Q1-= Reverse the number for n input and convert reverse number into single digit number

**package** ManhattanPractice;

**import** java.util.Scanner;

**public** **class** ReverseAndConvert {

**public** **static** **void** main(String[]args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a number: ");

**int** number = scanner.nextInt();

**int** reversedNumber = *reverseNumber*(number);

**int** singleDigitNumber = *convertToSingleDigit*(reversedNumber);

System.***out***.println("Reversed number: " + reversedNumber);

System.***out***.println("Single-digit number: " + singleDigitNumber);

}

// Reverses the digits of a given number

**public** **static** **int** reverseNumber(**int** number) {

**int** reversedNumber = 0;

**while** (number != 0) {

**int** digit = number % 10;

reversedNumber = reversedNumber \* 10 + digit;

number /= 10;

}

**return** reversedNumber;

}

// Converts a given number into a single-digit number

**public** **static** **int** convertToSingleDigit(**int** number) {

**while** (number >= 10) {

**int** sum = 0;

**while** (number != 0) {

**int** digit = number % 10;

sum += digit;

number /= 10;

}

number = sum;

}

**return** number;

}

}

===O(Log10^n)===

Question 2: String s=”Kishan” count of each character in no. of occurrence==

**import** java.util.HashMap;

**import** java.util.Map;

**public** **class** CharacterCount {

**public** **static** **void** main(String[] args) {

String s = "Kishan";

// Create a HashMap to store character counts

Map<Character, Integer> charCountMap = **new** HashMap<>();

// Iterate through the string

**for** (**char** c : s.toCharArray()) {

// Check if the character is already present in the map

**if** (charCountMap.containsKey(c)) {

// If present, increment the count

**int** count = charCountMap.get(c);

charCountMap.put(c, count + 1);

} **else** {

// If not present, add the character to the map with count 1

charCountMap.put(c, 1);

}

}

// Print the character counts

**for** (Map.Entry<Character, Integer> entry : charCountMap.entrySet()) {

System.***out***.println(entry.getKey() + ": " + entry.getValue());

}

}

}

**Que=//Write a code using recursion where the sum of natural numbers that are a multiple of 3 or 5 are printed using numbers below 1000.**

**public** **class** SumOfMultiples {

**public** **static** **void** main(String[] args) {

**int** sum = *calculateSum*(1);

System.***out***.println("Sum of natural numbers that are multiples of 3 or 5: " + sum);

}

**public** **static** **int** calculateSum(**int** number) {

**if** (number >= 1000) {

**return** 0;

} **else** **if** (number % 3 == 0 || number % 5 == 0) {

**return** number + *calculateSum*(number + 1);

} **else** {

**return** *calculateSum*(number + 1);

}

}

}

**Que//Code To find duplicate strings from string list and print them(collection framework)**

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**public** **class** DuplicateStrings {

**public** **static** **void** main(String[] args) {

List<String> stringList = **new** ArrayList<>();

stringList.add("apple");

stringList.add("banana");

stringList.add("apple");

stringList.add("orange");

stringList.add("banana");

stringList.add("grape");

stringList.add("banana");

Map<String, Integer> stringCountMap = **new** HashMap<>();

// Count the occurrences of each string in the list

**for** (String str : stringList) {

**if** (stringCountMap.containsKey(str)) {

**int** count = stringCountMap.get(str);

stringCountMap.put(str, count + 1);

} **else** {

stringCountMap.put(str, 1);

}

}

// Print the duplicate strings

System.***out***.println("Duplicate strings:");

**for** (Map.Entry<String, Integer> entry : stringCountMap.entrySet()) {

**if** (entry.getValue() > 1) {

System.***out***.println(entry.getKey());

}

}

}

}

Select the 2nd highest salary

SELECT MAX(salary) AS second\_highest\_salary

FROM your\_table

WHERE salary < (SELECT MAX(salary) FROM your\_table);

Write a Query of Self Join:

SELECT e.Name AS EmployeeName, m.Name AS ManagerName

FROM Employees e

JOIN Employees m ON e.ManagerID = m.ID

**Que=//Code to find same int elements from 2 list and return count (collection framework)**

**import** java.util.ArrayList;

**import** java.util.HashSet;

**import** java.util.List;

**import** java.util.Set;

**public** **class** SameIntElements {

**public** **static** **void** main(String[] args) {

List<Integer> list1 = **new** ArrayList<>();

list1.add(1);

list1.add(2);

list1.add(3);

list1.add(4);

list1.add(5);

List<Integer> list2 = **new** ArrayList<>();

list2.add(3);

list2.add(4);

list2.add(5);

list2.add(6);

list2.add(7);

**int** count = *findSameIntElementsCount*(list1, list2);

System.***out***.println("Count of same integer elements: " + count);

}

**public** **static** **int** findSameIntElementsCount(List<Integer> list1, List<Integer> list2) {

Set<Integer> set1 = **new** HashSet<>(list1);

Set<Integer> set2 = **new** HashSet<>(list2);

// Intersection of set1 and set2 will contain the same integer elements

set1.retainAll(set2);

**return** set1.size();

}

}

**Synchronised Block:**

**public** **class** ExampleClass {

**private** **int** sharedVariable = 0;

**public** **void** increment() {

// Synchronized block

**synchronized** (**this**) {

// Critical section - only one thread can execute this block at a time

sharedVariable++;

}

}

}

checked vs unchecked exception:

**import** java.io.FileReader;

**import** java.io.IOException;

**public** **class** CheckedExceptionExample {

**public** **static** **void** main(String[] args) {

FileReader reader = **null**;

**try** {

reader = **new** FileReader("example.txt");

// Code that performs operations on the file

} **catch** (IOException e) {

// Exception handling code

e.printStackTrace();

} **finally** {

// Close the resources (e.g., file) in the finally block

**if** (reader != **null**) {

**try** {

reader.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

}

}

Unchecked:

**public** **class** UncheckedExceptionExample {

**public** **static** **void** main(String[] args) {

**int**[] numbers = { 1, 2, 3 };

System.***out***.println(numbers[3]); // Throws ArrayIndexOutOfBoundsException

}

}

Searching topic also explain anyone of them

How to create Database object

Prepare..... 10-20 basic interviews asked programming questions. Basic SQL query ....joins... primary keys Most important knowledge about Microservices. OOP’S

Function Overloading and Function Overriding ,How you achieve abstraction

Access modifier, Static keyword and final keyword Finally keyword

Collection framework (#most important topic)

HAVE YOU USED ANY TESTING TECHNIQUE…, DESIGN PATTERN IN JAVA