Name:Thota GuruTheja Reddy

Regno:19BCD7034

Slot:L1

Lab: 8

1. Write a multi-threaded Java program to print all numbers below 100 that are both prime and Fibonacci number (some examples are 2, 3, 5, 13, etc.). Design a thread that generates prime numbers below 100 and writes them into a pipe. Design another thread that generates Fibonacci numbers and writes them to another pipe. The main thread should read both the pipes to identify numbers common to both.

Ans:

import java.util.\*;

import java.io.\*;

class Fibonacci extends Thread

{

private PipedWriter out = new PipedWriter();

public PipedWriter getPipedWriter()

{

return out;

}

public void run()

{

Thread t = Thread.currentThread();

int fibo1=0,fibo2=1,fibo=0;

while(true)

{

try

{

fibo = fibo1 + fibo2;

if(fibo>100)

{

out.close();

break;

}

out.write(fibo);

sleep(100);

}

catch(Exception e)

{

System.out.println("Fibonacci:"+e);

}

fibo1=fibo2;

fibo2=fibo;

}

}

}

class Prime extends Thread

{

private PipedWriter out1 = new PipedWriter();

public PipedWriter getPipedWriter()

{

return out1;

}

public void run()

{

Thread t= Thread.currentThread();

int prime=1;

while(true)

{

try

{

if(prime>100)

{

out1.close();

break;

}

if(isPrime(prime))

out1.write(prime);

prime++;

sleep(0);

}

catch(Exception e)

{

System.exit(0);

}

}

}

public boolean isPrime(int n)

{

int m=(int)Math.round(Math.sqrt(n));

if(n==1 || n==2)

return true;

for(int i=2;i<=m;i++)

if(n%i==0)

return false;

return true;

}

}

public class Main

{

public static void main(String[] args) throws Exception

{

Thread t=Thread.currentThread();

Fibonacci fibonacci = new Fibonacci();

Prime prime = new Prime();

PipedReader fpr = new PipedReader(fibonacci.getPipedWriter());

PipedReader ppr = new PipedReader(prime.getPipedWriter());

fibonacci.start();

prime.start();

int fib=fpr.read(), prm=ppr.read();

System.out.println("Numbers both Fibonacci and prime below 100 are: ");

while((fib!=-1) && (prm!=-1))

{

while(prm<=fib)

{

if(fib==prm)

System.out.println(prm);

prm=ppr.read();

}

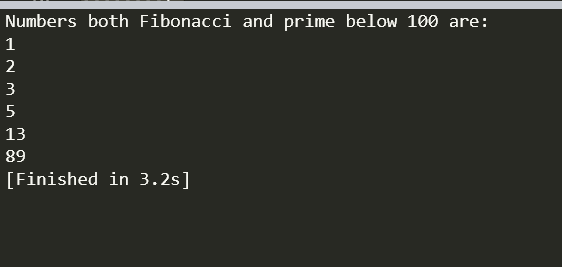
fib=fpr.read();

}

}

}

Output:



2. Create a class “A” with two methods, “arms” to check whether a given number is Armstrong number (sum of cube of individual digit is the same as the number) or not and “multiarray” ( multiply two consecutive array elements and for the last number again multiply with first number and print the value). Create 3 threads to perform the following operation simultaneously

1.multiply two consecutive array elements and print it. Eg. Input:(1 2 3 4 5) output: (2 6 12 20 5)

2. check whether the given number 153 is Armstrong number or not. 3.check whether the given number 120 is Armstrong number or not

Ans:

import java.util.Scanner;

class A extends Thread{

int n,m;

int a[];

public void arms(int n){

this.n=n;

int c=0,a,temp;

temp=n;

while(n>0){

a=n%10;

n=n/10;

c=c+(a\*a\*a);

}

if(temp==c){

System.out.println("armstrong number");

}else{

System.out.println("Not armstrong number");

}

}

public void multiarray(int a[]){

this.a=a;

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

if(i==a.length-1){

m=a[i]\*a[0];

}else{

m=a[i]\*a[i+1];

}

System.out.println(m);

}

}

}

class Thread1 extends A{

int b[];

Thread1(int[]b){

this.b=b;

}

public void run(){

super.multiarray(b);

}

}

class Thread2 extends A{

int g;

Thread2(int g){

this.g=g;

}

public void run(){

super.arms(g);

}

}

class Main{

public static void main(String[] args) {

int[]f={1,3,5,6};

Thread1 t1 = new Thread1(f);

Thread2 t2 = new Thread2(153);

Thread2 t3 = new Thread2(120);

t2.setPriority(10);

t3.setPriority(1);

t1.start();

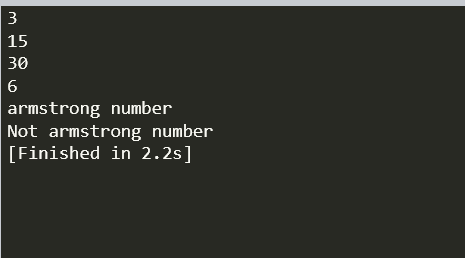
t2.start();

t3.start();

}

}

Output:



1. 3. Create a class with method to count odd and even number in 3X3 matrix. Create two threads to perform the operation concurrently. Display the odd and even number count in the matrix.(use Runnable interface)

Original array:

4 1 3

3 5 7

8 2 6

Frequency of odd numbers: 5

Frequency of even numbers: 4

Ans:

import java.util.Scanner;

class counte implements Runnable{

int arr[][];

public counte(int a[][]) {

this.arr=a;

}

public void run() {

int c=0;

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

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

if(arr[i][j]%2==0) {

c++;

}

}

}

System.out.println("Total frequency of even numbers is "+c);

}

}

class counto implements Runnable{

int arr[][];

public counto(int a[][]) {

this.arr=a;

}

public void run() {

int c=0;

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

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

if(arr[i][j]%2!=0) {

c++;

}

}

}

System.out.println("Total frequency of odd numbers is "+c);

}

}

public class Main

{

public static void main(String[] args) {

Scanner in=new Scanner(System.in);

System.out.println("Enter the size of matrix");

int n=in.nextInt();

int a[][]= new int[n][n];

System.out.println("Enter elements in matrix");

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

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

a[i][j]=in.nextInt();

}

}

counto o=new counto(a);

counte e=new counte(a);

Thread t1=new Thread(o);

t1.start();

Thread t2=new Thread(e);

t2.start();

}

}

Output:

