JAVA THREADS

1. THREADS ALLOW A PROGRAM TO OPERATE MORE EFFECIENTLY BY DOING MULTIPLE THINGS AT THE SAME TIME
2. THREADS CAN BE USED TO PERFORM COMPLICATED TASK IN BACKGROUND

\*CREATING A THREAD

THERE ARE TWO WAYS TO CREATE A THREAD

IT CAN BE CREATED BY EXTENDING THREAD CLASS AND OVERRIDING ITS RUN() METHOD

class Main extends Thread{

public void run(){

System.out.println("This code is running in thread");

}

public static void main(String []args){

Main thread=new Main();

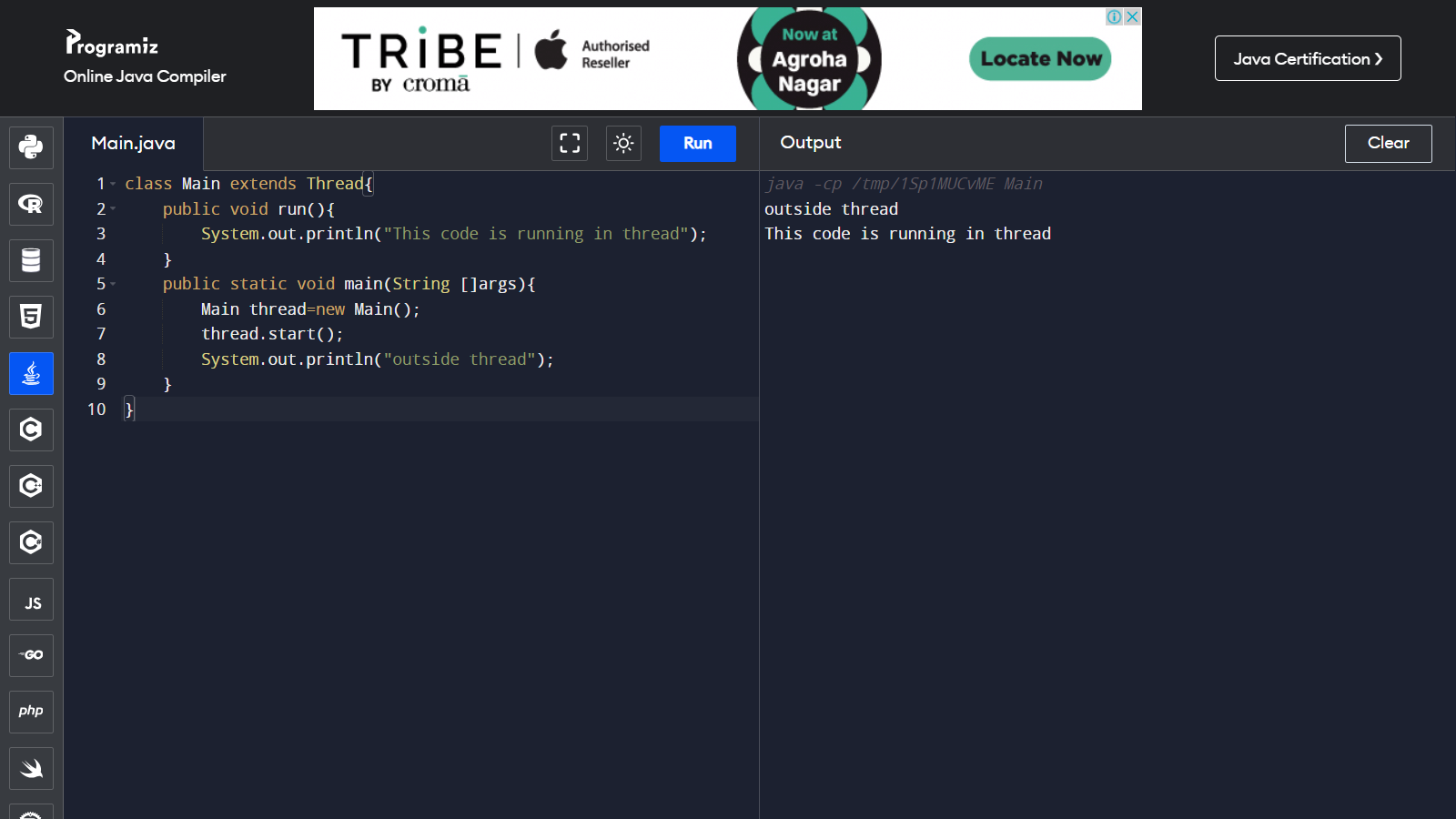
thread.start();

System.out.println("outside thread");

}

}

USING INHERITANCE



public class Main implements Runnable {

public static void main(String[] args) {

Main obj = new Main();

Thread thread = new Thread(obj);

thread.start();

System.out.println("This code is outside of the thread");

}

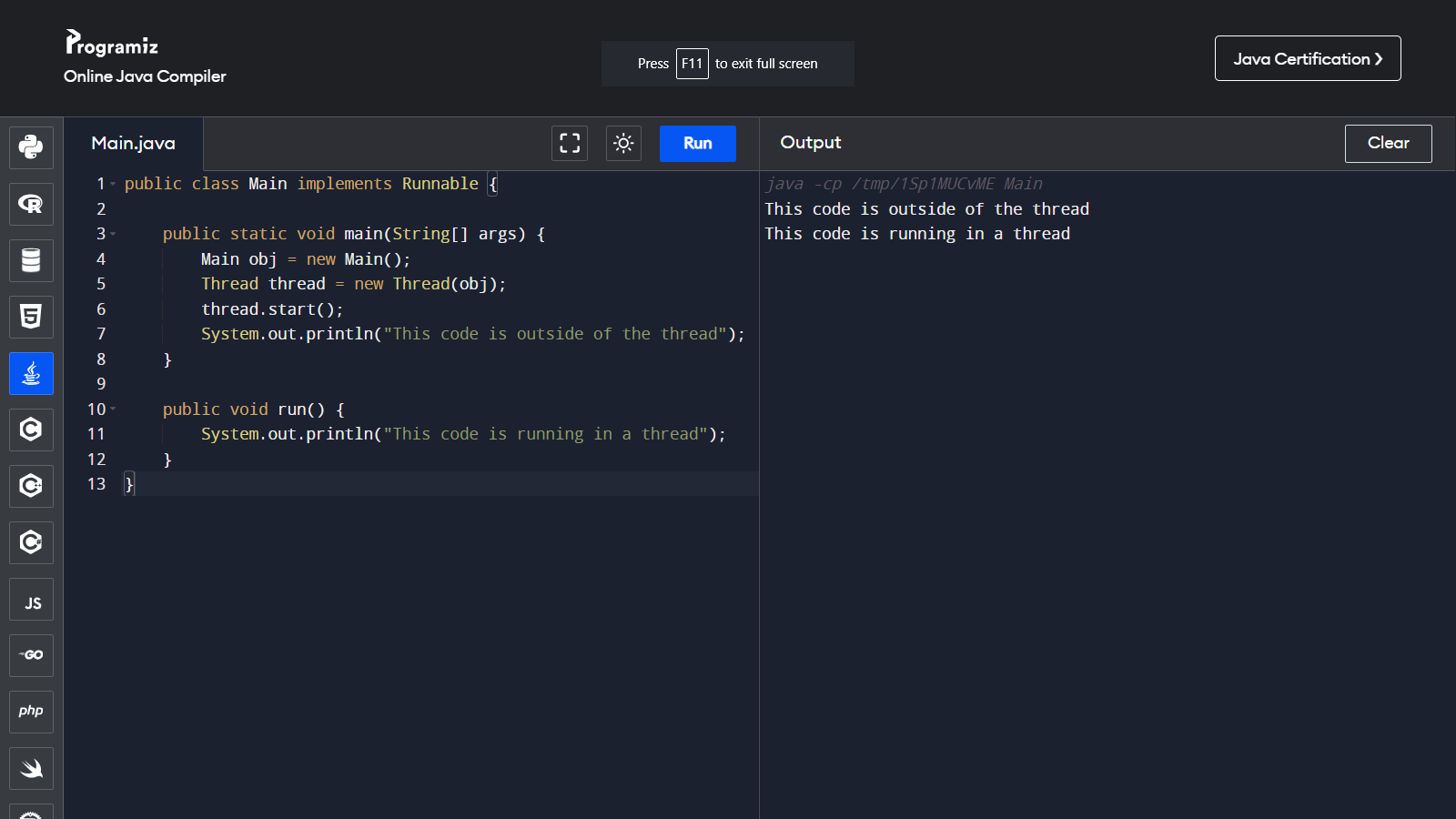
public void run() {

System.out.println("This code is running in a thread");

}

}

USING INTERFACE



public class Main extends Thread {

public void run() {

System.out.println("Welcome To CSIT");

}

public static void main(String[] args) {

Main t1 = new Main();

Main t2 = new Main();

Main t3 = new Main();

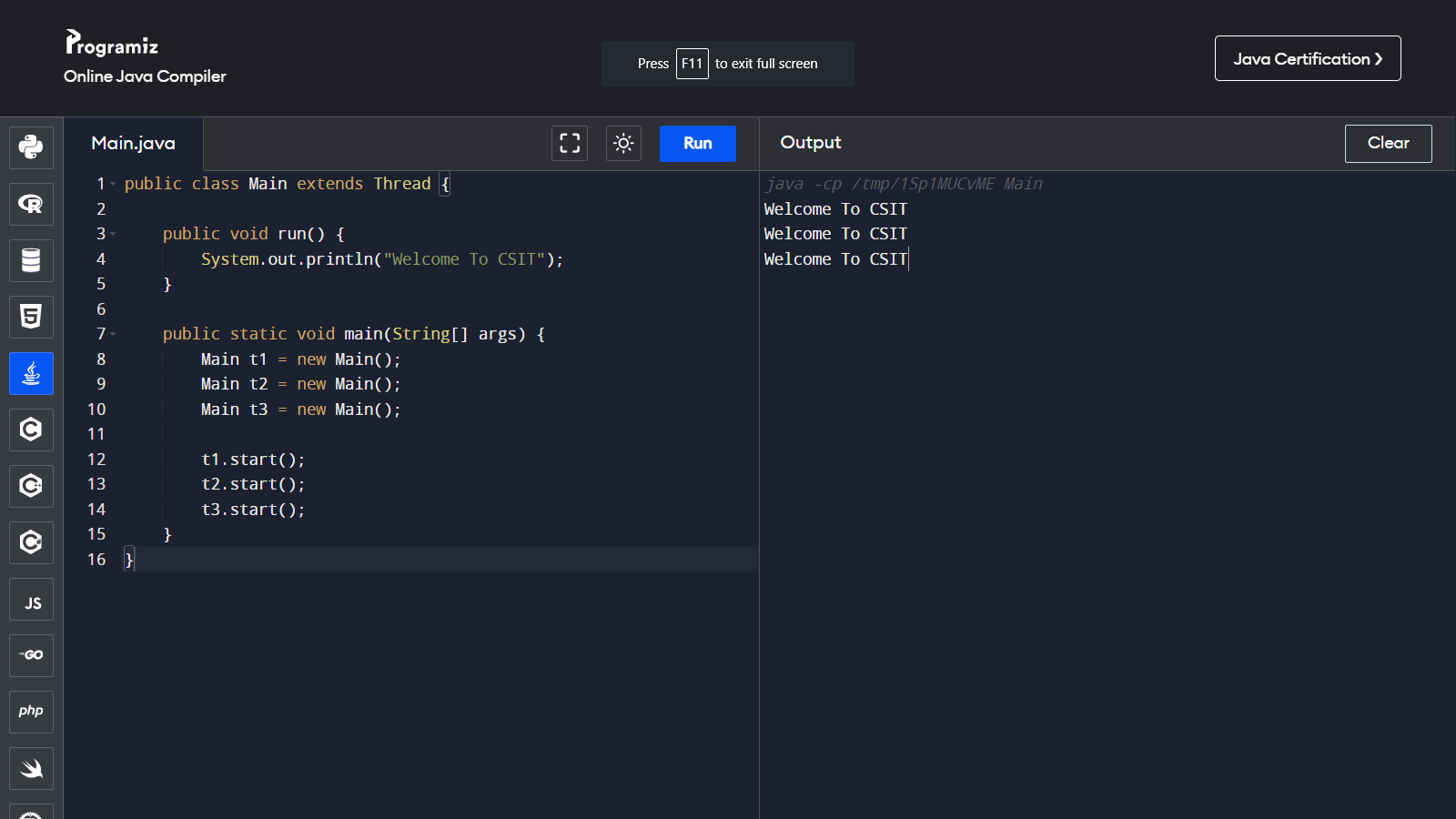
t1.start();

t2.start();

t3.start();

}

}



public class Main implements Runnable{

public void run() {

System.out.println("Welcome To CSIT");

}

public static void main(String[] args) {

Main t1 = new Main();

Thread n1=new Thread(t1);

Main t2 = new Main();

Thread n2=new Thread(t2);

Main t3 = new Main();

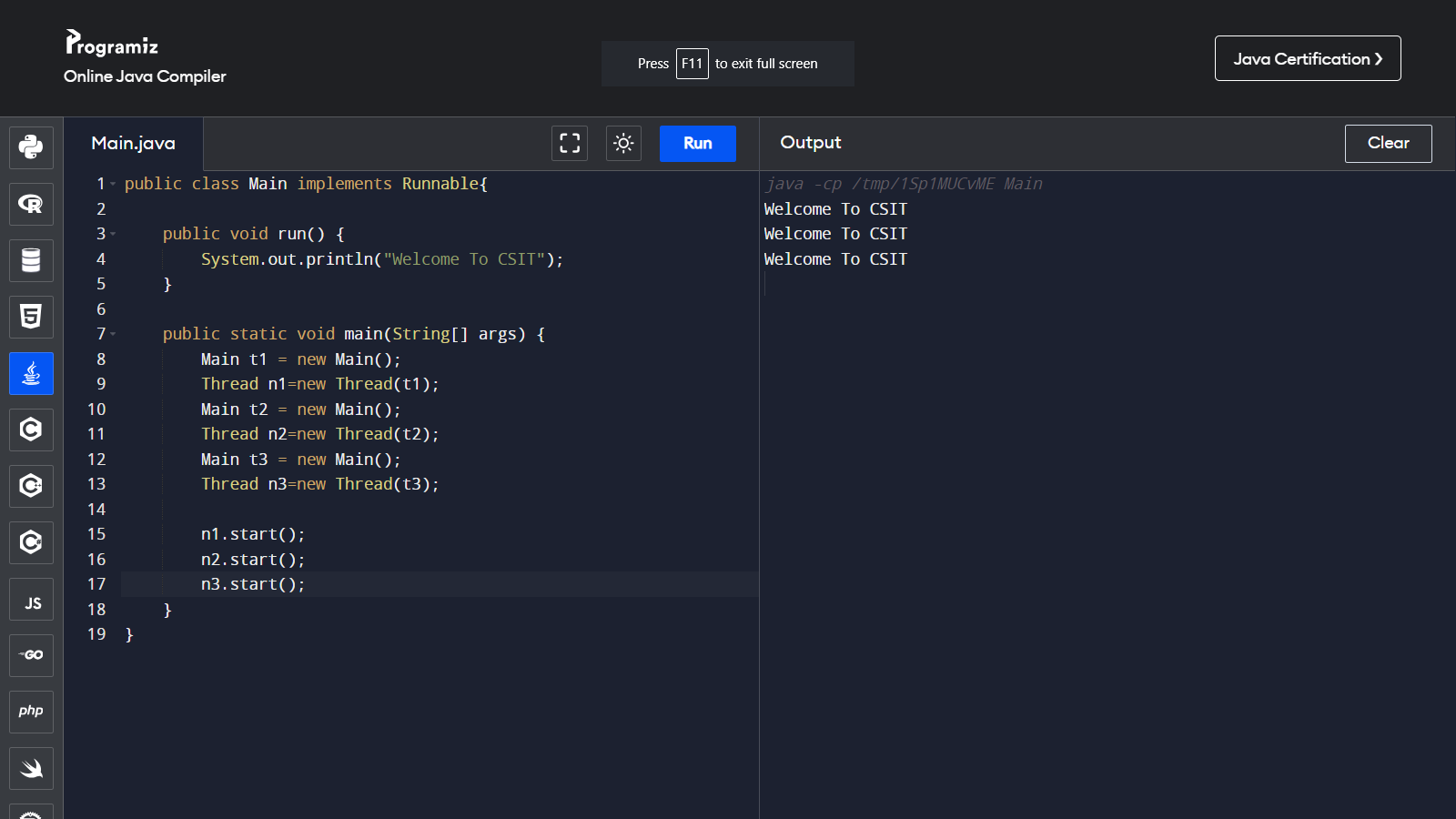
Thread n3=new Thread(t3);

n1.start();

n2.start();

n3.start();

}}



public class MultithreadingDemo implements Runnable {

public void run() {

try {

System.out.println("Thread " + Thread.currentThread().getId() + " is running by Java");

} catch (Exception e) {

System.out.println("Exception is caught here");

}

}

public static void main(String[] args) {

int n = 5;

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

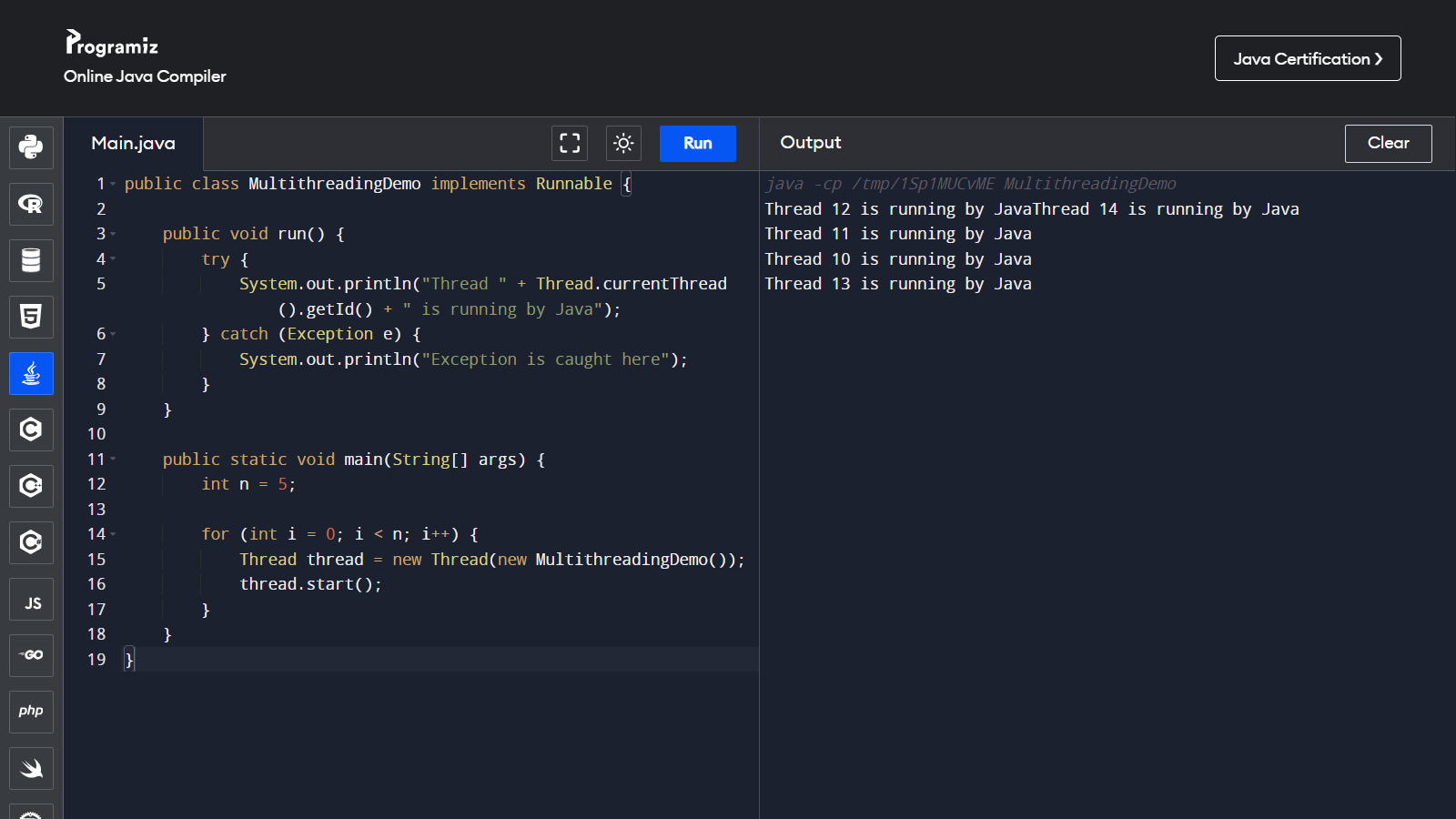
Thread thread = new Thread(new MultithreadingDemo());

thread.start();

}

}

}



class TablePrinter extends Thread {

private int start;

private int end;

public TablePrinter(int start, int end) {

this.start = start;

this.end = end;

}

public void run() {

for (int i = start; i <= end; i++) {

System.out.println("2 \* " + i + " = " + (2 \* i));

}

}

public static void main(String[] args) {

TablePrinter thread1 = new TablePrinter(1, 5);

TablePrinter thread2 = new TablePrinter(6, 10);

thread1.start();

thread2.start();

try {

thread1.join();

thread2.join();

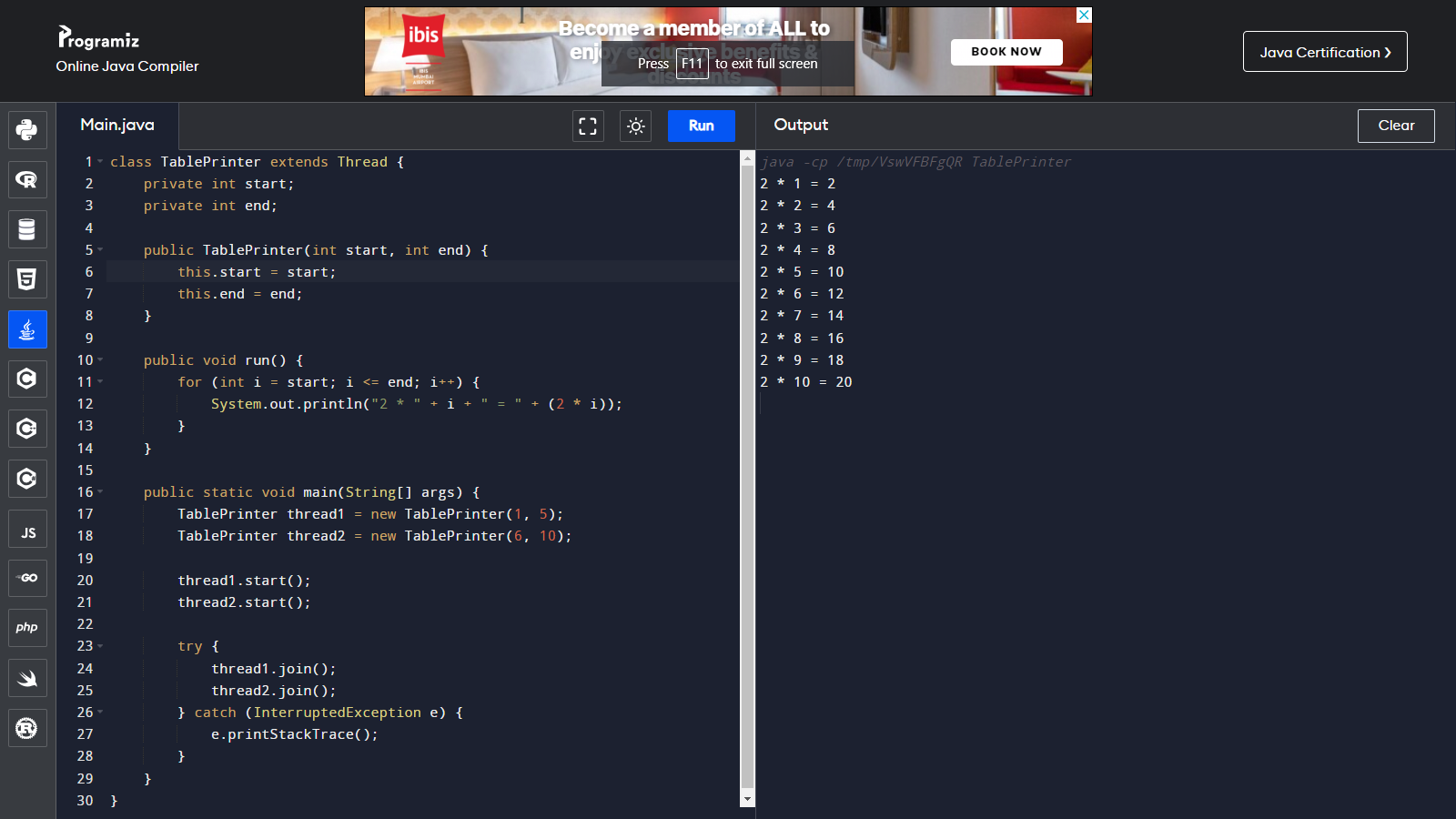
} catch (InterruptedException e) {

e.printStackTrace();

}

}

}



public class Main {

public static void main(String[] args) {

for (int i=1; i<=10; i++){

Calculator calculator=new Calculator(i);

Thread thread=new Thread(calculator);

thread.start();

}

}

}

class Calculator implements Runnable {

private int number;

public Calculator(int number) {

this.number=number;

}

public void run() {

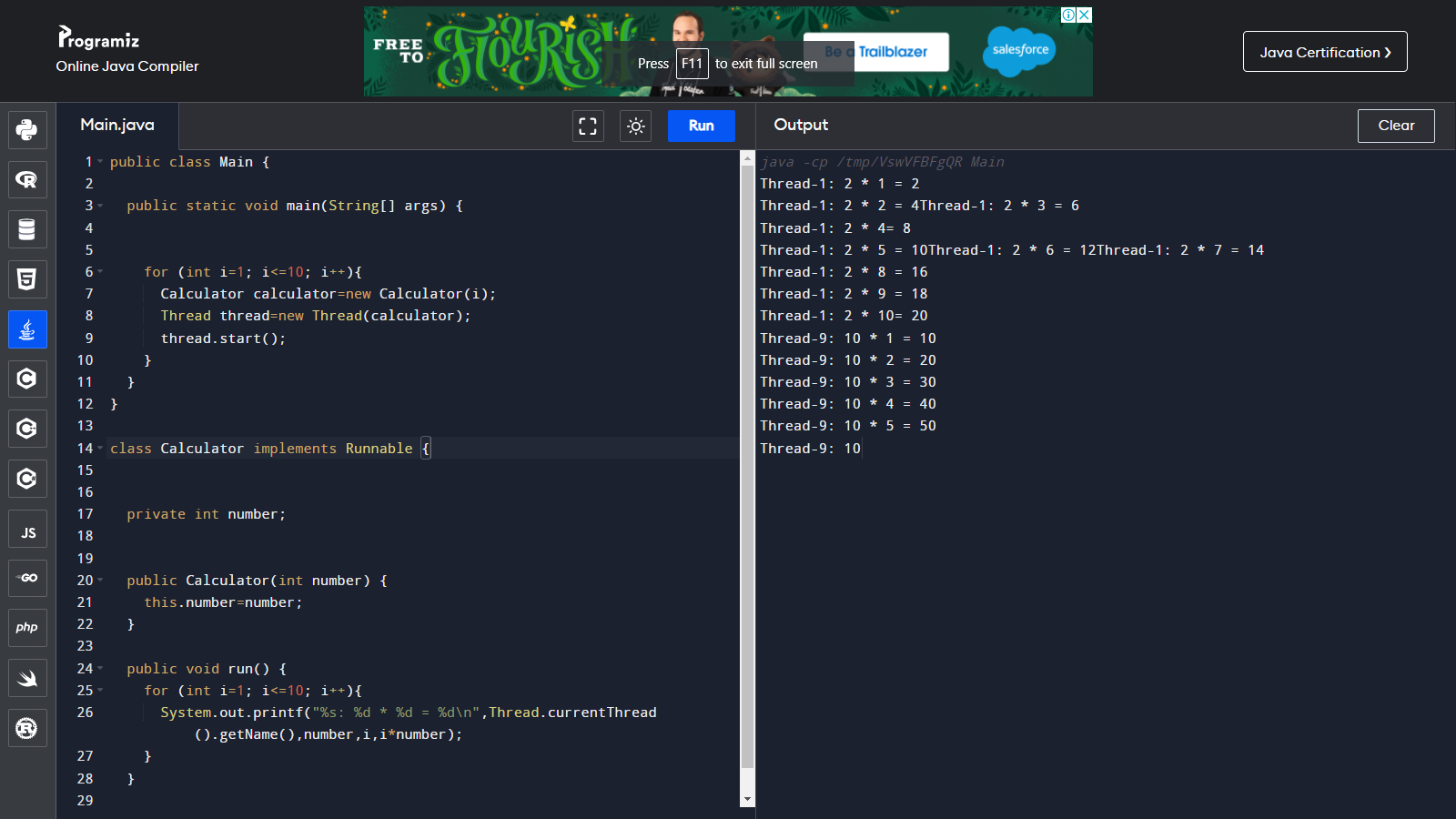
for (int i=1; i<=10; i++){

System.out.printf("%s: %d \* %d = %d\n",Thread.currentThread().getName(),number,i,i\*number);

}

}

}



class ThreadCount extends Thread {

ThreadCount(String s) {

super(s);

System.out.println("New thread created: " + s);

}

public void run() {

try {

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

System.out.println("New thread created \* d ^ (- t) \* hicl");

Thread.sleep(1500);

}

} catch (InterruptedException e) {

System.out.println("Currently executing thread is interrupted");

}

System.out.println("Currently executing thread run is terminated");

}

public static void main(String[] args) {

ThreadCount thread = new ThreadCount("Thread");

thread.start();

}

}

