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.

   Code:

```java
import java.util.ArrayList;

class students{
    Run | Debug
    public static void main(String[] args) {

        ArrayList<String> name = new ArrayList<>();
        name.add(e:"sushant");
        name.add(e:"rahul");
        name.add(e:"pasang");
        name.add(e:"pritam");
        name.add(e:"sampanna");
        System.out.println(x:"The examinee particip
        System.out.println(name);

        name.remove(o:"sushant");
        System.out.println(name);
    }
}
```

Output:

```
The examinee participater are
[sushant, rahul, pasang, pritam, sampanna]
[rahul, pasang, pritam, sampanna]
```

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.

   Code:

```
import java.util.LinkedList;

public class workshop6 {
    Run | Debug
    public static void main(String[] args) {
        LinkedList<Integer> member=new LinkedList<>();
        member.addFirst(e:20);
        member.addLast(e:30);
        System.out.println(member.isEmpty());
    }
}
```

   Output:

```
false
PS C:\Users\user\OneDrive\Desktop\
```

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].

   Code:

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

public class ArrayListRotation {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<Integer> arrayList = new ArrayList<>();
        arrayList.add(e:1);
        arrayList.add(e:2);
        arrayList.add(e:3);
        arrayList.add(e:4);
        arrayList.add(e:5);

        // Rotate the ArrayList to the right by 2 positi
        rotateArrayList(arrayList, positions:2);

        // Print the rotated ArrayList
        System.out.println("The rotated array list is : 
    }

    public static void rotateArrayList(ArrayList<Integer
        // Calculate the actual number of positions to r
        int actualPositions = positions % arrayList.size
```

```java
        // Rotate the ArrayList
        Collections.rotate(arrayList, actualPositions);
    }
}
```

Output:

```
The rotated array list is : [4, 5, 1, 2, 3]
PS C:\Users\user\OneDrive\Desktop\OOP\workshop6>
```

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

   Code:

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

public class LinkedListOperations {
    Run | Debug
    public static void main(String[] args) {
        LinkedList<String> colors = new LinkedList<
        colors.add(e:"Red");
        colors.add(e:"Green");
        colors.add(e:"Blue");
        colors.add(e:"Yellow");
        colors.add(e:"Orange");

        boolean redExists = colors.contains(o:"Red"
        System.out.println("Red exists in the list:

        Collections.shuffle(colors);
        System.out.println("Shuffled List: " + colo

        Collections.sort(colors);
        System.out.println("Sorted List: " + colors
    }
}
```

Output:

```
Red exists in the list: true
Shuffled List: [Blue, Red, Orange, Yellow, Green]
Sorted List: [Blue, Green, Orange, Red, Yellow]
PS C:\Users\user\OneDrive\Desktop\OOP\workshop6>
```

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.

   Code:

```java
import java.util.Stack;

public class StackOperations {
    Run | Debug
    public static void main(String[] args) {
        Stack<String> tasks = new Stack<>();
        tasks.push(item:"Read");
        tasks.push(item:"Write");
        tasks.push(item:"Code");

        System.out.println("Popped task: " + tasks.p

        tasks.push(item:"Debug");
        tasks.push(item:"Test");

        System.out.println("Top task: " + tasks.peek

        System.out.println("Stack contents: " + task
    }
}
```

Output:

```
Popped task: Code
Top task: Test
Stack contents: [Read, Write, Debug, Test]
PS C:\Users\user\OneDrive\Desktop\OOP\workshop6>
```

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".

   Code:

```java
import java.util.Stack;

public class ReverseSentence {
    Run | Debug
    public static void main(String[] args) {
        String sentence = "Hello World";
        System.out.println("Original Sentence: " + sente

        String reversedSentence = reverseWords(sentence)
        System.out.println("Reversed Sentence: " + rever
    }

    public static String reverseWords(String sentence) {
        String[] words = sentence.split(regex:" ");
        Stack<String> stack = new Stack<>();
        for (String word : words) {
            stack.push(word);
        }

        StringBuilder reversedSentence = new StringBuilc
        while (!stack.isEmpty()) {
            reversedSentence.append(stack.pop()).append(
        }
        return reversedSentence.toString().trim();
    }
}
```

Output:

```
\jdt_ws\jdt.ls-java-project\bin' '
Original Sentence: Hello World
Reversed Sentence: World Hello
```

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.

Code:

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

public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        Queue<String> printQueue = new LinkedList<>();

        printQueue.offer(e:"Document1");
        printQueue.offer(e:"Document2");
        printQueue.offer(e:"Document3");

        String dequeuedJob = printQueue.poll();
        System.out.println("Dequeued job: " + dequeuedJob);

        printQueue.offer(e:"Document4");
        printQueue.offer(e:"Document5");

        String nextJob = printQueue.peek();
        System.out.println("Next print job: " + nextJob);

        System.out.println(x:"Print jobs in the queue:");
        for (String job : printQueue) {
            System.out.println(job);
        }
    }
```

Output:

```
\jdt_ws\jdt.ls-java-project\bin'
Dequeued job: Document1
Next print job: Document2
Print jobs in the queue:
Document2
Document3
Document4
Document5
PS C:\Users\user>
```

Set Operations

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

Code:

```
import java.util.TreeSet;

public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        TreeSet<String> names = new TreeSet<>();

        names.add(e:"Sushant");
        names.add(e:"Sampanna");
        names.add(e:"Gaurav");
        names.add(e:"Rahul");

        System.out.println(x:"Names in alphabetical order
        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

Output:

```
\jdt_ws\jdt.is-java-project\bin
Names in alphabetical order:
Gaurav
Rahul
Sampanna
Sushant
PS C:\Users\user>
```

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.

   Code:

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

public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        Set<String> set1 = new HashSet<>();
        Set<String> set2 = new HashSet<>();

        set1.add(e:"Dog");
        set1.add(e:"Cat");
        set1.add(e:"Elephant");
        set1.add(e:"Lion");

        set2.add(e:"Cat");
        set2.add(e:"Giraffe");
        set2.add(e:"Dog");
        set2.add(e:"Monkey");

        System.out.println("Set 1: " + set1);
        System.out.println("Set 2: " + set2);

        Set<String> union = new HashSet<>(set1);
        union.addAll(set2);
        System.out.println("Union: " + union);

        Set<String> intersection = new HashSet<>(set1
        intersection.retainAll(set2);
        System.out.println("Intersection: " + intersec
```

```
        Set<String> difference = new HashSet<>(set1);
        difference.removeAll(set2);
        System.out.println("Difference (set1 - set2):
    }
}
```

Output:

```
\jdt_ws\jdt.ls-java-project\bin' 'Workshop6'
Set 1: [Cat, Elephant, Lion, Dog]
Set 2: [Cat, Monkey, Dog, Giraffe]
Union: [Cat, Elephant, Monkey, Lion, Dog, Giraffe]
Intersection: [Cat, Dog]
Difference (set1 - set2): [Elephant, Lion]
PS C:\Users\user>
```

Map(HashMap, LinkedHashMap, TreeMap):

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

Code:

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

public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        Map<String, String> contacts = new HashMap<>();

        contacts.put(key:"Rah", value:"98332147562");
        contacts.put(key:"RAm", value:"988754214");
        contacts.put(key:"Prasha", value:"9845213541");

        System.out.println(x:"Contact information:");
        for (Map.Entry<String, String> entry : contacts.e
            System.out.println("Name: " + entry.getKey() +
        }
    }
}

// import java.util.HashMap;
```

Output:

```
Contact information:
Name: Rah, Phone Number: 98332147562
Name: Prasha, Phone Number: 9845213541
Name: RAm, Phone Number: 988754214
PS C:\Users\user>
```

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.

  Code:

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

public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        // Initialize a HashMap to store country-capit
        Map<String, String> countryCapitals = new Hash

        // Add country-capital pairs to the HashMap
        countryCapitals.put(key:"USA", value:"Washingt
        countryCapitals.put(key:"UK", value:"London");
        countryCapitals.put(key:"France", value:"Paris
        countryCapitals.put(key:"Japan", value:"Tokyo"
        countryCapitals.put(key:"India", value:"New De

        // Print all the key-value pairs in the countr
        printMap(countryCapitals);

        // Test the getCapital method
        String capitalOfUK = getCapital(country:"UK",
        System.out.println("Capital of UK: " + capital

        // Test the containsCapital method
        boolean containsLondon = containsCapital(capit
        System.out.println("Does the map contain Londo

        // Iterate through the countryCapitals map and
        System.out.println(x:"Country-Capital pairs:")
        for (Map.Entry<String, String> entry : country
```

```
            System.out.println("Country: " + entry.ge
        }
    }

    // Method to print all key-value pairs in a HashM
    public static <K, V> void printMap(Map<K, V> map)
        System.out.println(x:"Key-Value pairs:");
        for (Map.Entry<K, V> entry : map.entrySet())
            System.out.println("Key: " + entry.getKey
        }
    }

    // Method to get the capital of a country from th
    public static String getCapital(String country, M
        return countryCapitals.get(country);
    }

    // Method to check if a capital exists in the cou
    public static boolean containsCapital(String capi
        return countryCapitals.containsValue(capital)
    }
}
```

Output:

```
PS C:\Users\user> & "C:\Program Files\Java\jdk-21\b
\jdt_ws\jdt.ls-java-project\bin' 'Workshop6'
Key-Value pairs:
Key: USA, Value: Washington, D.C.
Key: UK, Value: London
Key: Japan, Value: Tokyo
Key: France, Value: Paris
Key: India, Value: New Delhi
Capital of UK: London
Does the map contain London? true
Country-Capital pairs:
Country: USA, Capital: Washington, D.C.
Country: UK, Capital: London
Country: Japan, Capital: Tokyo
Country: France, Capital: Paris
Country: India, Capital: New Delhi
PS C:\Users\user>
```

Collection Algorithm

Sorting

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

Code:

```java
import java.util.Arrays;

public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        int[] numbers = {5, 2, 9, 1, 7};

        Arrays.sort(numbers);
        System.out.println("Sorted array in ascending ord

        for (int i = 0; i < numbers.length / 2; i++) {
            int temp = numbers[i];
            numbers[i] = numbers[numbers.length - 1 - i];
            numbers[numbers.length - 1 - i] = temp;
        }
        System.out.println("Sorted array in descending or
    }
}
```

Output:

```
\jdt_ws\jdt.ls-java-project\bin' 'Workshop6'
Sorted array in ascending order: [1, 2, 5, 7, 9]
Sorted array in descending order: [9, 7, 5, 2, 1]
```

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

Code:

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

public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<String> colors = new ArrayList<>();
        colors.add(e:"Red");
        colors.add(e:"Blue");
        colors.add(e:"Green");
        colors.add(e:"Yellow");
        colors.add(e:"Orange");

        Collections.sort(colors);
        System.out.println("Sorted colors in ascending

        Collections.sort(colors, Collections.reverseOrd
        System.out.println("Sorted colors in descending
    }
}
```

Output:

```
\jdt_ws\jdt.ls-java-project\bin' 'Workshop6'
Sorted colors in ascending order: [Blue, Green, Orange, Red,
Sorted colors in descending order: [Yellow, Red, Orange, Gree
PS C:\Users\user>
```

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.

Code:

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

public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        Collections.addAll(numbers, ...elements:10, 20, 3

        int target = 30;
        int index = Collections.binarySearch(numbers, tar

        if (index >= 0) {
            System.out.println("Integer " + target + " fo
        } else {
            System.out.println("Integer " + target + " no
        }
    }
}
```

Output:

```
PS C:\Users\user>  &  C:\Prograi
ws_d62c9\jdt_ws\jdt.ls-java-proj
Integer 30 found at index 2
```

Regular Expressions

15. Write a Java program to check whether a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

Code:

```
public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        String s1 = "Hello123";
        String s2 = "Hello@123";

        boolean res1 = containsOnlyAlphaNumeric(s1);
        boolean res2 = containsOnlyAlphaNumeric(s2);

        System.out.println(s1 + " contains only alphanu
        System.out.println(s2 + " contains only alphanu
    }

    public static boolean containsOnlyAlphaNumeric(Stri
        return str.matches(regex:"[a-zA-Z0-9]+");
    }
}
```

Output:

```
ws_d62c9\jdt_ws\jdt.ls-java-project\bin' 'Workshop6'
Hello123 contains only alphanumeric characters: true
Hello@123 contains only alphanumeric characters: false
PS C:\Users\user>
```

16. Write a Java program to find the sequence of one upper case letter followed by lower case letters. Z

Code:

```
public class Workshop6 {
    Run | Debug
    public static void main(String[] args) {
        String input = "SusHantSaMpAnnaRahuL";

        findSequence(input);
    }

    public static void findSequence(String str) {
        for (int i = 0; i < str.length() - 1; i++) {
            char currentChar = str.charAt(i);
            char nextChar = str.charAt(i + 1);

            if (Character.isUpperCase(currentChar) && Cha
                System.out.println("Sequence found: " + c
            }
        }
    }
}
```

Output:

```
ws_d62c9\jdt_ws\jdt.ls-java-project\bin' 'Workshop6
Sequence found: Su
Sequence found: Ha
Sequence found: Sa
Sequence found: Mp
Sequence found: An
Sequence found: Ra
PS C:\Users\user>
```

17. Develop a Java program to check if a given string represents a file with a ".txt" extension.

Code:

```
public class Workshop6 {
    public static void main(String[] args) {
        Run | Debug
        String fileName1 = "document.txt";
        String fileName2 = "image.jpg";

        System.out.println(fileName1 + " represents a .tx
        System.out.println(fileName2 + " represents a .tx
    }

    public static boolean isTxtFile(String fileName) {
        return fileName.endsWith(".txt");
    }
}
```

Output:

```
\vscodesws_d88ca\jdt_ws\jdt.ls-java-project\b
document.txt represents a .txt file: true
image.jpg represents a .txt file: false
PS C:\Users\user>
```