**LAB 1**

1. Write a program in which we are trying to ask the user to guess a randomly generated number. The number is in range of 1-100. If the user guesses the number which is greater than the generated number, print “The number is too large”. If the user guesses the number which is less than the generated number, print “The number is too small”. If the user guesses the number correctly, print “Correct Number!”.

At the end we will print the number that was randomly generated.

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

//generate a random Number

int randomNumber = (int)(Math.random() \* 100);

// creating object of scanner

Scanner scanner = new Scanner(System.in);

//Taking input from User

System.out.println("Enter a number between 1-100");

int guessNumber = scanner.nextInt();

//condition for checking Random number

if (guessNumber > randomNumber){

System.out.println("The number is greater");

}

else if(guessNumber < randomNumber){

System.out.println("The number is smaller");

}

else {

System.out.println("Correct Number!");

}

System.out.println(randomNumber+" is the Random Number");

scanner.close();

}

}

1. Write a program that takes your full name as input and displays the abbreviations of the first and middle names except the last name which is displayed as it is. For example, if your name is Robert Brett Roser, then the output should be R.B.Roser.

import java.util.Scanner;

public class NameAbbreviation {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter your full name: ");

String fullName = scanner.nextLine();

String[] names = fullName.split(" ");

StringBuilder abbreviation = new StringBuilder();

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

if (i == names.length - 1) {

abbreviation.append(names[i]);

} else {

abbreviation.append(names[i].charAt(0)).append(".");

}

}

System.out.println("Abbreviated name: " + abbreviation.toString());

scanner.close();

}

}

1. Input a string of alphabets. Find out the number of occurrences of all alphabets in that string. Find out the alphabet with maximum occurrence.

import java.util.HashMap;

import java.util.Map;

public class AlphabetCounter {

public static void main(String[] args) {

// Input string of alphabets

String input = "Hello World";

// Convert the input string to lowercase

input = input.toLowerCase();

// Create a HashMap to store the count of each alphabet

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

// Iterate through each character in the input string

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

char c = input.charAt(i);

// Check if the character is an alphabet

if (Character.isLetter(c)) {

// If the alphabet is already present in the HashMap, increment its count

if (alphabetCount.containsKey(c)) {

alphabetCount.put(c, alphabetCount.get(c) + 1);

} else {

// If the alphabet is not present in the HashMap, add it with count 1

alphabetCount.put(c, 1);

}

}

}

// Find the alphabet with maximum occurrence

char maxAlphabet = ' ';

int maxCount = 0;

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

if (entry.getValue() > maxCount) {

maxAlphabet = entry.getKey();

maxCount = entry.getValue();

}

}

// Print the count of each alphabet

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

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

}

// Print the alphabet with maximum occurrence

System.out.println("Alphabet with maximum occurrence: " + maxAlphabet);

}

}

1. Write a program to find out longest palindrome in a given string?

**Input:** Given string:"forgeeksskeegfor",

**Output:** "geeksskeeg".

public class LongestPalindrome {

public static String findLongestPalindrome(String str) {

if (str == null || str.length() < 2) {

return str;

}

int start = 0;

int end = 0;

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

int len1 = expandAroundCenter(str, i, i);

int len2 = expandAroundCenter(str, i, i + 1);

int len = Math.max(len1, len2);

if (len > end - start) {

start = i - (len - 1) / 2;

end = i + len / 2;

}

}

return str.substring(start, end + 1);

}

private static int expandAroundCenter(String str, int left, int right) {

while (left >= 0 && right < str.length() && str.charAt(left) == str.charAt(right)) {

left--;

right++;

}

return right - left - 1;

}

public static void main(String[] args) {

String input = "forgeeksskeegfor";

String longestPalindrome = findLongestPalindrome(input);

System.out.println("Longest Palindrome: " + longestPalindrome);

}

}

The program starts by defining a class called LongestPalindrome. Inside this class, we have a static method findLongestPalindrome that takes a string as input and returns the longest palindrome found in the string.

The findLongestPalindrome method first checks if the input string is null or has a length less than 2. If so, it returns the input string itself, as a single character or an empty string is considered a palindrome.

Next, we initialize two variables start and end to keep track of the indices of the longest palindrome found so far.

We then iterate through each character in the string using a for loop. For each character, we call the expandAroundCenter method twice, once considering the current character as the center of the palindrome and once considering the current character and the next character as the center. The expandAroundCenter method returns the length of the palindrome found by expanding from the given center.

We compare the lengths obtained from both cases and store the maximum length in the variable len. If the length is greater than the difference between end and start, we update the values of start and end to the indices of the new longest palindrome.

Finally, we return the substring of the input string from start to end + 1, which represents the longest palindrome found.

In the main method, we create an instance of the LongestPalindrome class and call the findLongestPalindrome method with the given input string "forgeeksskeegfor". The resulting longest palindrome is then printed to the console.

1. Write a program to remove duplicate characters from String.

public class RemoveDuplicates {

public static void main(String[] args) {

String input = "Hello World";

String result = removeDuplicates(input);

System.out.println("String after removing duplicates: " + result);

}

public static String removeDuplicates(String str) {

StringBuilder sb = new StringBuilder();

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

char c = str.charAt(i);

if (sb.indexOf(String.valueOf(c)) == -1) {

sb.append(c);

}

}

return sb.toString();

}

}

This program defines a class called RemoveDuplicates with a main method. Inside the main method, we have a string variable input initialized with the value "Hello World". We then call the removeDuplicates method passing the input string as an argument and store the result in a string variable called result.

The removeDuplicates method takes a string as input and returns a string with duplicate characters removed. It uses a StringBuilder to build the resulting string. We iterate over each character in the input string using a for loop. For each character, we check if it already exists in the StringBuilder using the indexOf method. If the character is not found, we append it to the StringBuilder. Finally, we return the resulting string by converting the StringBuilder to a string using the toString method.

In the main method, we print the string after removing duplicates using the System.out.println statement.