Core Java-Assessment5

### ****1. Employee Attendance Tracker (Streams, Optional, Date/Time API)****

import java.time.LocalDate;import java.util.\*;import java.util.stream.\*;

class Employee {

String name;

LocalDate attendanceDate;

Employee(String name, LocalDate attendanceDate) {

this.name = name;

this.attendanceDate = attendanceDate;

}

public LocalDate getAttendanceDate() {

return attendanceDate;

}

@Override

public String toString() {

return name + " attended on " + attendanceDate;

}

}

public class EmployeeAttendanceTracker {

public static void main(String[] args) {

List<Employee> employees = Arrays.asList(

new Employee("John", LocalDate.of(2023, 4, 1)),

new Employee("Alice", LocalDate.of(2023, 4, 2)),

new Employee("Bob", LocalDate.of(2023, 4, 3)),

new Employee("David", LocalDate.of(2023, 4, 4))

);

// Using Stream to find employee attendance on a specific date

LocalDate targetDate = LocalDate.of(2023, 4, 3);

Optional<Employee> employee = employees.stream()

.filter(e -> e.getAttendanceDate().equals(targetDate))

.findFirst();

employee.ifPresentOrElse(

e -> System.out.println("Employee found: " + e),

() -> System.out.println("No employee attended on " + targetDate)

);

}

}

### ****2. Movie Streaming Recommendation System (Lambda, Streams, Comparator)****

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

class Movie {

String title;

int year;

double rating;

Movie(String title, int year, double rating) {

this.title = title;

this.year = year;

this.rating = rating;

}

@Override

public String toString() {

return title + " (" + year + ") - Rating: " + rating;

}

}

public class MovieStreamingRecommendationSystem {

public static void main(String[] args) {

List<Movie> movies = Arrays.asList(

new Movie("Movie A", 2021, 4.5),

new Movie("Movie B", 2019, 3.8),

new Movie("Movie C", 2023, 5.0),

new Movie("Movie D", 2020, 4.2)

);

// Using Stream to filter and sort movies based on rating

movies.stream()

.filter(m -> m.rating >= 4.0) // Filter movies with a rating >= 4.0

.sorted(Comparator.comparingDouble(Movie::rating).reversed()) // Sort by rating descending

.forEach(System.out::println); // Print the movies

}

}

### ****5. Online Shopping Order Processing (Streams, Lambda, Date API, Optional)****

import java.time.LocalDate;import java.util.\*;import java.util.stream.\*;

class Order {

String orderId;

double amount;

LocalDate orderDate;

Order(String orderId, double amount, LocalDate orderDate) {

this.orderId = orderId;

this.amount = amount;

this.orderDate = orderDate;

}

@Override

public String toString() {

return "Order{" + "orderId='" + orderId + "', amount=" + amount + ", orderDate=" + orderDate + "}";

}

}

public class OnlineShoppingOrderProcessing {

public static void main(String[] args) {

List<Order> orders = Arrays.asList(

new Order("A101", 500.0, LocalDate.of(2023, 4, 1)),

new Order("A102", 1000.0, LocalDate.of(2023, 4, 3)),

new Order("A103", 250.0, LocalDate.of(2023, 4, 2))

);

// Using Stream to find orders with amount greater than 300

Optional<Order> expensiveOrder = orders.stream()

.filter(o -> o.amount > 300)

.findFirst();

expensiveOrder.ifPresentOrElse(

o -> System.out.println("Expensive Order: " + o),

() -> System.out.println("No expensive orders found.")

);

}

}

窗体顶端

窗体底端