**Assignment No:-40**

Name:-Suryawanshi Sangramsingh Sambhaji

Batch: - Delta - DCA (Java) 2024 Date:-3/7/2024

**1. Write a program that creates two threads, one of which counts up to 100 and the other counts down from 100. Print out the values each thread counts to.**

package AssignmentNo41;

class AscendingCount extends Thread

{

public void run()

{

System.***out***.println("\n"+Thread.*currentThread*().getName()+" : ");

for(int i=1;i<=100;i++)

{

System.***out***.print(i+" ");

}

}

}

class DescendingCount extends Thread

{

public void run()

{

System.***out***.println("\n\n"+Thread.*currentThread*().getName()+" : ");

for(int i=100;i>=1;i--)

{

System.***out***.print(i+" ");

}

}

}

public class Count

{

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

{

AscendingCount t1 = new AscendingCount();

DescendingCount t2 = new DescendingCount();

t1.start();

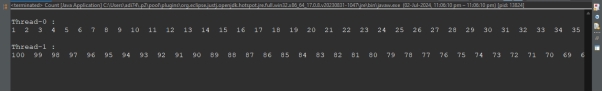
t1.join();

t2.start();

t2.join();

}

}



**2. Write a program that uses multiple threads to calculate the sum of an array of numbers. Each thread should sum a portion of the array, and then the main thread should add up the individual sums to get the total sum.**

package AssignmentNo41;

import java.util.Scanner;

class Sum

{

static int *sum*=0;

public void getSum(int s,int e, int a[])

{

int sum=0;

for(int i=s;i<e;i++)

{

sum+=a[i];

}

System.***out***.println("\n"+Thread.*currentThread*().getName()+" : sum = "+sum);

this.*sum*+=sum;

}

}

public class SumArray extends Thread

{

int a[];

int st;int end;

SumArray(int st, int end, int[]a)

{

this.a=a;

this.st = st;

this.end = end;

}

public void run()

{

Sum s = new Sum();

s.getSum(st, end, a);

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter array size : ");

int s = sc.nextInt();

int a[] = new int [s];

System.***out***.print("Enter array elements : ");

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

a[i] = sc.nextInt();

SumArray t1 = new SumArray(0, a.length/2, a);

SumArray t2 = new SumArray((a.length/2), a.length, a);

try

{

t1.start();

t1.join();

t2.start();

t2.join();

}

catch(InterruptedException w)

{

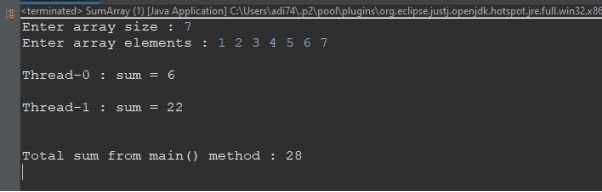
System.***out***.println(w);

}

System.***out***.println("\n\nTotal sum from main() method : "+Sum.*sum*);

}

}



**3. Write a program that simulates a traffic intersection using threads. Each direction of traffic should be represented by a thread, and the threads should coordinate to avoid collisions.**

package AssignmentNo41;

class Traffic

{

public synchronized void showDetails(String d)

{

try

{

Thread.*sleep*(2000);

}

catch(InterruptedException e)

{

System.***out***.println(e);

}

System.***out***.println("\n"+d+" side vehicles are passing.Please wait.");

}

}

public class TrafficIntersection extends Thread

{

String dir;

TrafficIntersection(String dir)

{

this.dir = dir;

}

public void run()

{

Traffic t = new Traffic();

t.showDetails(dir);

}

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

{

TrafficIntersection n = new TrafficIntersection("North");

TrafficIntersection s = new TrafficIntersection("South");

TrafficIntersection e = new TrafficIntersection("East");

TrafficIntersection w = new TrafficIntersection("West");

n.start();

n.join();

s.start();

s.join();

e.start();

e.join();

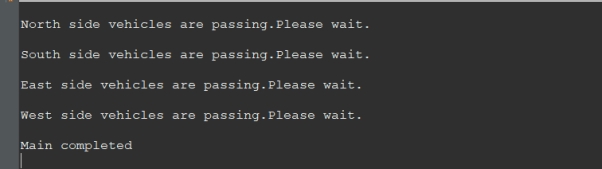
w.start();

w.join();

System.***out***.println("\nMain completed");

}

}



**4. Write a program that creates a thread that calculates the factorial of a number. The main thread should wait for the thread to finish and then print out the result.**

package AssignmentNo41;

import java.util.Scanner;

class FactThread extends Thread

{

int n;

FactThread(int n)

{

this.n = n;

}

public void run()

{

System.***out***.println(Thread.*currentThread*().getName());

int fact =1;

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

{

fact \*= i;

}

System.***out***.println("Factorial is : "+fact);

}

}

public class Factorial {

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter a number : ");

int n = sc.nextInt();

FactThread t1 = new FactThread(n);

try

{

t1.start();

t1.join();

}

catch(InterruptedException e)

{

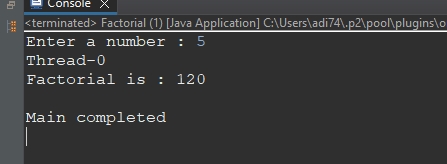
System.***out***.println(e);

}

System.***out***.println("\nMain completed");

}

}



**5. Write a program that simulates a bank account using threads. Multiple threads should be able to withdraw and deposit money concurrently, but the balance of the account should always be accurate.**

package AssignmentNo41;

import java.util.Scanner;

class BankThread

{

static int *bal*;

public synchronized void deposit(int a)

{

System.***out***.println("\*\*\*\*\*\* "+Thread.*currentThread*().getName()+" \*\*\*\*\*\*");

System.***out***.println("Deposit completed..");

*bal* = *bal*+a;

System.***out***.println("Current balance : "+*bal*);

}

public synchronized void withdraw(int a)

{

System.***out***.println("\n\*\*\*\*\*\* "+Thread.*currentThread*().getName()+" \*\*\*\*\*\*");

if(*bal* >= a)

{

System.***out***.println("Withdraw succcessfull.");

*bal*-=a;

System.***out***.println("Current balance : "+*bal*);

}

else

{

System.***out***.println("Insuffienct balance.");

}

}

}

public class Bank extends Thread

{

Scanner sc;

static Bank *b*;

static int *dep*;

static int *wit*;

public void getDeposit()

{

sc = new Scanner(System.***in***);

System.***out***.println("Enter amount to deposit : ");

int d = sc.nextInt();

this.*dep* = d;

}

public void getWithdraw()

{

sc = new Scanner(System.***in***);

System.***out***.println("Enter amount to withdraw : ");

int w = sc.nextInt();

this.*wit* = w;

}

public void run()

{

BankThread bt = new BankThread();

sc = new Scanner(System.***in***);

System.***out***.println("\nPress 1 to deposit and 2 to withdraw : ");

int ch = sc.nextInt();

*b* = new Bank();

switch(ch)

{

case 1 : *b*.getDeposit();

bt.deposit(this.*dep*);

break;

case 2: *b*.getWithdraw();

bt.withdraw(this.*wit*);

break;

default : System.***out***.println("Wrong input");

}

}

public static void main(String[] args)

{

Bank t1 = new Bank();

Bank t2 = new Bank();

Bank t3 = new Bank();

try

{

t1.start();

t1.join();

t2.start();

t2.join();

t3.start();

t3.join();

}

catch(InterruptedException e)

{

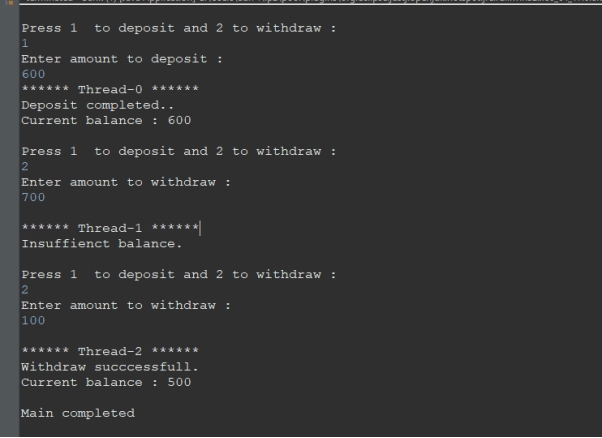
System.***out***.println(e);

}

System.***out***.println("\nMain completed");

}

}



**6. Write a program that uses threads to calculate the Fibonacci sequence. Each thread should calculate a portion of the sequence, and then the main thread should combine the results to get the full sequence.**

package AssignmentNo41;

import java.util.Scanner;

class Fibo extends Thread

{

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

int n1;int n2;

Fibo(int a, int b)

{

this.n1=a;

this.n2=b;

}

public void run()

{

for(int i=n1;i<=n2/2;i++)

{

System.***out***.print(*a*+" ");

*c* = *a*+*b*;

*a* = *b*;

*b* = *c*;

}

System.***out***.println();

}

}

public class FiboSeries

{

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

{

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter start and end range : ");

int a = sc.nextInt();

int b = sc.nextInt();

Fibo t1 = new Fibo(a,b);

Fibo t2 = new Fibo(a,b);

System.***out***.println("\nFibonacii sereis : ");

t1.start();

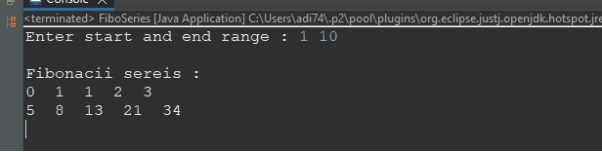
t1.join();

t2.start();

t2.join();

}

}



**7. Write a program that uses threads to calculate the sum of the digits of a large number. Each thread should sum a portion of the digits, and then the main thread should combine the results.**

package AssignmentNo41;

import java.util.Scanner;

class Digit extends Thread

{

String num;

static int *sum*;

int st;int ed;

Digit(String s, int st, int ed)

{

this.num = s;

this.st = st;

this.ed = ed;

}

public void run()

{

int sum=0;

for(int i=st;i<ed;i++)

{

sum+= Character.*getNumericValue*(num.charAt(i));

}

System.***out***.println(Thread.*currentThread*().getName()+" sum : "+sum);

this.*sum*+=sum;

}

}

public class DigitSum {

public static void main(String[] args)

{

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter a number : ");

String s = sc.next();

Digit t1 = new Digit(s, 0, s.length()/2);

Digit t2 = new Digit(s, s.length()/2, s.length());

try

{

t1.start();

t1.join();

t2.start();

t2.join();

}

catch(InterruptedException e)

{

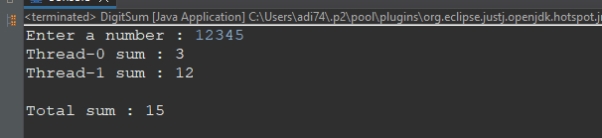
System.***out***.println(e);

}

System.***out***.println("\nTotal sum : "+Digit.*sum*);

}

}



**8. Write a program that creates a thread that generates random numbers. The main thread should wait for the thread to finish and then print out the generated numbers.**

package AssignmentNo41;

import java.util.Random;

class RandomNum extends Thread

{

public void run()

{

Random rd = new Random();

int num = rd.nextInt(10);

System.***out***.println("Random number from "+Thread.*currentThread*().getName()+" is : "+num);

}

}

public class RandomNumbers {

public static void main(String[] args) {

RandomNum t1 = new RandomNum();

RandomNum t2 = new RandomNum();

RandomNum t3 = new RandomNum();

try

{

t1.start();

t1.join();

t2.start();

t2.join();

t3.start();

t3.join();

}

catch(InterruptedException e)

{

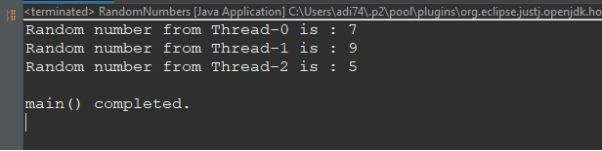
System.***out***.println(e);

}

System.***out***.println("\nmain() completed.");

}

}



**9. Write a Java program that creates two threads to find and print even and odd numbers from 1 to 20.**

package AssignmentNo41;

import java.util.Scanner;

class Even extends Thread

{

int a;int b;

Even(int a,int b)

{

this.a=a;

this.b=b;

}

public void run()

{

System.***out***.println("\nEven numbers ");

for(int i=a;i<=b;i++)

{

if(i%2==0)

{

System.***out***.print(i+" ");

}

}

System.***out***.println();

}

}

class Odd extends Thread

{

int a;int b;

Odd(int a,int b)

{

this.a=a;

this.b=b;

}

public void run()

{

System.***out***.println("\nOdd numbers ");

for(int i=a;i<=b;i++)

{

if(i%2!=0)

{

System.***out***.print(i+" ");

}

}

System.***out***.println();

}

}

public class EvenOdd {

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

{

Scanner sc = new Scanner(System.***in***);

System.***out***.println("Enter first and last number : ");

int a = sc.nextInt();

int b = sc.nextInt();

Even t1 = new Even(a, b);

Odd t2 = new Odd(a, b);

t1.start();

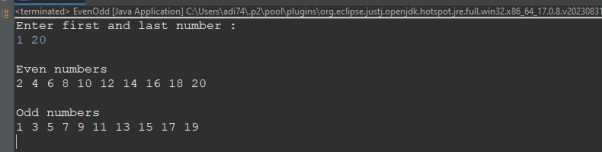
t1.join();

t2.start();

t2.join();

}

}



**10. Write a Java program that sorts an array of integers using multiple threads.**

package AssignmentNo41;

import java.util.Arrays;

import java.util.Scanner;

public class ArraySort extends Thread

{

static int a[];

int st;int end;

ArraySort(int st, int end, int[]a)

{

this.a=a;

this.st = st;

this.end = end;

}

public void run()

{

for(int i=st;i<end;i++)

{

for(int j=i+1;j<end;j++)

{

if(a[i] > a[j])

{

int t = a[i];

a[i] = a[j];

a[j] = t;

}

}

}

System.out.println("\n"+Thread.currentThread().getName());

System.out.println(Arrays.toString(a));

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter array size : ");

int s = sc.nextInt();

int a[] = new int [s];

System.out.print("Enter array elements : ");

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

a[i] = sc.nextInt();

ArraySort t1 = new ArraySort(0,a.length/2,a);

ArraySort t2 = new ArraySort(a.length/2, a.length, a);

try

{

t1.start();

t1.join();

t2.start();

t2.join();

}

catch(InterruptedException w)

{

System.out.println(w);

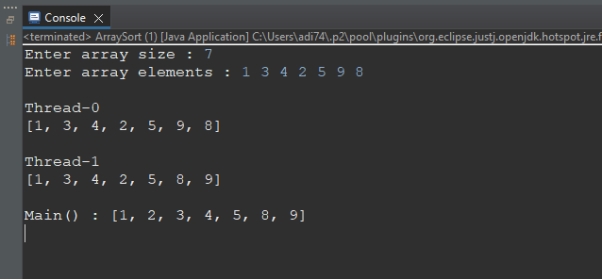
}

Arrays.sort(ArraySort.a);

System.out.println("\nMain() : "+Arrays.toString(ArraySort.a));

}

}



**11. Write a Java program that performs matrix multiplication using multiple threads.**

**12. Write a Java program that calculates the sum of all prime numbers up to a given limit using multiple threads.**

package AssignmentNo41;

import java.util.Scanner;

class CalculatePrime

{

static int *sum*=0;

public void getPrime(int f, int l)

{

int sum=0;

System.***out***.println("\nPrime Series : ");

for(int i=f;i<=l;i++)

{

if(*isPrime*(i))

{

System.***out***.print(i+" ");

sum+=i;

}

}

System.***out***.println("\n"+Thread.*currentThread*().getName()+" sum = "+sum);

this.*sum* += sum;

}

public static boolean isPrime(int n)

{

int cnt=0;

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

{

if(n%i==0)

{

cnt++;

}

}

if(cnt==2)

return true;

else

return false;

}

}

public class PrimeSeriesSync extends Thread

{

int f;int l;

public void run()

{

CalculatePrime obj = new CalculatePrime();

obj.getPrime(f, l);

}

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

{

Scanner sc = new Scanner(System.***in***);

System.***out***.println("Enter first number : ");

int f = sc.nextInt();

System.***out***.println("Enter last number : ");

int l = sc.nextInt();

PrimeSeriesSync t1 = new PrimeSeriesSync();

PrimeSeriesSync t2 = new PrimeSeriesSync();

t1.f=f;

t1.l = (f+l)/2;

t2.f = (f+l+1)/2;

t2.l = l;

t1.start();

t1.join();

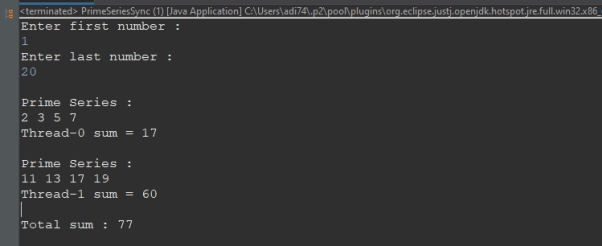
t2.start();

t2.join();

System.***out***.println("\nTotal sum : "+CalculatePrime.*sum*);

}

}



**13. Write a Java program that calculates the sum of all Pallindrome numbers up to a given limit using multiple threads.**

package multithreading;

import java.util.Scanner;

class PalindromeSeries

{

static int *sum*=0;

public void printPal(int a,int b)

{

int sum=0;

System.***out***.println("\nPalindrome Series : ");

for(int i=a;i<=b;i++)

{

if(isPal(i))

{

System.***out***.print(i+" ");

sum+=i;

}

}

System.***out***.println("\n"+Thread.*currentThread*().getName()+" sum --> "+sum);

PalindromeSeries.*sum* += sum;

}

public boolean isPal(int n)

{

int rem=0,rev=0;

int i=n;

while(i!=0)

{

rem = i%10;

rev=(rev\*10)+rem;

i/=10;

}

if(rev==n)

return true;

else

return false;

}

}

public class PalindromeSeriesSync extends Thread

{

int a; int b;

PalindromeSeriesSync(int a,int b)

{

this.a=a;

this.b=b;

}

public void run()

{

PalindromeSeries obj = new PalindromeSeries();

obj.printPal(a, b);

}

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

{

Scanner sc = new Scanner(System.***in***);

System.***out***.println("Enter first number : ");

int f = sc.nextInt();

System.***out***.println("Enter last number : ");

int l = sc.nextInt();

PalindromeSeriesSync t1 = new PalindromeSeriesSync(f, ((f+l)/2));

PalindromeSeriesSync t2 = new PalindromeSeriesSync(((f+l)/2+1) ,l );

t1.start();

t1.join();

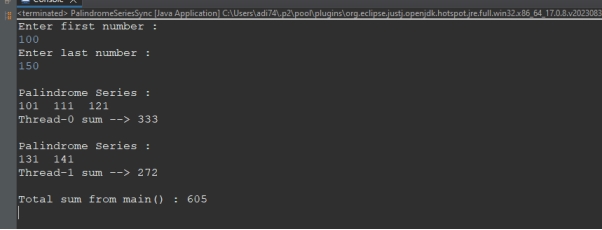
t2.start();

t2.join();

System.***out***.println("\nTotal sum from main() : "+PalindromeSeries.*sum*);

}

}



**14. Write a program that uses multiple threads to calculate the prime series. Each thread should calculate the half prime series and sum of prime numbers , and then the main thread should print the sum of all prime numbers.**

package AssignmentNo41;

import java.util.Scanner;

class CalculatePrime

{

static int *sum*=0;

public void getPrime(int f, int l)

{

int sum=0;

System.***out***.println("\nPrime Series : ");

for(int i=f;i<=l;i++)

{

if(*isPrime*(i))

{

System.***out***.print(i+" ");

sum+=i;

}

}

System.***out***.println("\n"+Thread.*currentThread*().getName()+" sum = "+sum);

this.*sum* += sum;

}

public static boolean isPrime(int n)

{

int cnt=0;

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

{

if(n%i==0)

{

cnt++;

}

}

if(cnt==2)

return true;

else

return false;

}

}

public class PrimeSeriesSync extends Thread

{

int f;int l;

public void run()

{

CalculatePrime obj = new CalculatePrime();

obj.getPrime(f, l);

}

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

{

Scanner sc = new Scanner(System.***in***);

System.***out***.println("Enter first number : ");

int f = sc.nextInt();

System.***out***.println("Enter last number : ");

int l = sc.nextInt();

PrimeSeriesSync t1 = new PrimeSeriesSync();

PrimeSeriesSync t2 = new PrimeSeriesSync();

t1.f=f;

t1.l = (f+l)/2;

t2.f = (f+l+1)/2;

t2.l = l;

t1.start();

t1.join();

t2.start();

t2.join();

System.***out***.println("\nTotal sum : "+CalculatePrime.*sum*);

}

}

