DAY-7
Quiz-1

1. Create Account A and Account B with an initial balance of 5000 and 2500 respectively. Transfer amount of 1500 from Account A to B and an amount of 3000 from Account B to A.

Print the receipt with the following details after each transaction

Output:
Account id: 12344,
Name: XXXX,
Account Balance: Rs.___

CODE:

```
public class BankAccount {
    private int accountId;
    private String name;
    private double balance;
    public BankAccount(int accountId, String name, double balance) {
        this.accountId = accountId;
        this.name = name;
        this.balance = balance;
    }
    public boolean transfer(BankAccount otherAccount, double amount) {
        if (balance >= amount) {
            balance -= amount;
            otherAccount.balance += amount;
            return true;
        } else {
            System.out.println("Insufficient funds in Account " + accountId + " to transfer Rs. " +
amount);
            return false;
        }
    }
```

```
    public void printReceipt() {
        System.out.println("Account id: " + accountId + ",");
        System.out.println("Name: " + name + ",");
        System.out.println("Account Balance: Rs. " + balance + "\n");
    }
    public static void main(String[] args) {
        BankAccount accountA = new BankAccount(12344, "Account A", 5000);
        BankAccount accountB = new BankAccount(56789, "Account B", 2500);
        if (accountA.transfer(accountB, 1500)) {
            System.out.println("Receipt after transferring Rs. 1500 from Account A to B:");
            accountA.printReceipt();
            accountB.printReceipt();
        }
        if (accountB.transfer(accountA, 3000)) {
            System.out.println("Receipt after transferring Rs. 3000 from Account B to A:");
            accountA.printReceipt();
            accountB.printReceipt();
        }
    }
}
```

OUTPUT:

```
Receipt after transferring Rs. 1500 from Account A to B:
Account id: 12344,
Name: Account A,
Account Balance: Rs. 3500.0

Account id: 56789,Name: Account B,
Account Balance: Rs. 4000.0

Receipt after transferring Rs. 3000 from Account B to A:
Account id: 12344,
Name: Account A,
Account Balance: Rs. 6500.0

Account id: 56789,
Name: Account B,
Account Balance: Rs. 1000.0
```

2. Given an array and a partition size, you have to partition the array with that value , then we will specify the partition order, you have to merge based on that order

Input:
Array : 1 2 3 4 5
Partition size 2 (so the array will be partitioned as **1 2**, 3 4, **5**)
Partition order 3 2 1

Output:
5 3 4 1 2

CODE:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
public class ArrayPartition {
  public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the elements of the array separated by spaces: ");
      String[] arrayInput = scanner.nextLine().split("\\s+");
      int[] array = new int[arrayInput.length];
      for (int i = 0; i < arrayInput.length; i++) {
         array[i] = Integer.parseInt(arrayInput[i]);
      }
      System.out.print("Enter the partition size: ");
      int partitionSize = scanner.nextInt();
      scanner.nextLine();
      System.out.print("Enter the partition order separated by spaces: ");
      String[] orderInput = scanner.nextLine().trim().split("\\s+");
      int[] partitionOrder = Arrays.stream(orderInput).mapToInt(Integer::parseInt).toArray();
      int[] result = mergeArrayWithPartitionOrder(array, partitionSize, partitionOrder);
      System.out.println("Merged Array with Partition Order: " + Arrays.toString(result));
  }
   public static int[] mergeArrayWithPartitionOrder(int[] array, int partitionSize, int[]
partitionOrder) {
      List<List<Integer>> partitions = new ArrayList<>();
      for (int i = 0; i < array.length; i += partitionSize) {
         int end = Math.min(i + partitionSize, array.length);
         List<Integer> partitionList = new ArrayList<>();
         for (int j = i; j < end; j++) {
            partitionList.add(array[j]);
         }
         partitions.add(partitionList);
      }
      Collections.sort(partitions, (a, b) -> {
         int minSize = Math.min(a.size(), b.size());
         for (int i = 0; i < minSize; i++) {
```

```
            int cmp = Integer.compare(partitionOrder[i], partitionOrder[i]);
            if (cmp != 0) {
                return cmp;
            }
        }
        return Integer.compare(a.size(), b.size());
    });
    List<Integer> mergedList = new ArrayList<>();
    for (List<Integer> partition : partitions) {
        mergedList.addAll(partition);
    }
    int[] result = mergedList.stream().mapToInt(Integer::intValue).toArray();
    return result;
  }
}
```

OUTPUT:

```
Enter the elements of the array separated by spaces: 1 2 3 4 5
Enter the partition size: 2
Enter the partition order separated by spaces: 3 2 1
Merged Array with Partition Order: [5, 1, 2, 3, 4]
```

3. A palindrome number - number that remains the same after reversing each digit of that number. A prime number - number that is divisible by only one or itself. A number that satisfies both the properties is said to be PalPrime Number.

Create a class PalPrime with a parameterised constructor PalPrime(int number, String message).

Given an positive integer array of numbers, you have to traverse the array and print the message “Number ___ is Prime/Palindrome/PalPrime”.

Note: Message should be printed via constructor of PalPrime class.

Input :
Array: [1, 34543, 565, 727, 10099]

Output -> Predict the output

CODE:

```
import java.util.Arrays;
class PalPrime {
  public PalPrime(int number, String message) {
    System.out.println("Number " + number + " is " + message);
  }
```

```
    public static void main(String[] args) {
        int[] numbers = {1, 34543, 565, 727, 10099};
        for (int number : numbers) {
            checkAndPrintMessage(number);
        }
    }
    private static void checkAndPrintMessage(int number) {
        String message = "";
        if (isPalindrome(number)) {
            message += "Palindrome";
        }
        if (isPrime(number)) {
            if (!message.isEmpty()) {
                message += "/";
            }
            message += "Prime";
        }
        new PalPrime(number, message.isEmpty() ? "Neither Prime nor Palindrome" :
message);
    }

    private static boolean isPalindrome(int number) {
        int originalNumber = number;
        int reversedNumber = 0;

        while (number > 0) {
            int digit = number % 10;
            reversedNumber = reversedNumber * 10 + digit;
            number /= 10;
        }

        return originalNumber == reversedNumber;
    }

    private static boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }

        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

OUTPUT:

```
Number 1 is Palindrome
Number 34543 is Palindrome/Prime
Number 565 is Palindrome
Number 727 is Palindrome/Prime
Number 10099 is Prime
```