**Multithreading**

1.   Write a Java program to perform a runnable interface, take two threads t1 and t2 and fetch the names of the thread using getName() method.

**CODE:**

public class Thread implements Runnable {

public void run() {

String threadName = Thread.currentThread().getName();

System.out.println("Thread name: " + threadName);

}

public static void main(String[] args) {

Thread thread = new Thread();

Thread t1 = new Thread(thread);

Thread t2 = new Thread(thread);

t1.setName("Thread 1");

t2.setName("Thread 2");

t1.start();

t2.start();

}

}

2.Given an integer N, the task is to write program to print the first N natural numbers in increasing order using two threads.

***Input:****N = 10****Output:****1 2 3 4 5 6 7 8 9 10*

***Input:****N = 18****Output:****1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18*

**CODE:**

import java.util.\*;

class Numbers implements Runnable {

private static final Object lock = new Object();

private static int currentNumber = 1;

private static int N;

public Numbers(int N) {

this.N = N;

}

public void run() {

while (true) {

synchronized (lock) {

if (currentNumber > N) {

break;

}

System.out.print(currentNumber + " ");

currentNumber++;

lock.notify();

try {

if (currentNumber <= N) {

lock.wait();

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

public class Number{

public static void main(String[] args) {

int N;

System.out.println("Enter n value:");

Scanner sc=new Scanner(System.in);

N=sc.nextInt();

Thread thread1 = new Thread(new PrintNumbers(N));

Thread thread2 = new Thread(new PrintNumbers(N));

thread1.start();

thread2.start();

}

}

3.  Write a two-threaded program, where one thread finds all prime numbers (in 0 to 10) and another thread finds all palindrome numbers (in 10 to 50). Schedule these threads in a sequential manner to get the results.

Palindrome numbers from 10 to 50 : 11 22 33 44

Prime numbers from 0 to 10 : 2 3 5 7

**CODE:**

import java.util.Scanner;

class PrimeNumberThread extends Thread {

private final int start;

private final int end;

public PrimeNumberThread(int start, int end) {

this.start = start;

this.end = end;

}

public void run() {

System.out.println("Prime numbers from " + start + " to " + end + ":");

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

if (isPrime(i)) {

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

}

}

System.out.println();

}

private boolean isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i <= Math.sqrt(num); i++) {

if (num % i == 0) {

return false;

}

}

return true;

}

}

class PalindromeNumberThread extends Thread {

private final int start;

private final int end;

public PalindromeNumberThread(int start, int end) {

this.start = start;

this.end = end;

}

public void run() {

System.out.println("Palindrome numbers from " + start + " to " + end + ":");

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

if (isPalindrome(i)) {

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

}

}

System.out.println();

}

private boolean isPalindrome(int num) {

int originalNum = num;

int reversedNum = 0;

while (num > 0) {

int digit = num % 10;

reversedNum = reversedNum \* 10 + digit;

num /= 10;

}

return originalNum == reversedNum;

}

}

public class Thread{

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the start range for prime numbers: ");

int primeStart = scanner.nextInt();

System.out.print("Enter the end range for prime numbers: ");

int primeEnd = scanner.nextInt();

System.out.print("Enter the start range for palindrome numbers: ");

int palindromeStart = scanner.nextInt();

System.out.print("Enter the end range for palindrome numbers: ");

int palindromeEnd = scanner.nextInt();

PrimeNumberThread primeThread = new PrimeNumberThread(primeStart, primeEnd);

PalindromeNumberThread palindromeThread = new PalindromeNumberThread(palindromeStart, palindromeEnd);

primeThread.start();

try {

primeThread.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

palindromeThread.start();

scanner.close();

}

}