**Exercise 1: ArrayList Manipulation**

**Create an ArrayList of integers. Perform the following operations:**

**Add numbers from 1 to 10 to the list.**

**Remove all even numbers.**

**Print the contents of the list.**

import java.util.ArrayList;

public class ArrayListExercise {

public static void main(String[] args) {

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

for (int i = 1; i <= 10; i++) {

numbers.add(i);

}

numbers.removeIf(n -> n % 2 == 0);

for (Integer number : numbers) {

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

}

}

}

**Exercise 2: HashMap Operations**

**Create a HashMap to store the names of countries as keys and their capitals as values. Perform the following operations:**

**Add at least five country-capital pairs to the map.**

**Check if a given country is present in the map.**

**Retrieve and print the capital of a specific country.**

**Print all countries and capitals using a loop.**

import java.util.HashMap;

import java.util.Map;

public class HashMapExercise {

public static void main(String[] args) {

Map<String, String> countryCapitalMap = new HashMap<>();

countryCapitalMap.put("USA", "Washington, D.C.");

countryCapitalMap.put("France", "Paris");

countryCapitalMap.put("Germany", "Berlin");

countryCapitalMap.put("Japan", "Tokyo");

countryCapitalMap.put("Brazil", "Brasília");

boolean containsCountry = countryCapitalMap.containsKey("France");

System.out.println("Contains France: " + containsCountry);

String capitalOfGermany = countryCapitalMap.get("Germany");

System.out.println("Capital of Germany: " + capitalOfGermany);

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

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

}

}

}

**Exercise 3: HashSet Deduplication**

**Create a HashSet of strings. Add some duplicate strings and non-duplicate strings to the set.**

**Print the size of the set before and after adding elements.**

**Iterate through the set and print its contents.**

import java.util.HashSet;

import java.util.Set;

public class HashSetExercise {

public static void main(String[] args) {

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

stringSet.add("apple");

stringSet.add("banana");

stringSet.add("cherry");

stringSet.add("apple");

stringSet.add("date");

System.out.println("Size before: " + stringSet.size());

for (String str : stringSet) {

System.out.println(str);

}

System.out.println("Size after: " + stringSet.size());

}

}

**Exercise 4: LinkedList Manipulation**

**Create a LinkedList of strings. Perform the following operations:**

**Add "apple," "banana," and "cherry" to the list.**

**Add "date" at the beginning of the list.**

**Add "grape" at index 2.**

**Remove the last element.**

**Print the contents of the list.**

import java.util.LinkedList;

public class LinkedListExercise {

public static void main(String[] args) {

LinkedList<String> fruits = new LinkedList<>();

fruits.add("apple");

fruits.add("banana");

fruits.add("cherry");

fruits.addFirst("date");

fruits.add(2, "grape");

fruits.removeLast();

for (String fruit : fruits) {

System.out.println(fruit);

}

}

}

**Exercise 5: TreeSet Sorting**

**Create a TreeSet of integers. Add random integers to the set.**

**Print the elements in ascending order.**

**Use the lower() and higher() methods to find the lower and higher elements relative to a given value.**

import java.util.TreeSet;

public class TreeSetExercise {

public static void main(String[] args) {

TreeSet<Integer> numbers = new TreeSet<>();

numbers.add(5);

numbers.add(3);

numbers.add(8);

numbers.add(1);

numbers.add(6);

for (Integer number : numbers) {

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

}

System.out.println("\nLower: " + numbers.lower(5));

System.out.println("Higher: " + numbers.higher(5));

}

}

**Exercise 6: Stack Implementation**

**Implement a basic stack using the Stack class in Java. Perform the following operations:**

**Push elements onto the stack.**

**Pop elements from the stack.**

**Print the elements after each push/pop operation.**

import java.util.Stack;

public class StackExercise {

public static void main(String[] args) {

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

stack.push(10);

stack.push(20);

stack.push(30);

System.out.println(stack);

stack.pop();

System.out.println(stack);

stack.push(40);

System.out.println(stack);

}

}

**Exercise 7: Queue Operations**

**Create a Queue of strings using the LinkedList class. Perform the following operations:**

**Enqueue several elements.**

**Dequeue elements.**

**Print the elements after each enqueue/dequeue operation.**

import java.util.LinkedList;

import java.util.Queue;

public class QueueExercise {

public static void main(String[] args) {

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

queue.offer("apple");

queue.offer("banana");

queue.offer("cherry");

System.out.println(queue);

queue.poll();

System.out.println(queue);

queue.offer("date");

System.out.println(queue);

}

}

**Exercise 8: TreeMap Usage**

**Create a TreeMap that stores names of people as keys and their ages as values. Perform the following operations:**

**Add several name-age pairs to the map.**

**Print the entries in ascending order of ages.**

**import java.util.Map;**

**import java.util.TreeMap;**

**public class TreeMapExercise {**

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

**TreeMap<String, Integer> ageMap = new TreeMap<>();**

**ageMap.put("Alice", 25);**

**ageMap.put("Bob", 30);**

**ageMap.put("Charlie", 28);**

**ageMap.put("David", 22);**

**ageMap.put("Eva", 26);**

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

**System.out.println(entry.getKey() + " - " + entry.getValue());**

**}**

**}**

**}**