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 {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<String> studentList = new ArrayList<>();
        studentList.add(e:"Santosh");
        studentList.add(e:"Sumit");
        studentList.add(e:"Arayan");
        studentList.add(e:"Rajni");
        studentList.add(e:"Barsigh");

        System.out.println("Initial List : " + studentList);

        String withdrawnStudent = "Arayan";
        studentList.remove(withdrawnStudent);

        System.out.println("Updated List : " + studentList);
    }
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Updated List : [Santosh, Sumit, Rajni, Barsigh]
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Document
s\OOPS\Week-9\Workshop\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Initial List : [Santosh, Sumit, Arayan, Rajni, Barsigh]
Updated List : [Santosh, Sumit, Rajni, Barsigh]
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

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.

```
Run | Debug
public static void main(String[] args) {
LinkedList<String> list = new LinkedList<String>();
    list.add(e:"SUMIT");
    list.add(e:"SANTOSH");
    list.add(e:"RAJNI");

    Iterator itr = list.iterator();
    while (itr.hasNext()) {
        System.out.println(itr.next());

    }
    list.addFirst(e:"Rukesh");
    list.addLast(e:"Keshav");

    System.out.println(x:"Upadted List : ");
    Iterator<String> upIterator = list.iterator();
    while (upIterator.hasNext()) {
        System.out.println(upIterator.next());
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
java Main }
SUMIT
SANTOSH
RAJNI
Upadted List :
Rukesh
SUMIT
SANTOSH
RAJNI
Keshav
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

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
    // 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].
    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);

        int rotateBy = 2;

        System.out.println("Original List : " + arrayList);


        Collections.rotate(arrayList, rotateBy);

        System.out.println("Rotated List : " + arrayList);


    }
}
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Document
s\OOPS\Week-9\Workshop\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Original List : [1, 2, 3, 4, 5]
Rotated List : [4, 5, 1, 2, 3]
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

5. 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;
import java.util.Iterator;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        // Create a LinkedList to store colors
        LinkedList<String> colors = new LinkedList<>();

        // Insert five colors into the linked list
        colors.add(e:"Red");
        colors.add(e:"Blue");
        colors.add(e:"Green");
        colors.add(e:"Yellow");
        colors.add(e:"Orange");

        // a. Iterate and print all the colors
        System.out.println(x:"All colors:");
        for (String color : colors) {
            System.out.println(color);
        }
```

```java
    // a. Iterate and print all the colors
    System.out.println(x:"All colors:");
    for (String color : colors) {
        System.out.println(color);
    }

    // b. Check if "Red" exists in the linked list or not
    if (colors.contains(o:"Red")) {
        System.out.println(x:"Red exists in the linked list.");
    } else {
        System.out.println(x:"Red does not exist in the linked list.");
    }

    // c. Shuffle the elements of the list and print them
    Collections.shuffle(colors);
    System.out.println(x:"\nShuffled colors:");
    for (String color : colors) {
        System.out.println(color);
    }
```

```java
        // d. Print the LinkedList in ascending order
        Collections.sort(colors);
        System.out.println(x:"\nColors in ascending order:");
        for (String color : colors) {
            System.out.println(color);
        }
    }
}
```

Stack:

6. 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;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        // Create a Stack to manage tasks
        Stack<String> tasks = new Stack<>();

        // a. Push the tasks "Read", "Write", and "Code" onto the stack
        tasks.push(item:"Read");
        tasks.push(item:"Write");
        tasks.push(item:"Code");

        // b. Pop a task from the stack
        String poppedTask = tasks.pop();
        System.out.println("Popped task: " + poppedTask);

        // c. Push tasks "Debug" and "Test" onto the stack
        tasks.push(item:"Debug");
        tasks.push(item:"Test");
```

```java
        // c. Push tasks "Debug" and "Test" onto the stack
        tasks.push(item:"Debug");
        tasks.push(item:"Test");

        // d. Peek at the top task without removing it
        String topTask = tasks.peek();
        System.out.println("Top task: " + topTask);

        // e. Print the stack
        System.out.println(x:"\nTasks on the stack:");
        for (String task : tasks) {
            System.out.println(task);
        }
    }
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Tasks on the stack:
Read
Write
Debug
Test
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

7. 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
// 7.   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".
import java.util.Stack;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        String input = "Hello World";
        String reversedSentence = reverseSentence(input);
        System.out.println("Reversed sentence: " + reversedSentence);
    }

    public static String reverseSentence(String sentence) {
        // Split the sentence into words
        String[] words = sentence.split(regex:" ");

        // Create a stack to store the words
        Stack<String> wordStack = new Stack<>();
        for (String word : words) {
            wordStack.push(word);
        }

        // Reconstruct the sentence by popping words from the stack
        StringBuilder reversedSentence = new StringBuilder();
        while (!wordStack.isEmpty()) {
            reversedSentence.append(wordStack.pop()).append(str:" ");
        }
```

```java
        // Create a stack to store the words
        Stack<String> wordStack = new Stack<>();
        for (String word : words) {
            wordStack.push(word);
        }

        // Reconstruct the sentence by popping words from the stack
        StringBuilder reversedSentence = new StringBuilder();
        while (!wordStack.isEmpty()) {
            reversedSentence.append(wordStack.pop()).append(str:" ");
        }

        // Remove trailing space and return the reversed sentence
        return reversedSentence.toString().trim();
    }
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Reversed sentence: World Hello
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

Queue

8. 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 {
    Run | Debug
    public static void main(String[] args) {
        // Create a Queue to manage print jobs
        Queue<String> printQueue = new LinkedList<>();

        // Enqueue print jobs "Document1", "Document2", and "Document3" into the print queue
        printQueue.add(e:"Document1");
        printQueue.add(e:"Document2");
        printQueue.add(e:"Document3");

        // Dequeue a print job from the front of the queue
        String dequeuedJob = printQueue.poll();
        System.out.println("Dequeued job: " + dequeuedJob);

        // Enqueue print jobs "Document4" and "Document5" into the print queue
        printQueue.add(e:"Document4");
        printQueue.add(e:"Document5");

        // Peek at the next print job without removing it
        String nextJob = printQueue.peek();
        System.out.println("Next job: " + nextJob);

        // Print the list of print jobs in the queue
        System.out.println(x:"\nPrint jobs in the queue:");
        for (String job : printQueue) {
            System.out.println(job);
        }
    }
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Dequeued job: Document1
Next job: Document2

Print jobs in the queue:
Document2
Document3
Document4
Document5
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

Set Operations

9. Implement a TreeSet to store unique names in alphabetical order.
10. 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 {
    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("Original set1: " + set1);
        System.out.println("Original set2: " + set2);

        Set<String> union = performUnion(set1, set2);
        System.out.println("Union: " + union);

        Set<String> intersection = performIntersection(set1, set2);
        System.out.println("Intersection: " + intersection);

        Set<String> difference = performDifference(set1, set2);
        System.out.println("Difference (set1 - set2): " + difference);
    }
```

```java
    public static <T> Set<T> performUnion(Set<T> set1, Set<T> set2) {
        Set<T> union = new HashSet<>(set1);
        union.addAll(set2);
        return union;
    }

    public static <T> Set<T> performIntersection(Set<T> set1, Set<T> set2) {
        Set<T> intersection = new HashSet<>(set1);
        intersection.retainAll(set2);
        return intersection;
    }

    public static <T> Set<T> performDifference(Set<T> set1, Set<T> set2) {
        Set<T> difference = new HashSet<>(set1);
        difference.removeAll(set2);
        return difference;
    }
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Original set1: [Cat, Elephant, Lion, Dog]
Original set2: [Cat, Monkey, Dog, Giraffe]
Union: [Cat, Elephant, Monkey, Lion, Dog, Giraffe]
Intersection: [Cat, Dog]
Difference (set1 - set2): [Elephant, Lion]
```

Map(HashMap, LinkedHashMap, TreeMap):

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

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

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Map<String, String> contacts = new HashMap<>();
        contacts.put(key:"Sumit Shah", value:"9818257300");
        contacts.put(key:"Rajni Shah", value:"9861446442");
        contacts.put(key:"Santosh Shah", value:"9818257400");

        System.out.println(x:"Contact Information:");
        contacts.forEach((name, phoneNumber) -> System.out.println("Name: " + name + ", Phone Number: " + phoneNumber));

        String sumitPhoneNumber = contacts.get(key:"Sumit Shah");
        System.out.println("\nPhone number for Sumit Shah: " + sumitPhoneNumber);

        String nameToCheck = "Rajni Shah";
        System.out.println((contacts.containsKey(nameToCheck) ? nameToCheck + " exists in contacts." : nameToCheck + " does not exist in contacts
    }
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

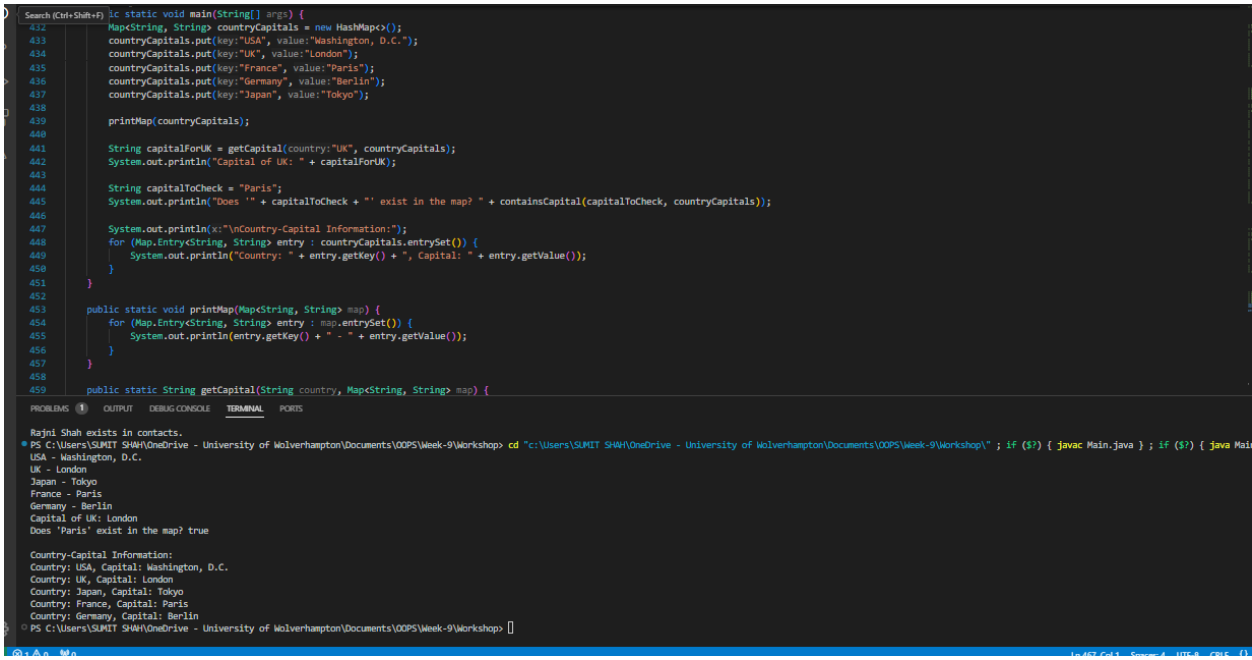
```
Name: Rajni Shah, Phone Number: 9861446442
Name: Sumit Shah, Phone Number: 9818257300
Name: Santosh Shah, Phone Number: 9818257400

Phone number for Sumit Shah: 9818257300
Rajni Shah exists in contacts.
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

12. 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.



```
Search (Ctrl+Shift+F) ic static void main(String[] args) {
        Map<String, String> countryCapitals = new HashMap<>();
        countryCapitals.put(key:"USA", value:"Washington, D.C.");
        countryCapitals.put(key:"UK", value:"London");
        countryCapitals.put(key:"France", value:"Paris");
        countryCapitals.put(key:"Germany", value:"Berlin");
        countryCapitals.put(key:"Japan", value:"Tokyo");

        printMap(countryCapitals);

        String capitalForUK = getCapital(country:"UK", countryCapitals);
        System.out.println("Capital of UK: " + capitalForUK);

        String capitalToCheck = "Paris";
        System.out.println("Does '" + capitalToCheck + "' exist in the map? " + containsCapital(capitalToCheck, countryCapitals));

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

    public static void printMap(Map<String, String> map) {
        for (Map.Entry<String, String> entry : map.entrySet()) {
            System.out.println(entry.getKey() + " - " + entry.getValue());
        }
    }

    public static String getCapital(String country, Map<String, String> map) {
```

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Rajni Shah exists in contacts.
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop\" ; if ($?) { javac Main.java } ; if ($?) { java Mai
USA - Washington, D.C.
UK - London
Japan - Tokyo
France - Paris
Germany - Berlin
Capital of UK: London
Does 'Paris' exist in the map? true

Country-Capital Information:
Country: USA, Capital: Washington, D.C.
Country: UK, Capital: London
Country: Japan, Capital: Tokyo
Country: France, Capital: Paris
Country: Germany, Capital: Berlin
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

Collection Algorithm

Sorting

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

Workshop > Main.java > Main > printArray(int[])

```java
import java.util.Arrays;
import java.util.Comparator;

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

        Arrays.sort(numbers);
        printArray(numbers);

        Integer[] integerNumbers = Arrays.stream(numbers).boxed().toArray(Integer[]::new);
        Arrays.sort(integerNumbers, Comparator.reverseOrder());
        for (int i = 0; i < numbers.length; i++) {
            numbers[i] = integerNumbers[i];
        }
        printArray(numbers);
    }

    public static void printArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
1 2 5 7 9
9 7 5 2 1
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```

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

```java
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(x:"Sorted in ascending order:");
        printList(colors);

        Collections.sort(colors, Collections.reverseOrder());
        System.out.println(x:"\nSorted in descending order:");
        printList(colors);
    }

    public static void printList(ArrayList<String> list) {
        for (String color : list) {
            System.out.print(color + " ");
        }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Sorted in descending order:
Yellow Red Orange Green Blue
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Document
s\OOPS\Week-9\Workshop\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Sorted in ascending order:
Blue Green Orange Red Yellow
```

Binary search

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

```java
// 15.  Write a program to initialize an ArrayList with a set of integers. Implement a binary search algorithm to find a particular integer.
import java.util.ArrayList;
import java.util.Collections;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        Collections.addAll(numbers, ...elements:2, 5, 8, 12, 16, 23, 38, 56, 72, 91);

        int target = 23;
        int index = binarySearch(numbers, target);

        if (index != -1) {
            System.out.println(target + " found at index " + index);
        } else {
            System.out.println(target + " not found in the list");
        }
    }

    public static int binarySearch(ArrayList<Integer> arr, int target) {
        int left = 0, right = arr.size() - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr.get(mid) == target) return mid;
            if (arr.get(mid) < target) left = mid + 1;
            else right = mid - 1;
        }
        return -1;
    }
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
s\OOPS\Week-9\Workshop\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
23 found at index 5
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\Week-9\Workshop>
```