без.

**Теоретичні відомості**

* **Основні типи колекцій**:
  + Множини (**Set**)
  + Списки (**List**)
  + Черги (**Queue**)
  + Асоційовані масиви (**Map**)
* **Інтерфейси**
  + дозволяють маніпулювати колекціями незалежно від типу реалізації
* **Реалізації** інтерфейсів колекції
  + надаються різноманітні реалізації для колекцій різних типів. Наприклад для колекцій типу List надаються реалізації ArrayList, LinkedList тощо
* **Алгоритми** пошуку, сортування тощо
  + для колекцій
  + для масивів
* **Пакети**
  + Пакет **java.util**
* Пакет **java.util.concurrent**
* Основні методи інтерфейсу **Collection**

add – додавання елементу

addAll – додавання колекції елементів

clear() – очистка колекції

contains(…) – чи містить колекція вказаний елемент

containsAll(…) – чи містить колекція усі вказані елементи

remove(…) – видалити із колекції вказаний елемент

removeAll(…) – видалити із колекції усі вказані елементи

retainAll(…) – залишити в колекції тільки вказані елементи

size() – кількість елементів

isEmpty() – чи порожня колекція

toArray() – перетворення у масив

* contains, containsAll, remove, removeAll, retainAll

При виконанні цих методів вибір елементів колекції - на основі методу **equals** для елементів

**Завдання до лабораторної роботи**

Кожен варіант завдань передбачає використання різних видів колекцій. Покажіть 2 види реалізацій кожного з підзавданням, а саме – з використанням технології Stream.api та без складних операцій з Stream API та безпечну обробку даних через Optional.

**Варіант 3: Управління студентськими гуртожитками**

Опис: У вас є система обліку студентів, що проживають у гуртожитках.

**Кожен студент має:**

* Ім'я (String)
* Прізвище (String)
* Гуртожиток (String)
* Номер кімнати (int)
* Плату за проживання (double)
* Вік (int)

Деякі студенти є пільговиками, і мають право на знижку на проживання.

**Виконайте наступні завдання:**

1. Розділити студентів на тих, хто є пільговиками, і тих, хто не є.
2. Згрупувати студентів за гуртожитками.
3. Створити нову колекцію, в якій для кожної кімнати вказати кількість студентів, що там проживають.
4. Відсортувати студентів за віком та кількістю пільг.
5. Вивести список всіх унікальних номерів кімнат.
6. Знайти студента, який сплачує найбільшу плату за проживання, і обробити відсутність таких студентів через Optional.

**Хід роботи**

***Student.java***

import java.util.Objects;  
  
public class Student {  
 private String firstName;  
 private String lastName;  
 private String dormitory;  
 private int roomNumber;  
 private double fee;  
 private int age;  
 private boolean isDiscounted;  
  
 public Student(String firstName, String lastName, String dormitory, int roomNumber, double fee, int age, boolean isDiscounted) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.dormitory = dormitory;  
 this.roomNumber = roomNumber;  
 this.fee = fee;  
 this.age = age;  
 this.isDiscounted = isDiscounted;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public String getDormitory() {  
 return dormitory;  
 }  
  
 public int getRoomNumber() {  
 return roomNumber;  
 }  
  
 public double getFee() {  
 return fee;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public boolean isDiscounted() {  
 return isDiscounted;  
 }  
  
 @Override  
 public String toString() {  
 return firstName + " " + lastName + ", dormitory: " + dormitory + ", room: " + roomNumber + ", fee: " + fee + ", age: " + age;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Student student = (Student) o;  
 return roomNumber == student.roomNumber && Double.*compare*(student.fee, fee) == 0 &&  
 age == student.age && isDiscounted == student.isDiscounted && Objects.*equals*(firstName, student.firstName) &&  
 Objects.*equals*(lastName, student.lastName) && Objects.*equals*(dormitory, student.dormitory);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(firstName, lastName, dormitory, roomNumber, fee, age, isDiscounted);  
 }  
}

***DormitoryManagement.java***

import java.util.\*;  
import java.util.stream.Collectors;  
  
public class DormitoryManagement {  
  
 public static void main(String[] args) {  
 List<Student> students = Arrays.*asList*(  
 new Student("John", "Doe", "Dorm A", 101, 300.1d, 20, true),  
 new Student("Jane", "Smith", "Dorm B", 102, 150.1d, 22, false),  
 new Student("Mark", "Brown", "Dorm A", 101, 300.1d, 23, true),  
 new Student("Emma", "Wilson", "Dorm C", 103, 250.1d, 19, false),  
 new Student("Emily", "Jones", "Dorm B", 102, 100.1d, 21, false),  
 new Student("Joe", "Shmoe", "Dorm C", 101, 500.1d, 18, false),  
 new Student("Joe", "Mama", "Dorm C", 107, 200.1d, 18, false)  
 );  
  
 // 1. Розділити студентів на тих, хто є пільговиками, і тих, хто не є  
 // Stream API  
 Map<Boolean, List<Student>> partitionedByDiscount = students.stream()  
 .collect(Collectors.*partitioningBy*(Student::isDiscounted));  
 System.*out*.println("Stream API: Students partitioned by discount: ");  
 *OutPartitionedStudents*(partitionedByDiscount);  
  
 // Без Stream API  
 Map<Boolean, List<Student>> partitionedByDiscountManual = new HashMap<>();  
 partitionedByDiscountManual.put(true, new ArrayList<>());  
 partitionedByDiscountManual.put(false, new ArrayList<>());  
 for (Student student : students) {  
 partitionedByDiscountManual.get(student.isDiscounted()).add(student);  
 }  
 System.*out*.println("Manual: Students partitioned by discount: ");  
 *OutPartitionedStudents*(partitionedByDiscountManual);  
  
 // 2. Згрупувати студентів за гуртожитками  
 // Stream API  
 Map<String, List<Student>> groupedByDormitory = students.stream()  
 .collect(Collectors.*groupingBy*(Student::getDormitory));  
 System.*out*.println("Stream API: Students grouped by dormitory: ");  
 *OutGroupedByDormsStudents*(groupedByDormitory);  
  
 // Без Stream API  
 Map<String, List<Student>> groupedByDormitoryManual = new HashMap<>();  
 for (Student student : students) {  
 groupedByDormitoryManual.computeIfAbsent(student.getDormitory(), k -> new ArrayList<>()).add(student);  
 }  
 System.*out*.println("Manual: Students grouped by dormitory: ");  
 *OutGroupedByDormsStudents*(groupedByDormitoryManual);  
  
 // 3. Створити нову колекцію для кількості студентів у кожній кімнаті  
 // Stream API  
 Map<Integer, Long> studentsPerRoom = students.stream()  
 .collect(Collectors.*groupingBy*(Student::getRoomNumber, Collectors.*counting*()));  
 System.*out*.println("Stream API: Students per room: ");  
 *OutStudentsByRoom*(studentsPerRoom);  
  
 // Без Stream API  
 Map<Integer, Long> studentsPerRoomManual = new HashMap<>();  
 for (Student student : students) {  
 studentsPerRoomManual.put(student.getRoomNumber(),  
 studentsPerRoomManual.getOrDefault(student.getRoomNumber(), 0L) + 1L);  
 }  
 System.*out*.println("Manual: Students per room: ");  
 *OutStudentsByRoom*(studentsPerRoomManual);  
  
 // 4. Відсортувати студентів за віком та кількістю пільг  
 // Використання Stream API  
 List<Student> sortedStudents = students.stream()  
 .sorted(Comparator.*comparingInt*(Student::getAge)  
 .thenComparing(Student::isDiscounted))  
 .collect(Collectors.*toList*());  
 System.*out*.println("Stream API: Sorted students: ");  
 *OutSortedStudents*(sortedStudents);  
  
 // Без Stream API  
 List<Student> sortedStudentsManual = new ArrayList<>(students);  
 sortedStudentsManual.sort(Comparator.*comparingInt*(Student::getAge)  
 .thenComparing(Student::isDiscounted));  
 System.*out*.println("Manual: Sorted students: " + sortedStudentsManual);  
 *OutSortedStudents*(sortedStudentsManual);  
  
 // 5. Вивести список унікальних номерів кімнат  
 // Використання Stream API  
 Set<Integer> uniqueRooms = students.stream()  
 .map(Student::getRoomNumber)  
 .collect(Collectors.*toSet*());  
 System.*out*.println("Stream API: Unique room numbers: ");  
 *OutUniqueRooms*(uniqueRooms);  
  
 // Без Stream API  
 Set<Integer> uniqueRoomsManual = new HashSet<>();  
 for (Student student : students) {  
 uniqueRoomsManual.add(student.getRoomNumber());  
 }  
 System.*out*.println("Manual: Unique room numbers: ");  
 *OutUniqueRooms*(uniqueRoomsManual);  
  
 // 6. Знайти студента з найбільшою платою за проживання  
 // Використання Stream API та Optional  
 Optional<Student> maxFeeStudent = students.stream()  
 .max(Comparator.*comparingDouble*(Student::getFee));  
 maxFeeStudent.ifPresentOrElse(  
 student -> System.*out*.println("Stream API: Student with max fee: " + student.getFirstName() + " " +  
 student.getLastName()),  
 () -> System.*out*.println("Stream API: Student with max fee: No students with max fee found"));  
  
 // Без Stream API, з Optional  
 if (!students.isEmpty()) {  
 Student maxFeeStudentManual = students.get(0);  
 for (Student student : students) {  
 if (student.getFee() > maxFeeStudentManual.getFee()) {  
 maxFeeStudentManual = student;  
 }  
 }  
 Optional<Student> optStudent = Optional.*of*(maxFeeStudentManual);  
 optStudent.ifPresent(student -> System.*out*.println("Manual: Student with max fee: " +  
 student.getFirstName() + " " + student.getLastName()));  
 } else {  
 System.*out*.println("Manual: Student with max fee: No students with max fee found");  
 }  
 }  
  
 private static void OutPartitionedStudents(Map<Boolean, List<Student>> partitionedByDiscount) {  
 for (boolean key : partitionedByDiscount.keySet()) {  
 if (key)  
 System.*out*.println("Discountable students:");  
 else  
 System.*out*.println("Non-discountable students:");  
  
 for(Student student : partitionedByDiscount.get(key)){  
 System.*out*.println(student.getFirstName() + " " + student.getLastName());  
 }  
 }  
 System.*out*.println("\n");  
 }  
  
 private static void OutGroupedByDormsStudents(Map<String, List<Student>> groupedByDormitory) {  
 for (String key : groupedByDormitory.keySet()) {  
 System.*out*.println(key + " " + "students:");  
  
 for(Student student : groupedByDormitory.get(key)){  
 System.*out*.println(student.getFirstName() + " " + student.getLastName());  
 }  
 }  
 System.*out*.println("\n");  
 }  
  
 private static void OutStudentsByRoom(Map<Integer, Long> studentsPerRoom) {  
 for (Integer key : studentsPerRoom.keySet()) {  
 System.*out*.println("Students per room " + key + ": " + studentsPerRoom.get(key));  
 }  
 System.*out*.println("\n");  
 }  
  
 private static void OutSortedStudents(List<Student> students) {  
 students.forEach(student -> System.*out*.println(student.getFirstName() + " " + student.getLastName() + ", " +  
 student.getAge() + ", " + student.isDiscounted()));  
 System.*out*.println("\n");  
 }  
  
 private static void OutUniqueRooms(Set<Integer> uniqueRooms){  
 uniqueRooms.forEach(room -> System.*out*.println("Room " + room));  
 System.*out*.println("\n");  
 }  
}

Посилання на проєкт на GitLab - <https://gitlab.com/crossplatformprog/cpp_lab1.git>

**Висновок:** Протягом виконання цієї лабораторної роботи я навчився користуватись колекціями у Java із використанням StreamApi та без. Також дослідив роботу з класом Optional.