**CSA09 – Programming in Java**

**Day 3 Assignment Questions**

1. Write a program in Java for dynamically changing the color of Text using Multithreading.
2. Differentiate Multiprocessing and Multithreading. Display Multiplication table for 5 and 10 using various stages of life cycle of the thread by generating a suitable code in Java.

Program:

import java.io.\*;

import java.util.\*;

class Table

{

void printTable(int n)

{

synchronized(this)

{

for(int i=1;i<=5;i++)

{

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

try

{

Thread.sleep(500);

}

catch(Exception e)

{

System.out.println("INVALID");

}

}

}

}

}

class Mythread1 extends Thread

{

Table a;

Mythread1(Table a)

{

this.a=a;

}

public void run()

{

a.printTable(5);

}

}

class Mythread2 extends Thread

{

Table a;

Mythread2(Table a)

{

this.a=a;

}

public void run()

{

a.printTable(10);

}

}

class t

{public static void main(String args[])

{

Table obj=new Table();

Mythread1 th1=new Mythread1(obj);

Mythread2 th2=new Mythread2(obj);

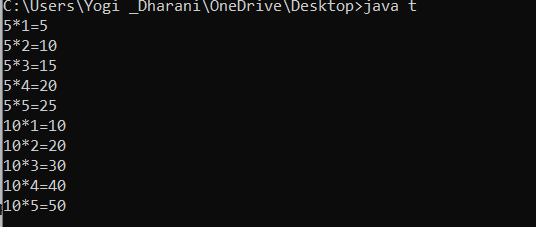
th1.start();

th2.start();

}

}

Output:



1. An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5.

Given an integer n, return true if n is an ugly number.

Example 1:

Input: n = 6

Output: true

Explanation: 6 = 2 × 3

Example 2:

Input: n = 1

Output: true

Explanation: 1 has no prime factors, therefore all of its prime factors are limited to 2, 3, and 5.

Example 3:

Input: n = 14

Output: false

Explanation: 14 is not ugly since it includes the prime factor 7.

Constraints:

-231 <= n <= 231 - 1

class Solution {

public:

bool isUgly(int n) {

}

}

Program:

import java.util.Scanner;

public class UglyNumber {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int n = input.nextInt();

boolean isUgly = isUglyNumber(n);

if (isUgly) {

System.out.println(n + " is an ugly number");

} else {

System.out.println(n + " is not an ugly number");

}

}

public static boolean isUglyNumber(int n) {

if (n <= 0) {

return false;

}

while (n % 2 == 0) {

n /= 2;

}

while (n % 3 == 0) {

n /= 3;

}

while (n % 5 == 0) {

n /= 5;

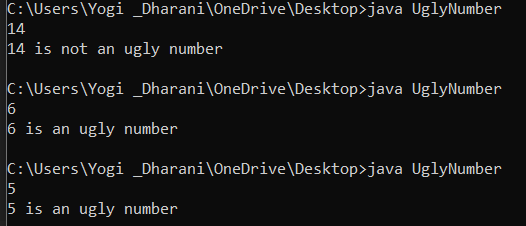
}

return n == 1;

}

}

Output:



1. The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,

F(0) = 0, F(1) = 1

F(n) = F(n - 1) + F(n - 2), for n > 1.

Given n, calculate F(n).

Example 1:

Input: n = 2

Output: 1

Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.

Example 2:

Input: n = 3

Output: 2

Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.

Example 3:

Input: n = 4

Output: 3

Explanation: F(4) = F(3) + F(2) = 2 + 1 = 3.

Constraints:

0 <= n <= 30

class Solution {

public:

int fib(int n) {

}

}

Program:

import java.io.\*;

import java.util.\*;

class fibo

{

public static void main(String args[])

{

try

{

int a=0,b=1,c=0;

Scanner sc=new Scanner(System.in);

System.out.println("Enter the number:");

int n=sc.nextInt();

for(int i=0;i<=n;i++)

{

if(n==0)

{

System.out.println("the fibonacci number at position "+n+" is "+a);

}

else if(n==1)

{

System.out.println("the fibonacci number at position "+n+" is "+b);

}

else{

c=a+b;

a=b;

b=c;

System.out.println("the fibonacci number at position "+n+" is "+c);

}

}

}

catch(Exception e)

{

System.out.println("Invalid");

}

}

}

Output:



1. Removing duplicate elements in java : Find/Debug the errors and get output

class duplicate

{

// Function to remove duplicate elements

// This function returns new size of modified

// array.

static int removeDuplicates(int arr[], int n)

{

// Return, if array is empty

// or contains a single element

if (n==0 || n==1)

return n;

int[] temp = new int[n];

// Start traversing elements

int j = 0;

for (int j=0; i<n-1; i++)

// If current element is not equal

// to next element then store that

// current element

if (arr[i] != arr[i+1])

temp[j++] = arr[i];

// Store the last element as whether

// it is unique or repeated, it hasn't

// stored previously

temp[j++] = arr[n-1];

// Modify original array

for (int i=0; i<j; i++)

arr[i] = temp[i];

return j;

}

public static void main (String[] args)

{

it arr[] = {10, 20, 20, 30, 40, 40, 40, 50, 50};

int n = arr.length;

n = removeDuplicates(arr);

// Print updated array

for (int i=0; i<n; i++)

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

}

}

Program:

class duplicate

{

// Function to remove duplicate elements

// This function returns new size of modified

// array.

static int removeDuplicates(int arr[], int n)

{

// Return, if array is empty

// or contains a single element

if (n==0 || n==1)

return n;

int[] temp = new int[n];

// Start traversing elements

int j = 0;

for (int j=0; i<n-1; i++)

// If current element is not equal

// to next element then store that

// current element

if (arr[i] != arr[i+1])

temp[j++] = arr[i];

// Store the last element as whether

// it is unique or repeated, it hasn't

// stored previously

temp[j++] = arr[n-1];

// Modify original array

for (int i=0; i<j; i++)

arr[i] = temp[i];

return j;

}

public static void main (String[] args)

{

it arr[] = {10, 20, 20, 30, 40, 40, 40, 50, 50};

int n = arr.length;

n = removeDuplicates(arr);

// Print updated array

for (int i=0; i<n; i++)

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

}

}

Output:

