51. Write a program to print the multiplication table of number m up to n.

Sample Input:

M = 4

N = 5

Sample Output:

1x4=4

2x4=8

3x4=12

4x4=16

5x4=20

Test cases:

M = 6, N = -3

M = -3, N = 5

M = 4, N = 0

M = 0, N = 0

M = -5, N = -5

**Program :**

import java.util.Scanner;

public class MultiplicationTable {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number for multiplication table: ");

int m = scanner.nextInt();

System.out.print("Enter the range (up to which number) for the multiplication table: ");

int n = scanner.nextInt();

System.out.println("Multiplication table of " + m + " up to " + n + ":");

for (int i = 1; i <= n; i++) {

System.out.println(m + " \* " + i + " = " + (m \* i));

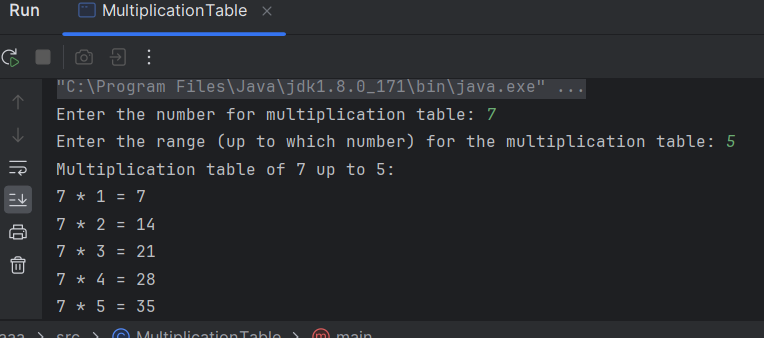
}

scanner.close();

}

}

**Output:**



1. Write Java programs to implement multiple threads and apply join method for thread and thread has to be started after 500ms using sleep ().

**Program:**

class MyThread extends Thread {

public void run() {

System.out.println(Thread.currentThread().getName() + " is running");

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.println(Thread.currentThread().getName() + " has completed");

}

}

public class Main {

public static void main(String[] args) {

MyThread thread1 = new MyThread();

MyThread thread2 = new MyThread();

try {

Thread.sleep(500);

} catch (InterruptedException e) {

System.out.println(e);

}

thread1.start();

thread2.start();

try {

thread1.join();

thread2.join();

} catch (InterruptedException e) {

System.out.println(e);

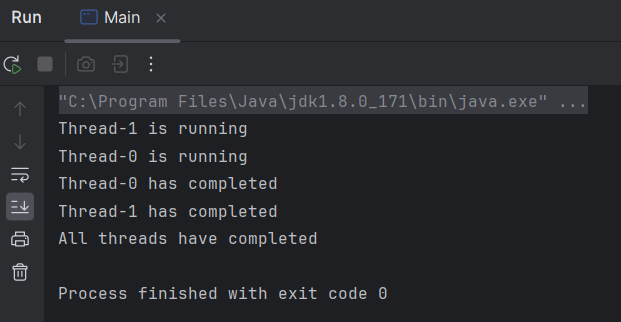
}

System.out.println("All threads have completed");

}

}

**Output:**

****

1. Generate a Java code that implements java selection and iteration statements. Use do while loop to process a menu selection. When a menu is selected, it should display the syntax of the selected statements.

**Program:**

import java.util.Scanner;

public class StatementSyntaxDemo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("Select an option:");

System.out.println("1. If-else statement");

System.out.println("2. Switch statement");

System.out.println("3. For loop");

System.out.println("4. While loop");

System.out.println("5. Do-while loop");

System.out.println("0. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.println("Syntax of If-else statement:");

System.out.println("if (condition) {");

System.out.println(" // statements to be executed if condition is true");

System.out.println("} else {");

System.out.println(" // statements to be executed if condition is false");

System.out.println("}");

break;

case 2:

System.out.println("Syntax of Switch statement:");

System.out.println("switch (expression) {");

System.out.println(" case value1:");

System.out.println(" // statements to be executed if expression == value1");

System.out.println(" break;");

System.out.println(" case value2:");

System.out.println(" // statements to be executed if expression == value2");

System.out.println(" break;");

System.out.println(" default:");

System.out.println(" // statements to be executed if expression doesn't match any case");

System.out.println("}");

break;

case 3:

System.out.println("Syntax of For loop:");

System.out.println("for (initialization; condition; update) {");

System.out.println(" // statements to be executed repeatedly");

System.out.println("}");

break;

case 4:

System.out.println("Syntax of While loop:");

System.out.println("while (condition) {");

System.out.println(" // statements to be executed repeatedly as long as condition is true");

System.out.println("}");

break;

case 5:

System.out.println("Syntax of Do-while loop:");

System.out.println("do {");

System.out.println(" // statements to be executed at least once");

System.out.println("} while (condition);");

break;

case 0:

System.out.println("Exiting the program.");

break;

default:

System.out.println("Invalid choice. Please enter a number between 0 and 5.");

}

System.out.println();

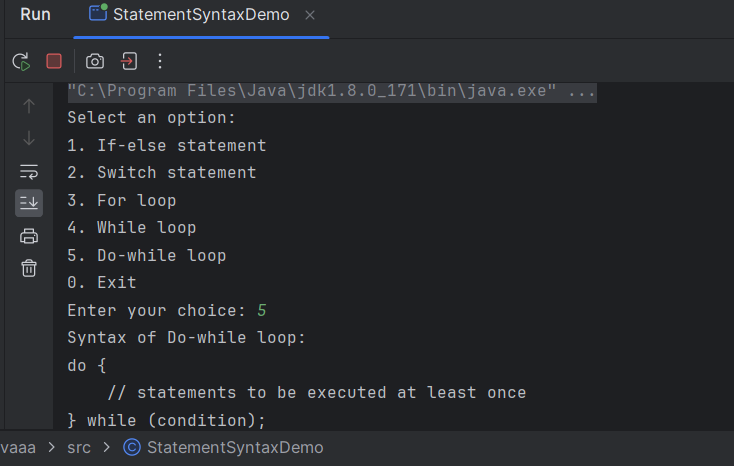
} while (choice != 0);

scanner.close();

}

}

**Output:**



1. Create a simple generics class with type parameters for sorting values of different types.

**Program:**

import java.util.Arrays;

class GenericSort<T extends Comparable<T>> {

private T[] array;

public GenericSort(T[] array) {

this.array = array;

}

public void sort() {

Arrays.sort(array);

}

public void printSortedArray() {

System.out.println(Arrays.toString(array));

}

}

public class Main {

public static void main(String[] args) {

// Example usage for sorting integers

Integer[] intArray = {5, 2, 8, 1, 7};

GenericSort<Integer> intSorter = new GenericSort<>(intArray);

intSorter.sort();

intSorter.printSortedArray();

// Example usage for sorting strings

String[] strArray = {"apple", "orange", "banana", "grape"};

GenericSort<String> strSorter = new GenericSort<>(strArray);

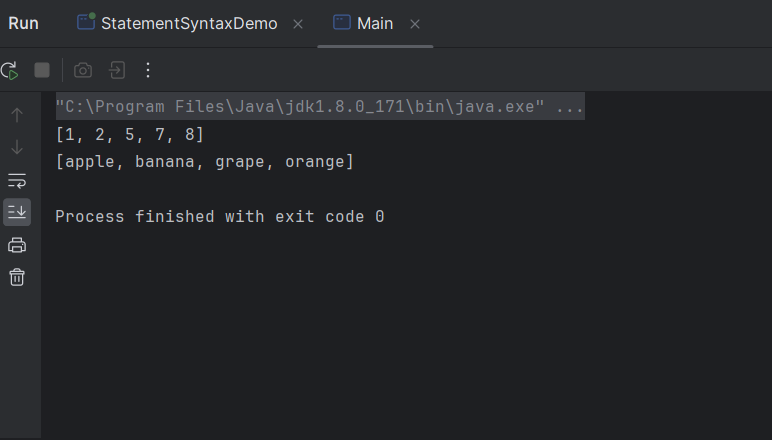
strSorter.sort();

strSorter.printSortedArray();

}

}

**Output:**

****

1. Create a class name ‘overload’. Write a program to assign the values for two values by different number of arguments using a single function.

**Program:**

public class Overload {

private int value1;

private int value2;

public void assignValues(int... args) {

if (args.length == 1) {

value1 = args[0];

value2 = 0; // Default value if only one argument provided

} else if (args.length == 2) {

value1 = args[0];

value2 = args[1];

} else {

System.out.println("Invalid number of arguments");

}

}

public static void main(String[] args) {

Overload obj = new Overload();

// Assigning a single value

obj.assignValues(10);

System.out.println(obj.value1); // Output: 10

System.out.println(obj.value2); // Output: 0 (default value)

// Assigning two values

obj.assignValues(20, 30);

System.out.println(obj.value1); // Output: 20

System.out.println(obj.value2); // Output: 30

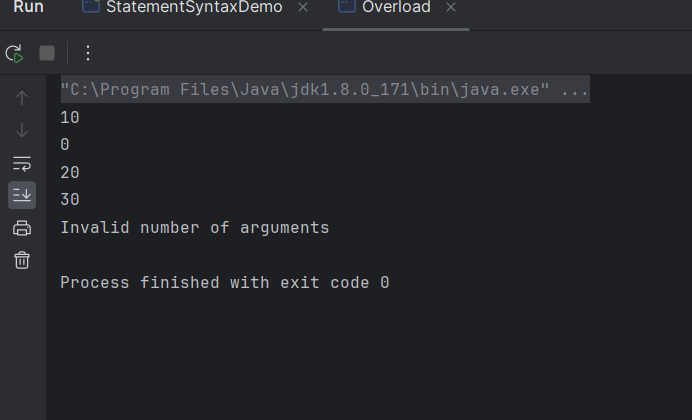
// Invalid number of arguments

obj.assignValues(5, 8, 12); // Output: Invalid number of arguments

}

}

**Output:**

****

1. Write a Java Program to count the number of words in a string using Hash Map

**Program:**

import java.io.\*;

import java.util.HashMap;

import java.util.Map;

class GFG {

public static void main(String[] args)

{

// Declaring the String

String str = "Alice is girl and Bob is boy";

// Declaring a HashMap of <String, Integer>

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

// Splitting the words of string

// and storing them in the array.

String[] words = str.split(" ");

for (String word : words) {

// Asking whether the HashMap contains the

// key or not. Will return null if not.

Integer integer = hashMap.get(word);

if (integer == null)

// Storing the word as key and its

// occurrence as value in the HashMap.

hashMap.put(word, 1);

else {

// Incrementing the value if the word

// is already present in the HashMap.

hashMap.put(word, integer + 1);

}

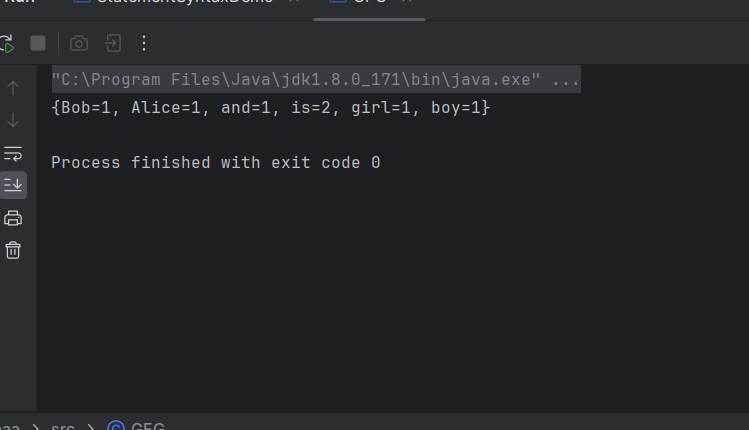
}

System.out.println(hashMap);

}

}

**Output:**

****

1. Write a Java Program to read an email and password from excel sheet by retrieving the cell using getRow() and getCell() method.

**Program:**

public class underflow {

public static void main(String[] args) {

int minInt = Integer.MIN\_VALUE;

int underflowedValue = minInt - 1;

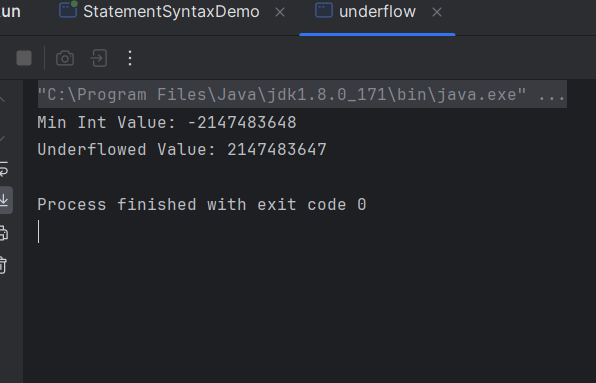
System.out.println("Min Int Value: " + minInt);

System.out.println("Underflowed Value: " + underflowedValue);

}

}

**Output:**

****

1. Write a Java program to sorts the given value using Hash Map.

**Program:**

**import java.util.HashMap;**

**import java.util.Scanner;**

**import java.util.TreeMap;**

**public class SortHashMapByKey {**

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

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

**// Get input**

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

**System.out.println("Enter key-value pairs (e.g., name:age). Enter 'done' to finish.");**

**while (true) {**

**String input = scanner.nextLine();**

**if (input.equals("done")) {**

**break;**

**}**

**String[] parts = input.split(":");**

**if (parts.length != 2) {**

**System.out.println("Invalid format. Please enter key:value.");**

**continue;**

**}**

**String key = parts[0].trim();**

**int value = Integer.parseInt(parts[1].trim());**

**map.put(key, value);**

**}**

**// Sort the map by keys**

**TreeMap<String, Integer> sortedMap = new TreeMap<>(map);**

**// Print the sorted map**

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

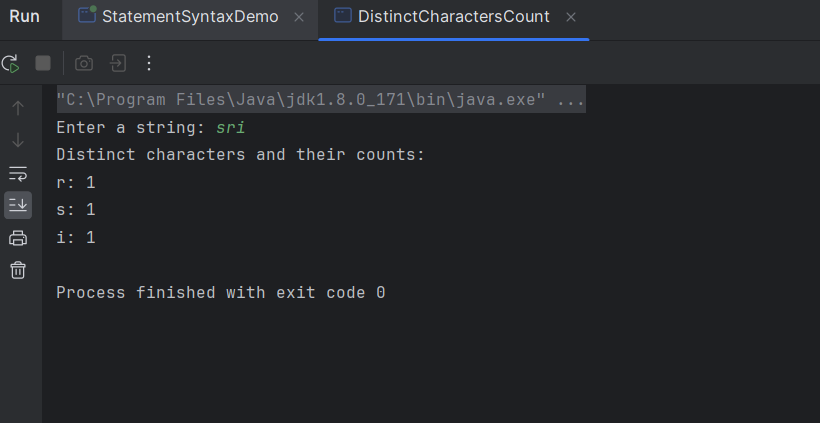
**System.out.println(entry.getKey() + ": " + entry.getValue());**

**}**

**}**

**}**

**Output:**

****

1. Write a Java program to find distinct characters and their count in a string.

**Program:**

import java.util.HashMap;

import java.util.Map;

public class CharacterCounter {

public static void main(String[] args) {

String inputString = "programming";

Map<Character, Integer> charCountMap = countDistinctCharacters(inputString);

System.out.println("Distinct characters and their count:");

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

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

}

}

private static Map<Character, Integer> countDistinctCharacters(String str) {

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

for (char c : str.toCharArray()) {

// If the character is already present, increment the count; otherwise, add it to the map with count 1.

charCountMap.put(c, charCountMap.getOrDefault(c, 0) + 1);

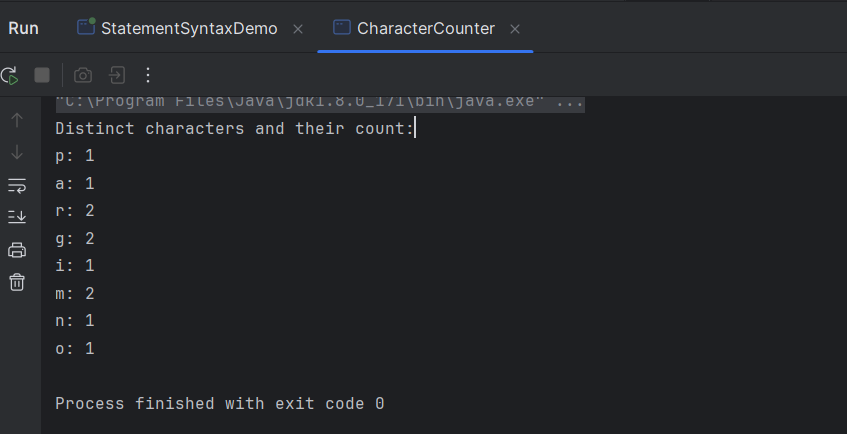
}

return charCountMap;

}

}

**Output:**

****

1. Write a program to print all the unique characters in a String. For instance, if the input string is “abcb”, the output will be the characters ‘a’ and ‘c’ as they are unique. The character ‘b’ repeats twice and so it will not be printed.

**Program:**

import java.util.HashSet;

import java.util.Scanner;

public class UniqueCharacters {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String inputString = scanner.nextLine();

// Use a HashSet to store unique characters

HashSet<Character> uniqueChars = new HashSet<>();

// Iterate over each character in the string

for (char character : inputString.toCharArray()) {

// Add character to the set only if it's not already present

if (!uniqueChars.contains(character)) {

uniqueChars.add(character);

}

}

// Print the unique characters

System.out.println("Unique characters:");

for (char character : uniqueChars) {

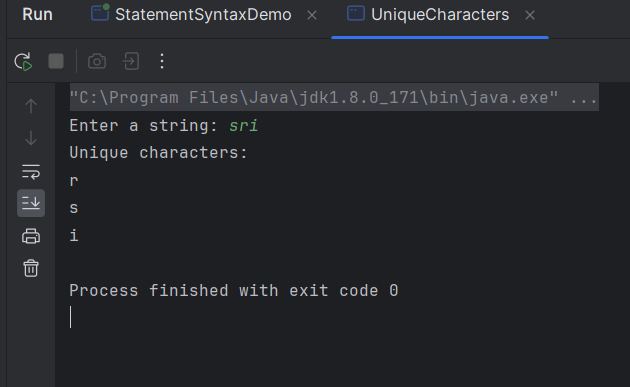
System.out.println(character);

}

}

}

**Output:**

****