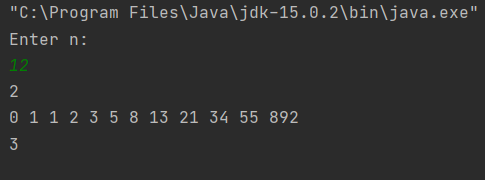
LAB 8

THREADS

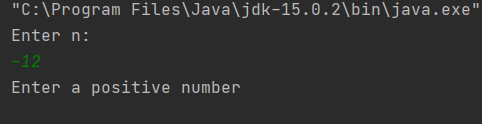
1.

package com.company;  
import com.sun.jdi.InvalidLineNumberException;  
  
import java.util.Scanner;  
  
class MyThread1 extends Thread{  
 int count;  
 public void fib(){  
 int n1=0,n2=1,n3,i;  
 System.*out*.print(n1+" "+n2);  
  
 for(i=2;i<count;++i)  
 {  
 n3=n1+n2;  
 System.*out*.print(" "+n3);  
 n1=n2;  
 n2=n3;  
 }  
 }  
  
 MyThread1(int count){  
 this.count=count;  
 }  
 public void run(){  
 fib();  
 }  
  
}  
class MyThread2 extends Thread{  
  
 int number;  
 public void prime(){  
 try {  
 for (int i = 2; i < number; i++) {  
 while (number % i == 0) {  
 System.*out*.println(i + " ");  
 Thread.*sleep*(500);  
 number = number / i;  
 }  
 }  
 if (number > 2) {  
 System.*out*.println(number);  
 }  
 }  
 catch(Exception e){  
 System.*out*.println(e);  
 }  
 }  
 MyThread2(int number){  
 this.number=number;  
 }  
 public void run(){  
 prime();  
 }  
}  
  
class Main{  
 public static void main(String args[]){  
 Scanner in= new Scanner(System.*in*);  
 try{  
 System.*out*.println("Enter n: ");  
 int n=in.nextInt();  
 //System.out.println("Thread1: ");  
 MyThread1 t1=new MyThread1(n);  
 //System.out.println("Thread2: ");  
 MyThread2 t2=new MyThread2(n);  
 t1.start();  
 t2.start();  
 }  
  
  
 }  
}



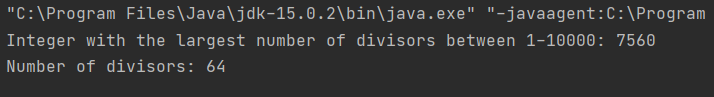
3.

package com.company;  
import com.sun.jdi.InvalidLineNumberException;  
  
import java.util.Scanner;  
  
class MyThread1 extends Thread{  
 int count;  
 public void fib(){  
 int n1=0,n2=1,n3,i;  
 System.*out*.print(n1+" "+n2);  
  
 for(i=2;i<count;++i)  
 {  
 n3=n1+n2;  
 System.*out*.print(" "+n3);  
 n1=n2;  
 n2=n3;  
 }  
 }  
  
 MyThread1(int count){  
 this.count=count;  
 }  
 public void run(){  
 fib();  
 }  
  
}  
class MyThread2 extends Thread{  
  
 int number;  
 public void prime(){  
 try {  
 for (int i = 2; i < number; i++) {  
 while (number % i == 0) {  
 System.*out*.println(i + " ");  
 Thread.*sleep*(500);  
 number = number / i;  
 }  
 }  
 if (number > 2) {  
 System.*out*.println(number);  
 }  
 }  
 catch(Exception e){  
 System.*out*.println(e);  
 }  
 }  
 MyThread2(int number){  
 this.number=number;  
 }  
 public void run(){  
 prime();  
 }  
}  
  
class Main{  
 public static void main(String args[]){  
 Scanner in= new Scanner(System.*in*);  
 try{  
 System.*out*.println("Enter n: ");  
 int n=in.nextInt();  
 if(n<=0){  
 throw new Exception();  
 }  
 //System.out.println("Thread1: ");  
 MyThread1 t1=new MyThread1(n);  
 //System.out.println("Thread2: ");  
 MyThread2 t2=new MyThread2(n);  
 t1.start();  
 t2.start();  
 }  
 catch(Exception e){  
 System.*out*.println("Enter a positive number");  
 }  
  
 }  
}



4.

package com.company;  
import java.util.Scanner;  
import java.util.Arrays;  
public class random  
{  
 public static void main(String[] args)  
 {  
 Scanner in=new Scanner(System.*in*); int count[]=new int[10000];  
 int number[]=new int[10000]; int temp,y=0;  
 for(int i=2;i<10000;i++)  
 {  
 for(int n=1;n<=i;n++)  
 {  
 if(i%n==0)  
 {  
 count[i-2]+=1;  
 }  
 }  
 number[i-2]=i;  
 }  
 for(int a=0;a<count.length;a++)  
 {  
 for(int b=a+1;b<count.length;b++)  
 {  
 int tmp = 0; if(count[a]<count[b])  
 {  
 tmp=count[a]; count[a]=count[b]; count[b]=tmp; tmp=number[a]; number[a]=number[b]; number[b]=tmp;  
 }  
 }  
 }  
  
 System.*out*.println("Integer with the largest number of divisors between 1-10000: "+number[0]); System.*out*.println("Number of divisors: "+count[0]);  
 in.close();  
 }  
}



5.

package com.company;  
  
import java.util.Scanner;  
public class DivC  
{  
 private final static int *MAX* = 100000;  
 private volatile static int *maxDivisorCount* = 0; private volatile static int *intWithMaxDivisorCount*;  
 synchronized private static void report(int maxCountFromThread, int intWithMaxFromThread) {  
 if (maxCountFromThread > *maxDivisorCount*) { *maxDivisorCount* = maxCountFromThread; *intWithMaxDivisorCount* = intWithMaxFromThread;  
 }  
 }  
 private static class CountDivisorsThread extends Thread { int min, max;  
 public CountDivisorsThread(int min, int max) { this.min = min;  
 this.max = max;  
 }  
 public void run() { int maxDivisors = 0; int whichInt = 0;  
 for (int i = min; i < max; i++) { int divisors = *countDivisors*(i); if (divisors > maxDivisors) { maxDivisors = divisors; whichInt = i;  
 }  
  
 }  
 *report*(maxDivisors,whichInt);  
 }  
 }  
 private static void countDivisorsWithThreads(int numberOfThreads) { System.*out*.println("\nCounting divisors using " +  
 numberOfThreads + " threads...");  
 long startTime = System.*currentTimeMillis*();  
 CountDivisorsThread[] worker = new CountDivisorsThread[numberOfThreads]; int integersPerThread = *MAX*/numberOfThreads;  
 int start = 1;  
 int end = start + integersPerThread - 1; for (int i = 0; i < numberOfThreads; i++) { if (i == numberOfThreads - 1) {  
 end = *MAX*;  
 }  
 worker[i] = new CountDivisorsThread( start, end ); start = end+1;  
 end = start + integersPerThread - 1;  
 }  
 *maxDivisorCount* = 0;  
 for (int i = 0; i < numberOfThreads; i++) worker[i].start();  
 for (int i = 0; i < numberOfThreads; i++) { while (worker[i].isAlive()) {  
 try { worker[i].join();  
 }  
 catch (InterruptedException e) {  
 }  
 }  
 }  
 long elapsedTime = System.*currentTimeMillis*() - startTime; System.*out*.println("\nThe largest number of divisors " +  
 "for numbers between 1 and " + *MAX* + " is " + *maxDivisorCount*); System.*out*.println("An integer with that many divisors is " + *intWithMaxDivisorCount*);  
 System.*out*.println("Total elapsed time: " + (elapsedTime/1000.0) + " seconds.\n");  
 }  
 public static void main(String[] args) { Scanner in = new Scanner(System.*in*); int numberOfThreads = 0;  
 while (numberOfThreads < 1 || numberOfThreads > 10) {  
  
 System.*out*.print("number of threads? "); numberOfThreads = in.nextInt();  
 if (numberOfThreads < 1 || numberOfThreads > 10) System.*out*.println("enter a number from 1 to 10");  
 }  
 *countDivisorsWithThreads*(numberOfThreads);  
 }  
 public static int countDivisors(int N) { int count = 0;  
 for (int i = 1; i <= N ; i++) { if ( N % i == 0 )  
 count ++;  
 }  
 return count;  
 }  
  
}

