| **Name:** | Mahadev Balla |
| --- | --- |
| **UID:** | 2023300010 |
| **Experiment No.** | 10A |

| **AIM:** | Implement programs to demonstrate multi-threading. |
| --- | --- |
| **Program 1** | |
| **PROBLEM STATEMENT :** | Write a Java program that implements a multi-thread application that has three threads. First thread generates a random integer for every 1 second; second thread computes the square of the number and prints; third thread will print the value of cube of the number.  Output:  Random Integer generated : 82  Square of 82 = 6724  Cube of 82 = 551368 |
| **PROGRAM:** | import java.util.\*;  class Number extends Thread{  static int random\_num;  public void run(){  while(true){  Random random = new Random();  int randomNumber = random.nextInt(1000);  System.out.println("Random Integer generated: " + randomNumber);  Square t2 = new Square(randomNumber);  Cube t3 = new Cube(randomNumber);  t2.start();  t3.start();  try {  Thread.sleep(1000);  System.out.println();  } catch (Exception e) {  System.out.println(e);  }  }  }  }  class Square extends Thread{  int num;  Square(int n){  this.num = n;  }  public void run(){  System.out.println("Square of " + num + " : " + num\*num);  }  }  class Cube extends Thread{  int num;  Cube(int n){  this.num = n;  }  public void run(){  System.out.println("Cube of " + num + " : " + num\*num\*num);  }  }  class random{  public static void main(String[] args) {  Number t1 = new Number();  t1.start();  }  } |
| **RESULT:** | |
| **Program 2** | |
| **PROBLEM STATEMENT :** | Write a two-threaded program, where one thread finds all prime numbers (in 0 to 100) and another thread finds all palindrome numbers (in 10 to 1000). Schedule these threads in a sequential manner to get the results. Now reschedule them as parallel threads. |
| **PROGRAM:** | import java.util.\*;  class PrimeThread extends Thread {  public void run() {  for (int i = 0; i <= 100; i++) {  if (isPrime(i)) {  System.out.println("Prime Number : " + i + " ");  }  }  System.out.println();  }  private boolean isPrime(int num) {  if (num <= 1) return false;  if (num <= 3) return true;  if (num % 2 == 0 || num % 3 == 0) return false;  for (int i = 5; i \* i <= num; i += 6) {  if (num % i == 0 || num % (i + 2) == 0) {  return false;  }  }  return true;  }  }  class PalindromeThread extends Thread {  public void run() {  for (int i = 10; i <= 600; i++) {  if (isPalindrome(i)) {  System.out.println("Palindrome Number : " + i + " ");  }  }  System.out.println();  }  private boolean isPalindrome(int num) {  int reversedNum = 0;  int originalNum = num;  while (num != 0) {  int digit = num % 10;  reversedNum = reversedNum \* 10 + digit;  num /= 10;  }  return originalNum == reversedNum;  }  }  public class number {  public static void main(String[] args) {    System.out.println("Sequential Execution : ");  PrimeThread primeThread = new PrimeThread();  PalindromeThread palindromeThread = new PalindromeThread();  primeThread.start();  try {  primeThread.join();  } catch (InterruptedException e) {  System.out.println(e);  }  palindromeThread.start();      System.out.println("\nParallel Execution:");  PrimeThread parallelPrimeThread = new PrimeThread();  PalindromeThread parallelPalindromeThread = new PalindromeThread();  parallelPrimeThread.start();  parallelPalindromeThread.start();  }  } |
| **RESULT:** | |
| **CONCLUSION :** Implemented multi-threading in the given problems. | |