**ASSIGNMENT 3**

**DUPLICATE**

|  |
| --- |
| 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 i=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) |
|  | { |
|  | int arr[] = {10, 20, 20, 30, 40, 40, 40, 50, 50}; |
|  | int n = arr.length; |
|  |  |
|  | n = removeDuplicates(arr, n); |
|  |  |
|  | // Print updated array |
|  | for (int i=0; i<n; i++) |
|  | System.out.print(arr[i]+" "); |
|  | } |
|  | } |

## ****Fibonaci Sequence****

|  |
| --- |
| import java.util.Scanner; |
|  |  |
|  | public class Fibonacci { |
|  | public static void main(String[] args) { |
|  | Scanner sc = new Scanner(System.in); |
|  | System.out.print("Enter the value of n: "); |
|  | int n = sc.nextInt(); |
|  |  |
|  | int a = 0, b = 1, c = 0; |
|  | 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 { |
|  | for (int i = 2; i <= n; i++) { |
|  | c = a + b; |
|  | a = b; |
|  | b = c; |
|  | } |
|  | System.out.println("The Fibonacci number at position " + n + " is " + c); |
|  | } |
|  | } |
|  | } |
|  |  |

## ****Mulitplication.java****

|  |
| --- |
| import java.io.\*; |
|  | class Table |
|  | { |
|  | void printTable(int n) |
|  | { |
|  | synchronized(this) |
|  | { |
|  | for (int i =1;i<=10;i++) |
|  | { |
|  | System.out.println(+n+"\*"+i+"="+(n\*i)); |
|  | try |
|  | { |
|  | Thread.sleep(500); |
|  | } |
|  | catch(Exception e) |
|  | { |
|  | System.out.println(e); |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | class MyThread1 extends Thread |
|  | { |
|  | Table t; |
|  | MyThread1(Table t) |
|  | { |
|  | this.t=t; |
|  | } |
|  | public void run() |
|  | { |
|  | t.printTable(5); |
|  | } |
|  | } |
|  | class MyThread2 extends Thread |
|  | { |
|  | Table t; |
|  | MyThread2(Table t) |
|  | { |
|  | this.t=t; |
|  | } |
|  | public void run() |
|  | { |
|  | t.printTable(10); |
|  | } |
|  | } |
|  | class Use |
|  | { |
|  | public static void main(String args[]) |
|  | { |
|  | Table obj = new Table(); |
|  | MyThread1 th1 = new MyThread1(obj); |
|  | MyThread2 th2 = new MyThread2(obj); |
|  | th1.start(); |
|  | th2.start(); |
|  | } |
|  | } |

## ****UglyNumber.java****

|  |
| --- |
| 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; |
|  | } |
|  | } |