ArrayList and LinkedList:

1. Create an ArrayList to store the names of students in a class. Add, remove, and print the list of students.
    - Initialize an empty ArrayList to store examinee names.
    - Add the names of five examinee participating in the exam to the ArrayList.
    - Remove the name of the examinee who withdrew from the exam.
    - Print the updated list of participants.

```java
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> students = new ArrayList<>();
        students.add("a");
        students.add("b");
        students.add("c");
        students.add("d");
        students.add("e");
        students.remove("d");
        System.out.println(students);
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
[a, b, c, e]
[nathan@archlinux Workshop-6]$
```

2. Write a program to insert elements into the linked list at the first and last positions. Also check if the linked list is empty or not.

```java
import java.util.LinkedList;
public class Main {
    public static void main(String[] args) {
        LinkedList<Integer> list = new LinkedList<>();
        System.out.println(list.isEmpty());
        list.addFirst(10);
        list.addLast(20);
        System.out.println(list);
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
true
[10, 20]
[nathan@archlinux Workshop-6]$
```

3. Rotate the elements of an ArrayList to the right by a given number of positions. For example, if the ArrayList is [1, 2, 3, 4, 5] and you rotate it by 2 positions, the result should be [4, 5, 1, 2, 3].

```java
import java.util.ArrayList;
import java.util.Collections;

public class Main {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        Collections.addAll(list, 1, 2, 3, 4, 5);
        int k = 2;
        Collections.rotate(list, k);
        System.out.println(list);
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
[4, 5, 1, 2, 3]
[nathan@archlinux Workshop-6]$
```

4. Write a program to declare a linkedList, colors to store String. Insert five colors into the linked list.
    a. Iterate and print all the colors.
    b. Check if “Red” exists in the linkedList or not.
    c. Shuffle the elements of the list and print them.
    d. Print the LinkedList in ascending order

```java
import java.util.LinkedList;
import java.util.Collections;

public class Main {
    public static void main(String[] args) {
        LinkedList<String> colors = new LinkedList<>();
        Collections.addAll(colors, "Red", "Blue", "Green", "Yellow", "Black");
        for (String color : colors) {
            System.out.println(color);
        }
        System.out.println(colors.contains("Red"));
        Collections.shuffle(colors);
        System.out.println(colors);
        Collections.sort(colors);
        System.out.println(colors);
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
Red
Blue
Green
Yellow
Black
true
[Red, Blue, Green, Yellow, Black]
[Black, Blue, Green, Red, Yellow]
[nathan@archlinux Workshop-6]$
```

Stack:

5. Create a Stack to manage a sequence of tasks. Implement the following operations:
    a. Push the tasks "Read", "Write", and "Code" onto the stack.

b. Pop a task from the stack.

c. Push tasks "Debug" and "Test" onto the stack.

d. Peek at the top task without removing it.

e. Print the stack.

```java
import java.util.Stack;
import java.util.Collections;

public class Main {
    public static void main(String[] args) {
        Stack<String> tasks = new Stack<>();
        Collections.addAll(tasks, "Read", "Write", "Code");
        tasks.pop();
        Collections.addAll(tasks, "Debug", "Test");
        System.out.println(tasks.peek());
        System.out.println(tasks);
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
Test
[Read, Write, Debug, Test]
```

6. Write a program that reverses the order of words in a sentence using a Stack. For example, if the input is "Hello World", the output should be "World Hello".

```java
import java.util.Stack;

public class Main {
    public static void main(String[] args) {
        String sentence = "Hello World";
        Stack<String> stack = new Stack<>();
        for (String word : sentence.split(" ")) stack.push(word);
        while (!stack.isEmpty()) System.out.print(stack.pop() + " ");
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
World Hello [nathan@archlinux Workshop-6]$
```

Queue

7. Imagine a scenario where a printer is managing print jobs. Create a Queue to handle these print jobs. Implement the following operations:
    - Enqueue print jobs "Document1", "Document2", and "Document3" into the print queue.
    - Dequeue a print job from the front of the queue.
    - Enqueue print jobs "Document4" and "Document5" into the print queue.

- Peek at the next print job without removing it.
- Print the list of print jobs in the queue.

```java
import java.util.LinkedList;
import java.util.Queue;

public class Main {
    public static void main(String[] args) {
        Queue<String> printQueue = new LinkedList<>();
        printQueue.add("Document1");
        printQueue.add("Document2");
        printQueue.add("Document3");
        printQueue.poll();
        printQueue.add("Document4");
        printQueue.add("Document5");
        System.out.println(printQueue.peek());
        System.out.println(printQueue);
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
Document2
[Document2, Document3, Document4, Document5]
```

Set Operations

8. Implement a TreeSet to store unique names in alphabetical order.

```java
import java.util.TreeSet;

public class Main {
    public static void main(String[] args) {
        TreeSet<String> names = new TreeSet<>();
        names.add("Coming");
        names.add("Up");
        names.add("With");
        names.add("Names is");
        names.add("Hard");
        System.out.println(names);
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
[Coming, Hard, Names is, Up, With]
[nathan@archlinux Workshop-6]$
```

9. Consider a scenario where you have two sets, each representing a group of animals. Implement a Java program to perform set operations (Union, Intersection, and Difference) on these sets:
    - Initialize two HashSet objects: **set1** with elements "Dog," "Cat," "Elephant," and "Lion," and **set2** with elements "Cat," "Giraffe," "Dog," and "Monkey."
    - Implement a method performUnion that takes two sets and returns their union.
    - Implement a method performIntersection that takes two sets and returns their intersection.

- Implement a method performDifference that takes two sets and returns the difference of the first set from the second set.
- Print the original sets, the union, intersection, and difference of the sets.

```java
import java.util.HashSet;
import java.util.Set;

public class Main {
    public static void main(String[] args) {
        Set<String> set1 = new HashSet<>();
        Set<String> set2 = new HashSet<>();

        set1.addAll(Set.of("Dog", "Cat", "Elephant", "Lion"));
        set2.addAll(Set.of("Cat", "Giraffe", "Dog", "Monkey"));

        System.out.println(set1);
        System.out.println(set2);
        performUnion(set1, set2);
        performIntersection(set1, set2);
        performDifference(set1, set2);
    }

    public static void performUnion(Set<String> set1, Set<String> set2) {
        Set<String> union = new HashSet<>(set1);
        union.addAll(set2);
        System.out.println("Union: " + union);
    }

    public static void performIntersection(Set<String> set1, Set<String> set2) {
        Set<String> intersection = new HashSet<>(set1);
        intersection.retainAll(set2);
        System.out.println("Intersection: " + intersection);
    }

    public static void performDifference(Set<String> set1, Set<String> set2) {
        Set<String> difference = new HashSet<>(set1);
        difference.removeAll(set2);
        System.out.println("Difference (Set1 - Set2): " + difference);
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
[Elephant, Cat, Lion, Dog]
[Monkey, Cat, Dog, Giraffe]
Union: [Elephant, Cat, Monkey, Lion, Dog, Giraffe]
Intersection: [Cat, Dog]
Difference (Set1 - Set2): [Elephant, Lion]
[nathan@archlinux Workshop-6]$
```

Map(HashMap, LinkedHashMap, TreeMap):

10. Write a program that uses a HashMap to store contact information (name and phone number).

```java
import java.util.HashMap;

public class Main{
    public static void main(String[] args) {
        HashMap<String, String> contacts = new HashMap<>();
        contacts.put("A", "000000000");

        for (String name : contacts.keySet()) {
            String phoneNumber = contacts.get(name);
            System.out.println(name + ": " + phoneNumber);
        }
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
A: 000000000
[nathan@archlinux Workshop-6]$
```

11. Imagine a scenario where you are managing information about countries and their capitals using a HashMap. Perform the following tasks:
    - Initialize a HashMap called countryCapitals to store the capitals of different countries. Add at least five country-capital pairs.
    - Implement a method called printMap that takes a HashMap and prints all the key-value pairs.
    - Implement a method called getCapital that takes a country name as a parameter and returns its capital from the countryCapitals map.
    - Implement a method called containsCapital that takes a capital name as a parameter and returns whether that capital exists in the countryCapitals map.
    - Iterate through the countryCapitals map and print each country and its capital.

```java
import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        Map<String, String> countryCapitals = new HashMap<>();

        countryCapitals.put("USA", "Washington D.C.");
        countryCapitals.put("Canada", "Ottawa");

        printMap(countryCapitals);
        System.out.println(getCapital("Japan", countryCapitals));

        System.out.println(containsCapital("Paris", countryCapitals));

        for (Map.Entry<String, String> entry : countryCapitals.entrySet()) {
            System.out.println("Country: " + entry.getKey() + ", Capital: " + entry.getValue());
        }
    }

    // Method to print all key-value pairs in a HashMap
    public static void printMap(Map<String, String> map) {
        System.out.println("Printing map contents:");
        for (Map.Entry<String, String> entry : map.entrySet()) {
            System.out.println("Country: " + entry.getKey() + ", Capital: " + entry.getValue());
        }
    }

    // Method to get the capital of a specific country from the HashMap
    public static String getCapital(String country, Map<String, String> map) {
        return map.get(country);
    }

    // Method to check if a capital exists in the HashMap
    public static boolean containsCapital(String capital, Map<String, String> map) {
        return map.containsValue(capital);
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
Country: Canada, Capital: Ottawa
Country: USA, Capital: Washington D.C.
null
false
Country: Canada, Capital: Ottawa
Country: USA, Capital: Washington D.C.
[nathan@archlinux Workshop-6]$
```

Collection Algorithm

Sorting

12. Write a program that sorts an array of integers using the sort() method. Also try sorting in reverse order.

```java
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        Integer[] numbers = { 5, 2, 9, 1, 5, 6 };

        Arrays.sort(numbers);
        System.out.println("asc order: " + Arrays.toString(numbers));

        for (int i = numbers.length - 1; i >= 0; i--) {
            System.out.print(numbers[i] + " ");
        }
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
asc order: [1, 2, 5, 5, 6, 9]
9 6 5 5 2 1 [nathan@archlinux Workshop-6]$
```

13. Write a program that sorts an array list of strings of colors using the sort() method. Also try sorting in reverse order.

```java
import java.util.Arrays;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        String[] colors = { "Blue", "Red", "Green", "Yellow", "Purple" };
        Arrays.sort(colors);
        System.out.println("asc: " + Arrays.toString(colors));
        System.out.println();
        for (int i = colors.length - 1; i >= 0; i--) {
            System.out.print(colors[i] + " ");
        }
        System.out.println();

    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
asc: [Blue, Green, Purple, Red, Yellow]

Yellow Red Purple Green Blue
[nathan@archlinux Workshop-6]$
```

Binary search

14. Write a program to initialize an ArrayList with a set of integers. Implement a binary search algorithm to find a particular integer.

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        Collections.addAll(numbers, 1, 3, 5, 7, 9, 11, 13, 15);
        Collections.sort(numbers);
        System.out.println("Sorted List: " + numbers);
        int target = 7;
        int index = binarySearch(numbers, target);
        System.out.println(index != -1 ? "Element " + target + " found at index: " + index : "Element " + target + " not found.");
    }

    public static int binarySearch(List<Integer> list, int target) {
        int left = 0, right = list.size() - 1;
        while (left <= right) {
            int mid = (left + right) / 2;
            if (list.get(mid) == target) return mid;
            if (list.get(mid) < target) left = mid + 1;
            else right = mid - 1;
        }
        return -1;
    }
}
```

```
[nathan@archlinux Workshop-6]$ javac Main.java && java Main
Sorted List: [1, 3, 5, 7, 9, 11, 13, 15]
Element 7 found at index: 3
[nathan@archlinux Workshop-6]$
```