**Java Collections Framework**

The **Java Collections Framework** provides a set of interfaces and classes to store and manipulate groups of objects. This framework simplifies data structure management, providing various implementations for lists, sets, queues, and maps.

**Real-Time Scenario:**

Let’s say you’re developing an application for managing a school system. You need to handle student data, manage courses, and enroll students in courses. The Java Collections Framework offers a variety of data structures for handling these operations efficiently.

Here’s an example illustrating the use of different Java collections, such as List, Set, Map, and how they can be used together in a real-time scenario.

**Example: School Management System**

**Scenario:**

* **List**: To store students in the order they are added.
* **Set**: To store unique courses (since a course should not be duplicated).
* **Map**: To store students and their enrolled courses.

**Explanation of Collection Usage:**

* **ArrayList** (a type of List) is used to store the students in the order they were added, as it maintains **insertion order**.
* **HashSet** (a type of Set) is used to store courses to ensure there are no duplicate course entries, as a Set stores only unique elements.
* **HashMap** (a type of Map) is used to map each student to the courses they are enrolled in. The HashMap provides fast access to the student's courses, and we store the courses in a Set to avoid enrolling a student in the same course multiple times.

**Collections Framework Classes in Use:**

1. **ArrayList**: Implements the List interface and allows duplicates and maintains order.
2. **HashSet**: Implements the Set interface and prevents duplicates.
3. **HashMap**: Implements the Map interface and stores key-value pairs, where each student (key) is associated with their enrolled courses (value).

**Code:**

**import java.util.\*;**

**class Student {**

**private String name;**

**private int id;**

**public Student(String name, int id) {**

**this.name = name;**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public int getId() {**

**return id;**

**}**

**@Override**

**public String toString() {**

**return "Student{id=" + id + ", name='" + name + "'}";**

**}**

**@Override**

**public boolean equals(Object obj) {**

**if (this == obj) return true;**

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

**Student student = (Student) obj;**

**return id == student.id && Objects.equals(name, student.name);**

**}**

**@Override**

**public int hashCode() {**

**return Objects.hash(name, id);**

**}**

**}**

**class Course {**

**private String courseName;**

**public Course(String courseName) {**

**this.courseName = courseName;**

**}**

**public String getCourseName() {**

**return courseName;**

**}**

**@Override**

**public String toString() {**

**return courseName;**

**}**

**@Override**

**public boolean equals(Object obj) {**

**if (this == obj) return true;**

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

**Course course = (Course) obj;**

**return Objects.equals(courseName, course.courseName);**

**}**

**@Override**

**public int hashCode() {**

**return Objects.hash(courseName);**

**}**

**}**

**public class SchoolManagementSystem {**

**public static void main(String[] args) {**

**// List to store students in the order they are added**

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

**students.add(new Student("Alice", 101));**

**students.add(new Student("Bob", 102));**

**students.add(new Student("Charlie", 103));**

**System.out.println("List of Students:");**

**for (Student student : students) {**

**System.out.println(student);**

**}**

**// Set to store unique courses**

**Set<Course> courses = new HashSet<>();**

**courses.add(new Course("Mathematics"));**

**courses.add(new Course("Physics"));**

**courses.add(new Course("Computer Science"));**

**System.out.println("\nSet of Courses (unique):");**

**for (Course course : courses) {**

**System.out.println(course);**

**}**

**// Map to store the students and the courses they are enrolled in**

**Map<Student, Set<Course>> studentCoursesMap = new HashMap<>();**

**// Enrolling students in courses**

**studentCoursesMap.put(students.get(0), new HashSet<>(Arrays.asList(new Course("Mathematics"), new Course("Physics"))));**

**studentCoursesMap.put(students.get(1), new HashSet<>(Arrays.asList(new Course("Physics"), new Course("Computer Science"))));**

**studentCoursesMap.put(students.get(2), new HashSet<>(Collections.singleton(new Course("Computer Science"))));**

**System.out.println("\nStudent Course Enrollment:");**

**for (Map.Entry<Student, Set<Course>> entry : studentCoursesMap.entrySet()) {**

**System.out.println(entry.getKey().getName() + " enrolled in: " + entry.getValue());**

**}**

**}**

**}**

**List of Students:**

**Student{id=101, name='Alice'}**

**Student{id=102, name='Bob'}**

**Student{id=103, name='Charlie'}**

**Set of Courses (unique):**

**Computer Science**

**Mathematics**

**Physics**

**Student Course Enrollment:**

**Alice enrolled in: [Mathematics, Physics]**

**Bob enrolled in: [Computer Science, Physics]**

**Charlie enrolled in: [Computer Science]**

**Explanation of the Code:**

1. **Student and Course Classes**:
   * The Student class holds information about a student (name and ID).
   * The Course class stores the course name.
   * Both classes override equals() and hashCode() to ensure that they work correctly when stored in collections like Set and Map.
2. **List** (ArrayList<Student> students):
   * Stores students in the order they were added.
   * You can access the list by index and maintain the insertion order.
   * Example: Adding students named "Alice," "Bob," and "Charlie" in order.
3. **Set** (HashSet<Course> courses):
   * A Set stores only unique elements, meaning no duplicate courses are allowed.
   * This ensures that even if we try to add the same course multiple times, only one instance of each course will be stored.
4. **Map** (HashMap<Student, Set<Course>> studentCoursesMap):
   * Maps each student to a set of courses they are enrolled in.
   * For each student, we store their enrolled courses in a Set<Course> to avoid duplicate course enrollments.