Program:1

```
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 void deposit(double amount) {
      balance += amount;
   }
   public void withdraw(double amount) {
      if (balance >= amount) {
         balance -= amount;
      } else {
         System.out.println("Insufficient funds!");
      }
   }
   public void printReceipt() {
      System.out.println("Account id: " + accountId);
      System.out.println("Name: " + name);
      System.out.println("Account Balance: Rs." + balance);
      System.out.println("--------------------------");
   }
   public static void main(String[] args) {
      BankAccount accountA = new BankAccount(12344, "Account A", 5000);
      BankAccount accountB = new BankAccount(12345, "Account B", 2500);
      accountA.withdraw(1500);
      accountB.deposit(1500);
```

```
        System.out.println("Transfer from Account A to B completed.");
        accountA.printReceipt();
        accountB.printReceipt();
        accountB.withdraw(3000);
        accountA.deposit(3000);
        System.out.println("Transfer from Account B to A completed.");
        accountA.printReceipt();
        accountB.printReceipt();
    }
}
```

Program:2

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
public class ArrayPartitionAndMerge {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Array: ");
        String[] arrayInput = scanner.nextLine().split(" ");
        int[] array = Arrays.stream(arrayInput).mapToInt(Integer::parseInt).toArray();
        System.out.print("Partition size: ");
        int partitionSize = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Partition order: ");
        String[] orderInput = scanner.nextLine().split(" ");
        int[] partitionOrder = Arrays.stream(orderInput).mapToInt(Integer::parseInt).toArray();
        List<int[]> partitions = partitionArray(array, partitionSize);
        int[] mergedArray = mergePartitions(partitions, partitionOrder);
        System.out.println("Output:");
```

```
        for (int num : mergedArray) {
            System.out.print(num + " ");
        }
    }
    private static List<int[]> partitionArray(int[] array, int partitionSize) {
        List<int[]> partitions = new ArrayList<>();
        for (int i = 0; i < array.length; i += partitionSize) {
            int[] partition = Arrays.copyOfRange(array, i, Math.min(i + partitionSize, array.length));
            partitions.add(partition);
        }
        return partitions;
    }
    private static int[] mergePartitions(List<int[]> partitions, int[] partitionOrder) {
        List<Integer> mergedList = new ArrayList<>();
        List<Integer> partitionIndices = new ArrayList<>();
        for (int i = 0; i < partitions.size(); i++) {
            partitionIndices.add(i);
        }
        if (partitionOrder.length != partitions.size()) {
            throw new IllegalArgumentException("Partition order length does not match the number of
partitions");
        }
        Collections.sort(partitionIndices, (a, b) -> Integer.compare(partitionOrder[a], partitionOrder[b]));
        for (int index : partitionIndices) {
            int[] partition = partitions.get(index);
            for (int num : partition) {
                mergedList.add(num);
            }
        }
        int[] mergedArray = new int[mergedList.size()];
        for (int i = 0; i < mergedList.size(); i++) {
```

```
            mergedArray[i] = mergedList.get(i);
        }
        return mergedArray;
    }
}
```

Program:3

```
public class PalPrime {
    private int number;
    private String message;
    public PalPrime(int number, String message) {
        this.number = number;
        this.message = message;
        System.out.println("Number " + number + " is " + message);
    }
    public static void main(String[] args) {
        int[] numbers = {1, 34543, 565, 727, 10099};
        for (int num : numbers) {
            if (isPalPrime(num)) {
                new PalPrime(num, "PalPrime");
            } else if (isPrime(num)) {
                new PalPrime(num, "Prime");
            } else if (isPalindrome(num)) {
                new PalPrime(num, "Palindrome");
            } else {
                System.out.println("Number " + num + " is neither Prime nor Palindrome.");
            }
        }
    }
    public static boolean isPrime(int num) {
        if (num <= 1) {
            return false;
```

```
        }
        for (int i = 2; i * i <= num; i++) {
            if (num % i == 0) {
                return false;
            }
        }
        return true;
    }
    public static boolean isPalindrome(int num) {
        int originalNum = num;
        int reverse = 0;
        while (num != 0) {
            int digit = num % 10;
            reverse = reverse * 10 + digit;
            num /= 10;
        }
        return originalNum == reverse;
    }
    public static boolean isPalPrime(int num) {
        return isPrime(num) && isPalindrome(num);
    }
}
```