1. **Create an EnumMap to store the number of days in each month of a non-leap year.**

**Print the number of days in each month.**

import java.util.EnumMap;

import java.util.Map;

public class EnumMapExample {

enum Month {

JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER

}

public static void main(String[] args) {

EnumMap<Month, Integer> daysInMonth = new EnumMap<>(Month.class);

daysInMonth.put(Month.JANUARY, 31);

daysInMonth.put(Month.FEBRUARY, 28);

daysInMonth.put(Month.MARCH, 31);

daysInMonth.put(Month.APRIL, 30);

daysInMonth.put(Month.MAY, 31);

daysInMonth.put(Month.JUNE, 30);

daysInMonth.put(Month.JULY, 31);

daysInMonth.put(Month.AUGUST, 31);

daysInMonth.put(Month.SEPTEMBER, 30);

daysInMonth.put(Month.OCTOBER, 31);

daysInMonth.put(Month.NOVEMBER, 30);

daysInMonth.put(Month.DECEMBER, 31);

for (Map.Entry<Month, Integer> entry : daysInMonth.entrySet()) {

System.out.println(entry.getKey() + ": " + entry.getValue() + " days");

}

}

}

1. **Create a LinkedHashMap to store the order of planets in the solar system.**

**Print the order of the planets.**

import java.util.LinkedHashMap;

import java.util.Map;

public class LinkedHashMapExample {

public static void main(String[] args) {

LinkedHashMap<String, Integer> planetOrder = new LinkedHashMap<>();

planetOrder.put("Mercury", 1);

planetOrder.put("Venus", 2);

planetOrder.put("Earth", 3);

planetOrder.put("Mars", 4);

planetOrder.put("Jupiter", 5);

planetOrder.put("Saturn", 6);

planetOrder.put("Uranus", 7);

planetOrder.put("Neptune", 8);

for (Map.Entry<String, Integer> entry : planetOrder.entrySet()) {

System.out.println(entry.getKey() + ": Position " + entry.getValue());

}

}

}

1. **Create a list of Person objects with name and age.**

**Sort the list based on age using a custom comparator.**

**Print the sorted list.**

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

class Person {

private String name;

private int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

}

public class CustomComparatorExample {

public static void main(String[] args) {

List<Person> people = new ArrayList<>();

people.add(new Person("Alice", 30));

people.add(new Person("Bob", 25));

people.add(new Person("Charlie", 40));

// Sort by age using a custom comparator

Collections.sort(people, Comparator.comparingInt(Person::getAge));

for (Person person : people) {

System.out.println(person.getName() + ": " + person.getAge() + " years old");

}

}

}

1. **Read a list of integers from the user.**

**Use a HashSet to remove duplicates from the list.**

**Print the list without duplicates.**

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Scanner;

import java.util.Set;

public class RemoveDuplicatesUsingHashSet {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

List<Integer> numbers = new ArrayList<>();

Set<Integer> uniqueNumbers = new HashSet<>();

System.out.print("Enter the number of elements: ");

int n = scanner.nextInt();

System.out.println("Enter the elements:");

for (int i = 0; i < n; i++) {

int num = scanner.nextInt();

numbers.add(num);

uniqueNumbers.add(num);

}

System.out.println("List without duplicates:");

for (Integer num : uniqueNumbers) {

System.out.print(num + " ");

}

scanner.close();

}

}

1. **Read a list of strings from the user.**

**Use a TreeMap to store the strings as keys with their lengths as values.**

**Print the strings in sorted order of their lengths.**

import java.util.Scanner;

import java.util.TreeMap;

public class TreeMapSortedKeys {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

TreeMap<Integer, String> stringLengths = new TreeMap<>();

System.out.print("Enter the number of strings: ");

int n = scanner.nextInt();

scanner.nextLine(); // Consume the newline

System.out.println("Enter the strings:");

for (int i = 0; i < n; i++) {

String str = scanner.nextLine();

stringLengths.put(str.length(), str);

}

System.out.println("Strings in sorted order of length:");

for (String str : stringLengths.values()) {

System.out.println(str);

}

scanner.close();

}

}

1. **Stack Implementation using ArrayList**

**Implement a basic stack data structure using an ArrayList.**

**Include methods for pushing, popping, and peeking.**

**Test the stack with a sequence of operations.**

import java.util.ArrayList;

import java.util.List;

class Stack<T> {

private List<T> elements = new ArrayList<>();

public void push(T element) {

elements.add(element);

}

public T pop() {

if (isEmpty()) {

throw new IllegalStateException("Stack is empty");

}

return elements.remove(elements.size() - 1);

}

public T peek() {

if (isEmpty()) {

throw new IllegalStateException("Stack is empty");

}

return elements.get(elements.size() - 1);

}

public boolean isEmpty() {

return elements.isEmpty();

}

}

public class StackImplementation {

public static void main(String[] args) {

Stack<Integer> stack = new Stack<>();

stack.push(5);

stack.push(10);

stack.push(15);

System.out.println("Peek: " + stack.peek()); // 15

System.out.println("Pop: " + stack.pop()); // 15

System.out.println("Pop: " + stack.pop()); // 10

System.out.println("Is empty: " + stack.isEmpty()); // false

System.out.println("Pop: " + stack.pop()); // 5

System.out.println("Is empty: " + stack.isEmpty()); // true

}

}

1. **Queue Implementation using LinkedList**

**Implement a basic queue data structure using a LinkedList.**

**Include methods for enqueue, dequeue, and peek.**

**Test the queue with a sequence of operations.**

import java.util.LinkedList;

import java.util.NoSuchElementException;

class Queue<T> {

private LinkedList<T> elements = new LinkedList<>();

public void enqueue(T element) {

elements.addLast(element);

}

public T dequeue() {

if (isEmpty()) {

throw new NoSuchElementException("Queue is empty");

}

return elements.removeFirst();

}

public T peek() {

if (isEmpty()) {

throw new NoSuchElementException("Queue is empty");

}

return elements.getFirst();

}

public boolean isEmpty() {

return elements.isEmpty();

}

}

public class QueueImplementation {

public static void main(String[] args) {

Queue<String> queue = new Queue<>();

queue.enqueue("Alice");

queue.enqueue("Bob");

queue.enqueue("Charlie");

System.out.println("Peek: " + queue.peek()); // Alice

System.out.println("Dequeue: " + queue.dequeue()); // Alice

System.out.println("Dequeue: " + queue.dequeue()); // Bob

System.out.println("Is empty: " + queue.isEmpty()); // false

System.out.println("Dequeue: " + queue.dequeue()); // Charlie

System.out.println("Is empty: " + queue.isEmpty()); // true

}

}

1. **Use a PriorityQueue to implement a to-do list where tasks have priorities.**

**Allow the user to add tasks with priorities and retrieve the highest priority task.**

**Implement a custom comparator to prioritize tasks**.

import java.util.PriorityQueue;

import java.util.Scanner;

class Task {

private String description;

private int priority;

public Task(String description, int priority) {

this.description = description;

this.priority = priority;

}

public String getDescription() {

return description;

}

public int getPriority() {

return priority;

}

}

public class PriorityQueueToDoList {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

PriorityQueue<Task> toDoList = new PriorityQueue<>((task1, task2) -> task2.getPriority() - task1.getPriority());

while (true) {

System.out.print("Enter task description (or 'exit' to quit): ");

String description = scanner.nextLine();

if (description.equalsIgnoreCase("exit")) {

break;

}

System.out.print("Enter priority (1-10): ");

int priority = Integer.parseInt(scanner.nextLine());

toDoList.add(new Task(description, priority));

}

System.out.println("Highest priority task: " + toDoList.poll().getDescription());

scanner.close();

}

}