1. import java.util.Scanner;

class Account {

private int accountId;

private String name;

private double balance;

public Account(int accountId, String name, double balance) {

this.accountId = accountId;

this.name = name;

this.balance = balance;

}

public void transfer(Account toAccount, double amount) {

if (this.balance >= amount) {

this.balance -= amount;

toAccount.balance += amount;

System.out.println("Transaction successful.");

} else {

System.out.println("Insufficient balance.");

}

}

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 class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Create Account A and Account B

Account accountA = new Account(12344, "XXXX", 5000);

Account accountB = new Account(12345, "XXXX", 2500);

// Transfer 1500 from Account A to B

accountA.transfer(accountB, 1500);

// Print receipts after transaction 1

System.out.println("Transaction 1:");

accountA.printReceipt();

accountB.printReceipt();

// Transfer 3000 from Account B to A

accountB.transfer(accountA, 3000);

// Print receipts after transaction 2

System.out.println("Transaction 2:");

accountA.printReceipt();

accountB.printReceipt();

}

}

2)

import java.util.Arrays;

public class PartitionAndMerge {

public static int[] partitionAndMerge(int[] arr, int partitionSize, int[] partitionOrder) {

// Create partitions based on the given size

int[][] partitions = new int[partitionSize][];

int partitionIndex = 0;

for (int i = 0; i < arr.length; i += partitionSize) {

int remaining = Math.min(partitionSize, arr.length - i);

partitions[partitionIndex++] = Arrays.copyOfRange(arr, i, i + remaining);

}

// Merge partitions based on the given order

int[] mergedArr = new int[arr.length];

int mergedIndex = 0;

for (int order : partitionOrder) {

for (int element : partitions[order - 1]) {

mergedArr[mergedIndex++] = element;

}

}

return mergedArr;

}

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

int partitionSize = 2;

int[] partitionOrder = {3, 2, 1};

int[] mergedArr = partitionAndMerge(arr, partitionSize, partitionOrder);

System.out.println("Original array: " + Arrays.toString(arr));

System.out.println("Partition size: " + partitionSize);

System.out.println("Partition order: " + Arrays.toString(partitionOrder));

System.out.println("Merged array: " + Arrays.toString(mergedArr));

}

}

3) class PalPrime {

int number;

String message;

public PalPrime(int number, String message) {

this.number = number;

this.message = message;

System.out.println(message); // Print the message in the constructor

}

}

public class PalPrimeDemo {

public static boolean isPalindrome(int number) {

int reversed = 0, original = number;

while (number != 0) {

int digit = number % 10;

reversed = reversed \* 10 + digit;

number /= 10;

}

return reversed == original;

}

public static boolean isPrime(int number) {

if (number <= 1) {

return false;

}

for (int i = 2; i \* i <= number; i++) {

if (number % i == 0) {

return false;

}

}

return true;

}

public static void main(String[] args) {

int[] arr = {1, 34543, 565, 727, 10099};

for (int num : arr) {

if (isPalindrome(num) && isPrime(num)) {

new PalPrime(num, "Number " + num + " is PalPrime");

} else if (isPalindrome(num)) {

new PalPrime(num, "Number " + num + " is Palindrome");

} else if (isPrime(num)) {

new PalPrime(num, "Number " + num + " is Prime");

}

}

}

}