**Steps:**

1. **Define the Student class.**
2. **Create an ArrayList<Student> and populate it with student objects.**
3. **Perform various stream operations** like filtering, mapping, and sorting.

**1. Define the Student Class:**

import java.util.Objects;

public class Student {

private String name;

private int age;

private double grade;

public Student(String name, int age, double grade) {

this.name = name;

this.age = age;

this.grade = grade;

}

// Getters

public String getName() {

return name;

}

public int getAge() {

return age;

}

public double getGrade() {

return grade;

}

@Override

public String toString() {

return "Student{name='" + name + "', age=" + age + ", grade=" + grade + '}';

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Student student = (Student) o;

return age == student.age &&

Double.compare(student.grade, grade) == 0 &&

Objects.equals(name, student.name);

}

@Override

public int hashCode() {

return Objects.hash(name, age, grade);

}

}

**2. Create and Populate ArrayList<Student>:**

import java.util.ArrayList;

import java.util.List;

public class StudentStreamExample {

public static void main(String[] args) {

List<Student> students = new ArrayList<>();

students.add(new Student("John", 18, 85.5));

students.add(new Student("Jane", 20, 92.0));

students.add(new Student("Tom", 19, 78.0));

students.add(new Student("Alice", 22, 88.5));

students.add(new Student("Bob", 21, 95.0));

// Example Stream Operations

// 1. Filter students with grade greater than 80

students.stream()

.filter(s -> s.getGrade() > 80)

.forEach(System.out::println);

// 2. Get the average grade of all students

double avgGrade = students.stream()

.mapToDouble(Student::getGrade)

.average()

.orElse(0);

System.out.println("Average Grade: " + avgGrade);

// 3. Sort students by age

students.stream()

.sorted((s1, s2) -> Integer.compare(s1.getAge(), s2.getAge()))

.forEach(System.out::println);

// 4. Get a list of student names using map

students.stream()

.map(Student::getName)

.forEach(System.out::println);

// 5. Check if any student has a grade greater than 90

boolean hasHighGrade = students.stream()

.anyMatch(s -> s.getGrade() > 90);

System.out.println("Has student with grade greater than 90: " + hasHighGrade);

// 6. Find the student with the maximum grade

students.stream()

.max((s1, s2) -> Double.compare(s1.getGrade(), s2.getGrade()))

.ifPresent(s -> System.out.println("Student with max grade: " + s));

}

}

**3. Stream Operations Breakdown:**

* **Filter**: Filters students based on the condition (e.g., grade > 80).
* **MapToDouble**: Computes the average grade using mapToDouble.
* **Sort**: Sorts students by their age.
* **Map**: Transforms the student list into a list of student names.
* **AnyMatch**: Checks if any student has a grade greater than 90.
* **Max**: Finds the student with the maximum grade.

**Output Example:**

Student{name='John', age=18, grade=85.5}

Student{name='Jane', age=20, grade=92.0}

Student{name='Alice', age=22, grade=88.5}

Student{name='Bob', age=21, grade=95.0}

Average Grade: 87.0

Student{name='John', age=18, grade=85.5}

Student{name='Tom', age=19, grade=78.0}

Student{name='Jane', age=20, grade=92.0}

Student{name='Bob', age=21, grade=95.0}

Student{name='Alice', age=22, grade=88.5}

John

Jane

Tom

Alice

Bob

Has student with grade greater than 90: true

Student with max grade: Student{name='Bob', age=21, grade=95.0}