Core Java-Assessment4

### ****1. Employee Management System (List, Set, Map, Comparator)****

import java.util.\*;

class Employee {

int id;

String name;

Employee(int id, String name) {

this.id = id;

this.name = name;

}

@Override

public String toString() {

return "Employee{id=" + id + ", name='" + name + "'}";

}

}

public class EmployeeManagementSystem {

public static void main(String[] args) {

// Using List

List<Employee> employeeList = new ArrayList<>();

employeeList.add(new Employee(1, "John"));

employeeList.add(new Employee(2, "Alice"));

// Using Set (duplicates not allowed)

Set<Employee> employeeSet = new HashSet<>(employeeList);

// Using Map

Map<Integer, Employee> employeeMap = new HashMap<>();

employeeMap.put(1, new Employee(1, "John"));

employeeMap.put(2, new Employee(2, "Alice"));

// Using Comparator for custom sorting by name

employeeList.sort(Comparator.comparing(e -> e.name));

// Display results

System.out.println("Employee List: " + employeeList);

System.out.println("Employee Set (No Duplicates): " + employeeSet);

System.out.println("Employee Map: " + employeeMap);

}

}

### ****2. Flight Ticket Booking System (Queue, PriorityQueue, Comparator)****

import java.util.\*;

class TicketBooking {

int ticketId;

String customerName;

TicketBooking(int ticketId, String customerName) {

this.ticketId = ticketId;

this.customerName = customerName;

}

@Override

public String toString() {

return "TicketBooking{id=" + ticketId + ", customer='" + customerName + "'}";

}

}

public class FlightBookingSystem {

public static void main(String[] args) {

// Queue for normal booking

Queue<TicketBooking> queue = new LinkedList<>();

queue.add(new TicketBooking(1, "John"));

queue.add(new TicketBooking(2, "Alice"));

// PriorityQueue for VIP booking (higher priority)

PriorityQueue<TicketBooking> priorityQueue = new PriorityQueue<>(Comparator.comparingInt(t -> t.ticketId));

priorityQueue.add(new TicketBooking(3, "VIP Customer"));

priorityQueue.add(new TicketBooking(4, "Regular Customer"));

// Display bookings

System.out.println("Queue (First-Come-First-Serve): " + queue);

System.out.println("Priority Queue (VIP Priority): " + priorityQueue);

}

}

### ****3. E-Commerce Product Catalog (TreeMap, HashSet, Comparator)****

import java.util.\*;

class Product {

String name;

double price;

Product(String name, double price) {

this.name = name;

this.price = price;

}

@Override

public String toString() {

return "Product{name='" + name + "', price=" + price + "}";

}

}

public class ECommerceCatalog {

public static void main(String[] args) {

// Using TreeMap (sorted by product name)

Map<String, Product> productCatalog = new TreeMap<>();

productCatalog.put("Laptop", new Product("Laptop", 1000.0));

productCatalog.put("Phone", new Product("Phone", 500.0));

// Using HashSet (no duplicates)

Set<Product> productSet = new HashSet<>();

productSet.add(new Product("Laptop", 1000.0));

productSet.add(new Product("Phone", 500.0));

// Display product catalog

System.out.println("Product Catalog (TreeMap): " + productCatalog);

System.out.println("Unique Products (HashSet): " + productSet);

}

}

### ****4. University Course Enrollment System (LinkedList, Iterator, ListIterator)****

import java.util.\*;

public class UniversityCourseEnrollment {

public static void main(String[] args) {

// Using LinkedList for courses

List<String> courses = new LinkedList<>();

courses.add("Math");

courses.add("Science");

courses.add("History");

// Using Iterator to traverse the list

Iterator<String> iterator = courses.iterator();

while (iterator.hasNext()) {

System.out.println("Course: " + iterator.next());

}

// Using ListIterator to traverse in both directions

ListIterator<String> listIterator = courses.listIterator();

while (listIterator.hasNext()) {

System.out.println("Forward Course: " + listIterator.next());

}

while (listIterator.hasPrevious()) {

System.out.println("Backward Course: " + listIterator.previous());

}

}

}

### ****5. Social Media Following System (HashSet, TreeSet, Iterators)****

import java.util.\*;

public class SocialMediaFollowingSystem {

public static void main(String[] args) {

// Using HashSet for unique followers

Set<String> followers = new HashSet<>();

followers.add("John");

followers.add("Alice");

// Using TreeSet to maintain sorted order

Set<String> sortedFollowers = new TreeSet<>(followers);

// Using Iterator to loop through

Iterator<String> iterator = sortedFollowers.iterator();

while (iterator.hasNext()) {

System.out.println("Follower: " + iterator.next());

}

}

}

### ****6. Student Exam Results Analyzer (HashMap, TreeMap, Utility Classes)****

import java.util.\*;

public class StudentResultsAnalyzer {

public static void main(String[] args) {

// Using HashMap for storing student results

Map<String, Integer> studentResults = new HashMap<>();

studentResults.put("John", 85);

studentResults.put("Alice", 92);

// Using TreeMap to sort students by names

TreeMap<String, Integer> sortedResults = new TreeMap<>(studentResults);

// Display sorted student results

System.out.println("Sorted Student Results: " + sortedResults);

}

}

### ****7. Multi-Language Dictionary (Generics, Bounded Types, TreeMap)****

import java.util.\*;

class Dictionary<K, V> {

private Map<K, V> map = new TreeMap<>();

public void addEntry(K key, V value) {

map.put(key, value);

}

public void printEntries() {

map.forEach((key, value) -> System.out.println(key + ": " + value));

}

}

public class MultiLanguageDictionary {

public static void main(String[] args) {

Dictionary<String, String> englishToFrench = new Dictionary<>();

englishToFrench.addEntry("Hello", "Bonjour");

englishToFrench.addEntry("Goodbye", "Au revoir");

englishToFrench.printEntries();

}

}

### ****8. Online Auction System (Queue, Generics, PriorityQueue, Comparator)****

import java.util.\*;

class Bid {

int amount;

String bidderName;

Bid(int amount, String bidderName) {

this.amount = amount;

this.bidderName = bidderName;

}

@Override

public String toString() {

return "Bid{" + "amount=" + amount + ", bidder='" + bidderName + "'}";

}

}

public class OnlineAuctionSystem {

public static void main(String[] args) {

// Using PriorityQueue for handling highest bid first

PriorityQueue<Bid> auctionQueue = new PriorityQueue<>(Comparator.comparingInt(b -> -b.amount)); // Max heap

auctionQueue.add(new Bid(1000, "John"));

auctionQueue.add(new Bid(1500, "Alice"));

auctionQueue.add(new Bid(1200, "Bob"));

// Display the highest bid

System.out.println("Highest Bid: " + auctionQueue.poll());

}

}