**QUIZ 12**

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.\_\_\_\_

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. 6500.0Account 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

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class ArrayPartitionAndMerge {

public static void main(String[] args) {

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

int partitionSize = 2;

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

List<List<Integer>> partitions = partitionArray(array, partitionSize);

List<Integer> mergedArray = mergePartitions(partitions, partitionOrder);

System.out.println("Input Array: " + arrayToString(array));

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

System.out.println("Partition Order: " + arrayToString(partitionOrder));

System.out.println("Output Array: " + arrayToString(mergedArray));

}

private static List<List<Integer>> partitionArray(int[] array, int partitionSize) {

List<List<Integer>> partitions = new ArrayList<>();

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

List<Integer> partition = new ArrayList<>();

for (int j = i; j < i + partitionSize && j < array.length; j++) {

partition.add(array[j]);

}

partitions.add(partition);

}

return partitions;

}

private static List<Integer> mergePartitions(List<List<Integer>> partitions, int[] partitionOrder) {

List<Integer> mergedArray = new ArrayList<>();

for (int order : partitionOrder) {

if (order >= 1 && order <= partitions.size()) {

mergedArray.addAll(partitions.get(order - 1));

}

}

return mergedArray;

}

private static String arrayToString(int[] arr) {

StringBuilder sb = new StringBuilder("[");

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

sb.append(arr[i]);

if (i < arr.length - 1) {

sb.append(", ");

}

}

sb.append("]");

return sb.toString();

}

private static String arrayToString(List<Integer> list) {

return arrayToString(list.stream().mapToInt(Integer::intValue).toArray());

}

}

OUTPUT:

Input Array: [1, 2, 3, 4, 5]

Partition Size: 2

Partition Order: [3, 2, 1]

Output Array: [5, 3, 4, 1, 2]

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

public class PalPrime {

public PalPrime(int number) {

String message = "";

if (isPalindrome(number)) {

message += "Palindrome ";

}

if (isPrime(number)) {

message += "Prime";

}

System.out.println("Number " + number + " is " + message);

}

private boolean isPalindrome(int number) {

String strNumber = Integer.toString(number);

String reversed = new StringBuilder(strNumber).reverse().toString();

return strNumber.equals(reversed);

}

private 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;

}

public static void main(String[] args) {

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

for (int number : numbers) {

new PalPrime(number);

}

}

}

OUTPUT:

Number 1 is Palindrome

Number 34543 is Palindrome Prime

Number 565 is Palindrome

Number 727 is Palindrome Prime

Number 10099 is Prime