**1).Create a package Mathematics with two classes Maximum and Power. Write a java program to accept two numbers from user and perform the following operations on it: a. Find Maximum of two numbers. b. Calculate the power (X, Y);**

import mathematics.\*;

import java.util.\*;

public class Assignment2Q1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter two No :" );

int a=sc.nextInt();

int b= sc.nextInt();

Maximum m = new Maximum(a,b);

m.display();

Power p = new Power(a,b);

p.disp();

sc.close();

}

}

package mathematics;

public class Maximum {

int a, b;

public Maximum(int a, int b) {

this.a = a;

this.b = b;

}

public void display() {

if (a > b) {

System.out.println("a is max");

} else {

System.out.println("b is max");

}

}

}

package mathematics;

public class Power {

int N, P;

public Power(int a, int b) {

N = a;

P = b;

}

public void disp() {

int pow = 1;

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

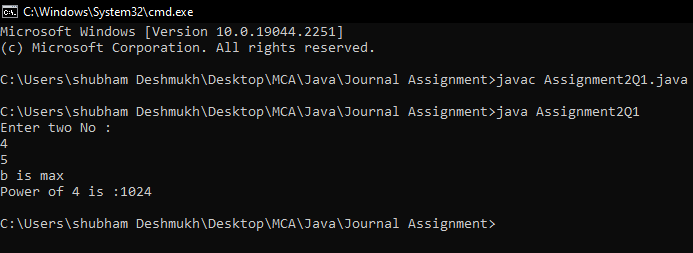
pow = pow \* N;

}

System.out.println("Power of " + N + " is :" + pow);

}

}



**2).Write a program that generates a custom exception if age entered for voting in election is less than 18 years**

import java.util.Scanner;

class Invalid\_Data extends Exception {}

public class Assignment2Q2 {

public static void main(String[] args) throws Invalid\_Data {

Scanner sc = new Scanner(System.in);

System.out.println("Enter Age : ");

int age = sc.nextInt();

if (age < 18) {

throw new Invalid\_Data();

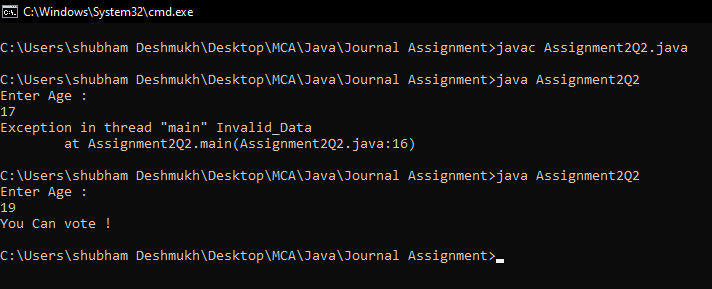
} else {

System.out.println("You Can vote !");

}

}

}



1. **.Write a program to accept 10 elements of an array from user and find:**

**a. Greatest element**

**b. Smallest element**

**c. Sum of elements**

**d. Average of elements of array**

import java.util.\*;

public class Assignment2Q3 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int[] arr = new int[10];

System.out.println("Enter 10 array elements : ");

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

arr[i] = sc.nextInt();

}

System.out.println("\n Array elements : ");

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

System.out.print(arr[i] + " ");

}

System.out.print("\n");

int max = arr[0];

for (int i = 0; i < 10; i++) {

if (arr[i] > max) {

max = arr[i];

}

}

System.out.println(

"\n" + max + " is Maximum among all the Array Elements "

);

int min = arr[0];

for (int i = 0; i < 10; i++) {

if (arr[i] < min) {

min = arr[i];

}

}

System.out.println(

"\n" + min + " is Minimum among all the Array Elements "

);

int sum = 0;

for (int i = 0; i < 10; i++) {

sum = sum + arr[i];

}

System.out.println("\n" + sum + " is the Sum of all the Array Elements ");

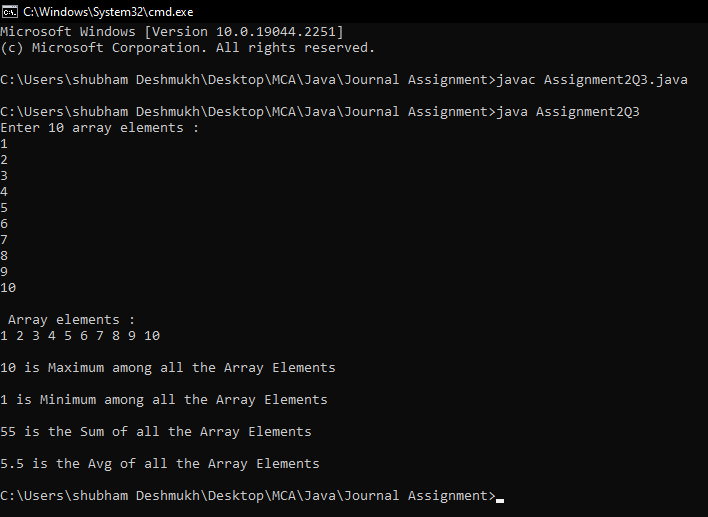
float avg = (float)sum / 10;

System.out.println("\n" + avg + " is the Avg of all the Array Elements ");

sc.close();

}

}



**4).Write a program to accept two string from user and check both strings are equal or not.**

import java.util.\*;

public class Assignment2Q4 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter First String : ");

String str1 = sc.next();

System.out.println("Enter Second String : ");

String str2 = sc.next();

if(str1.equals(str2)){

System.out.println("String are Equal ");

}

else{

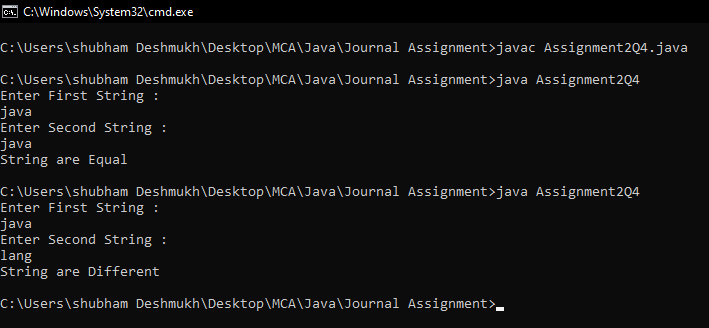
System.out.println("String are Different ");

}

sc.close();

}

}



**5).Write a package for Games in Java, which have two classes Indoor and Outdoor. Use a function display () to generate the list of players for the specific games. (Use Parameterized constructor, finalize() method and Array Of Objects)**

import java.util.Scanner;

import mathematics.\*;

public class Assignment2Q5{

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int ch;

do {

System.out.println("\n1.Indoor \n2.Outdoor\n\nEnter Your Choice : ");

ch = sc.nextInt();

switch (ch) {

case 1:

System.out.println("Enter how many players : ");

int n = sc.nextInt();

Indoor[] in = new Indoor[n];

System.out.println("\nEnter " + n + " Indoor Player Name : ");

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

String p\_name = sc.next();

in[i] = new Indoor(p\_name);

}

System.out.println("\nIndoor Games : ");

for (int j = 0; j < in.length; j++) {

in[j].display();

}

break;

case 2:

System.out.println("Enter how many players : ");

int n1 = sc.nextInt();

Outdoor[] out = new Outdoor[n1];

System.out.println("\nEnter " + n1 + " Outdoor Player Name : ");

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

String p\_name = sc.next();

out[i] = new Outdoor(p\_name);

}

System.out.println("\nOutdoor Games : ");

for (int j = 0; j < out.length; j++) {

out[j].display();

}

default:

break;

}

} while (ch < 3);

sc.close();

}

}

package mathematics;

public class Indoor {

String player;

public Indoor(String player) {

this.player = player;

}

public void display() {

System.out.println(player);

}

protected void finalize() {

System.out.println("Finalize is called");

}

}

package mathematics;

public class Outdoor {

String player;

public Outdoor(String player) {

this.player = player;

}

public void display() {

System.out.println(player);

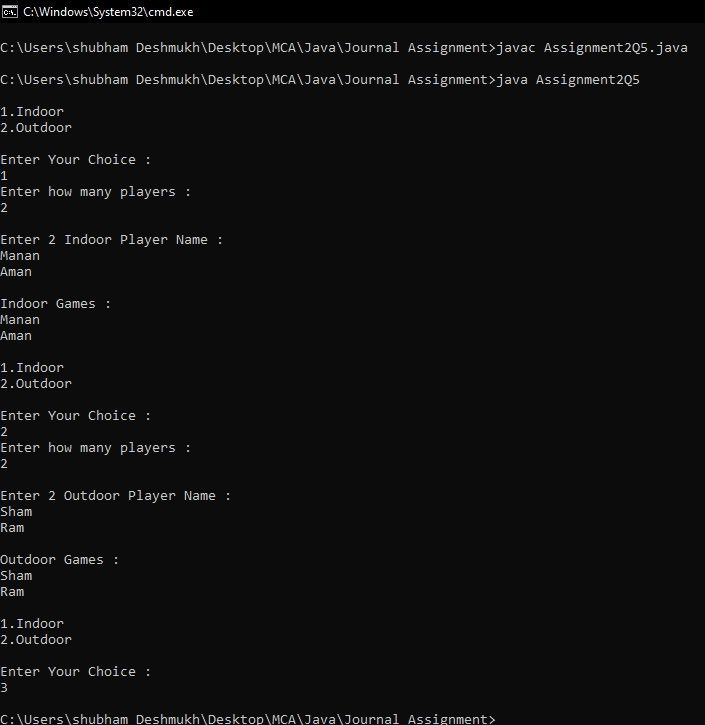
}

public void finalize() {

System.out.println("Finalize is called");

}

}



**7).Write a Menu driven program for the following string operations**

**a) To find reverse string**

**b) Concatenate Two Strings.**

**c) Find the Length of the String.**

**d) to Use Equals Method In a String Class**

**e) To convert given string in Uppercase.**

**f) To convert given string in lowercase.**

import java.util.Scanner;

public class Assignment2Q7 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int ch;

do {

System.out.println(

"\n1.Reverse String \n2.Concatenation \n3.Length \n4.Equal Method \n5.UpperCase\n6.Lowercase\n7.Exit"

);

System.out.println("Enter your Choice :");

ch = sc.nextInt();

switch (ch) {

case 1:

String nstr = " ";

System.out.println("Enter String : ");

String str = sc.next();

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

char c = str.charAt(i);

nstr = c + nstr;

}

System.out.println("Reversed String : " + nstr);

break;

case 2:

System.out.println("Enter First String : ");

String str1 = sc.next();

System.out.println("Enter Second String : ");

String str2 = sc.next();

System.out.println("Concatenated String : " + str1.concat(str2));

break;

case 3:

System.out.println("Enter String : ");

String str4 = sc.next();

System.out.println("Length of String : " + str4.length());

break;

case 4:

System.out.println("Enter First String : ");

String str5 = sc.next();

System.out.println("Enter Second String : ");

String str6 = sc.next();

if (str5.equals(str6)) {

System.out.println("Strings are Equal");

} else {

System.out.println("Strings are Different");

}

break;

case 5:

System.out.println("Enter String : ");

String str7 = sc.next();

System.out.println("UpperCase of String : " + str7.toUpperCase());

break;

case 6:

System.out.println("Enter String : ");

String str8 = sc.next();

System.out.println("Lowercase of String : " + str8.toLowerCase());

break;

default:

break;

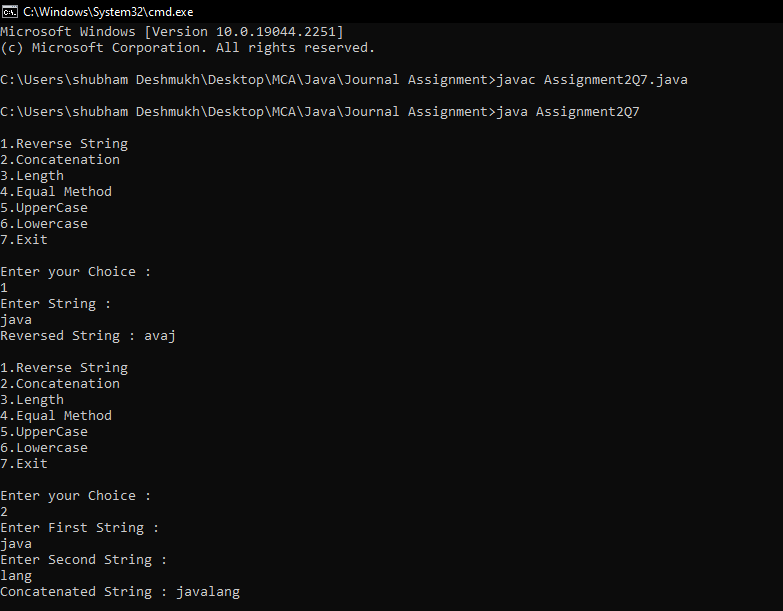
}

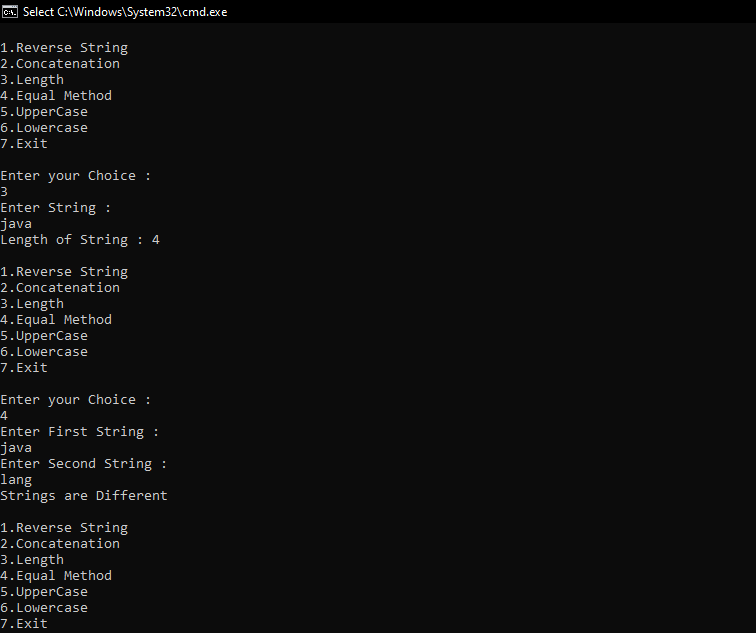
} while (ch < 7);

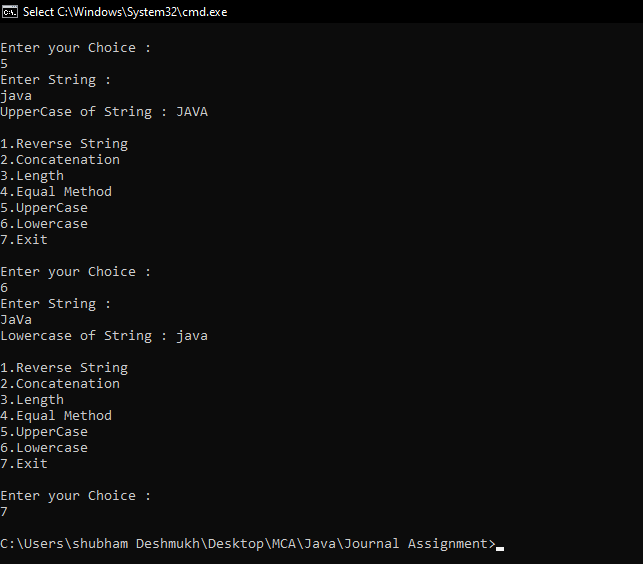
sc.close();

}

}







**8.Write a program to create a graph and perform the following operations: a)Add Vertex b)Add edge c)Display**

class Graph {

constructor() {

this.adjancylist = {};

}

addVertex(vertex) {

if (!this.adjancylist[vertex]) {

this.adjancylist[vertex] = new Set();

}

}

addEdge(vertex1, vertex2) {

if (!this.adjancylist[vertex1]) {

this.addVertex(vertex1);

}

if (!this.adjancylist[vertex2]) {

this.addVertex(vertex2);

}

this.adjancylist[vertex1].add(vertex2);

this.adjancylist[vertex2].add(vertex1);

}

display() {

for (let vertex in this.adjancylist) {

console.log(vertex + "->" + [... this.adjancylist[vertex]]);

}

}

}

const g = new Graph();

g.addVertex(10);

g.addVertex(20);

g.addVertex(30);

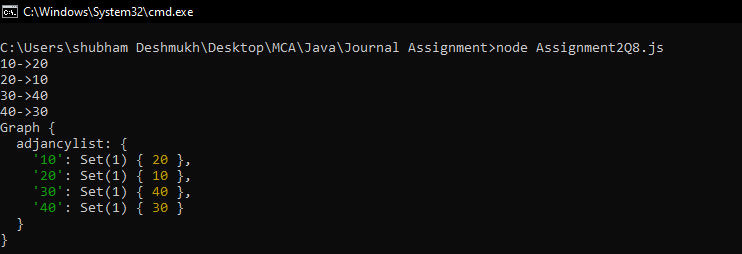
g.addVertex(40);

g.addEdge(10, 20);

g.addEdge(30, 40);

g.display();

console.log(g)



**9).Write a program to create a graph and perform the following operations: a)Delete vertex b)Delete edge**

class Graph {

constructor() {

this.adjancencyList = {};

}

addVertex(vertex) {

if (!this.adjancencyList[vertex]) {

this.adjancencyList[vertex] = new Set();

}

}

addEdge(vertex1, vertex2) {

if (!this.adjancencyList[vertex1]) {

this.addVertex(vertex1);

}

if (!this.adjancencyList[vertex2]) {

this.addVertex(vertex2);

}

this.adjancencyList[vertex1].add(vertex2);

this.adjancencyList[vertex2].add(vertex1);

}

removeEdge(vertex1, vertex2) {

this.adjancencyList[vertex1].delete(vertex2);

this.adjancencyList[vertex2].delete(vertex1);

}

display() {

for (let vert in this.adjancencyList) {

console.log(vert + "-->" + [...this.adjancencyList[vert]]);

}

}

}

const g = new Graph();

g.addVertex('A');

g.addVertex('B');

g.addVertex('C');

g.addVertex('D');

g.addEdge('A', 'B');

g.addEdge('C', 'A');

g.addEdge('B', 'C');

g.addEdge('C', 'B');

g.addEdge('D', 'A');

g.addEdge('C', 'A');

console.log("\nEdges Between Vertices : ");

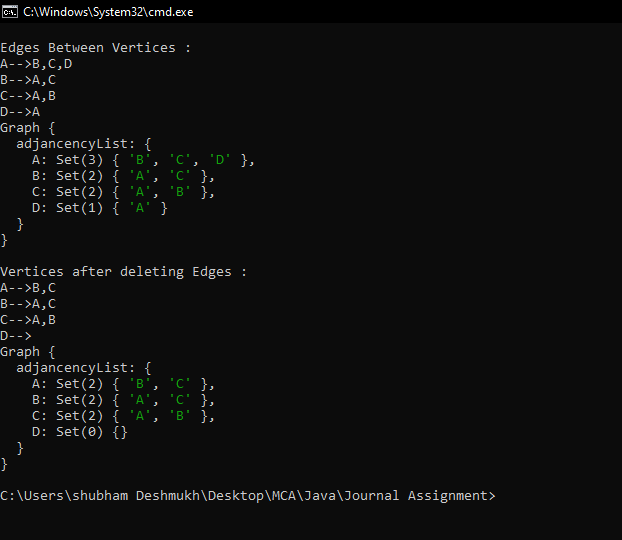
g.display();

g.removeEdge('A','B');

g.removeEdge('D','A');

console.log("\nVertices after deleting Edges : ");

g.display();



**10).Write a program to create a graph and perform DFS**

class Graph {

constructor(v) {

this.V = v;

this.adj = new Array(v);

for (let i = 0; i < v; i++) {

this.adj[i] = [];

}

}

addEdge(v, w) {

this.adj[v].push(w);

}

DFSUtil(v, visited) {

visited[v] = true;

console.log(v + " ");

for (let i of this.adj[v].values()) {

let n = i;

if (!visited[n]) this.DFSUtil(n, visited);

}

}

DFS(v) {

let visited = new Array(this.V);

for (let i = 0; i < this.V; i++) {

visited[i] = false;

}

this.DFSUtil(v, visited);

}

}

g = new Graph(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

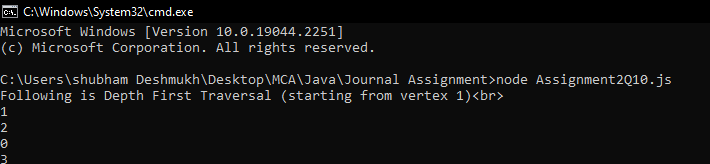
g.addEdge(3, 3);

console.log(

"Following is Depth First Traversal " + "(starting from vertex 1)<br>"

);

g.DFS(1)



**11).Write a program to create a graph and perform DFS**

class Graph {

constructor(v) {

this.V = v;

this.adj = new Array(v);

for (let i = 0; i < v; i++) this.adj[i] = [];

}

addEdge(v, w) {

this.adj[v].push(w);

}

BFS(s) {

let visited = new Array(this.V);

for (let i = 0; i < this.V; i++) {

visited[i] = false;

}

let queue = [];

visited[s] = true;

queue.push(s);

while (queue.length > 0) {

s = queue[0];

console.log(s + " ");

queue.shift();

this.adj[s].forEach((adjacent, i) => {

if (!visited[adjacent]) {

visited[adjacent] = true;

queue.push(adjacent);

}

});

}

}

}

g = new Graph(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

console.log(

"Following is Breadth First Traversal " + "(starting from vertex 1)<br>"

);

g.BFS(1)

