Java Programming [CSE201] Enrolment No.:23DCS002

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

**DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH**

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3rd

Subject Code: CSE201

Academic year: 2024-25

Part - 7

|  |  |
| --- | --- |
| **No.** | **Multithreading** |
| 32. | Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.  **PROGRAM CODE:**  class MyThread extends Thread {      public void run() {          System.out.println("Hello World");      }  }  public class prec32\_a {      public static void main(String[] args) {          MyThread t1 = new MyThread();          t1.start();      }  }  class MyRunnable implements Runnable {      public void run() {          System.out.println("Hello World");      }  }  public class prec32\_b {      public static void main(String[] args) {          MyRunnable m1 =  new MyRunnable();          Thread t1 = new Thread(m1);          t1.start();      }  }  **OUTPUT:**      **CONCLUSION:** Runnable Interface and Thread Class. |
| 33 | Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.  **PROGRAM CODE:**  import java.util.Scanner;  class MyThread extends Thread {      int start, end;      static int sum = 0;      MyThread(int start, int end) {          this.start = start;          this.end = end;      }      static void addSum(int partialSum) {          sum += partialSum;      }      public void run() {          int partialSum = 0;          for (int i = start; i <= end; i++) {              partialSum += i;          }          addSum(partialSum);      }  }  public class prec33 {      public static void main(String[] args) {          int N, numThreads;          Scanner s = new Scanner(System.in);          System.out.print("Enter the number 'N': ");          N = s.nextInt();          System.out.print("Enter the number of threads to be used (should be less than or equal to N): ");          numThreads = s.nextInt();          MyThread[] threads = new MyThread[numThreads];          int range = N / numThreads;          int start = 1, end;          for (int i = 0; i < numThreads; i++) {              end = (i == numThreads - 1) ? N : start + range - 1;              threads[i] = new MyThread(start, end);              threads[i].start();              start = end + 1;          }          try {              for (int i = 0; i < numThreads; i++) {                  threads[i].join();              }          } catch (InterruptedException e) {              System.out.println("Thread interrupted.");          }          System.out.println("The sum of numbers from 1 to " + N + " is: " + MyThread.sum);      }  }  **OUTPUT:**    **CONCLUSION:** Multithreading code to do addition of number using multiple thread. |
| 34 | Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.  **PROGRAM CODE:**  class even extends Thread {      int n;      public even(int n) {          this.n = n;      }      public void run() {          System.out.println(n \* n);      }  }  class odd extends Thread {      int n;      public odd(int n) {          this.n = n;      }      public void run() {          System.out.println(n \* n \* n);      }  }  public class prec34 {      public static void main(String[] args) {          even e;          odd o;          for (int i = 1; i <= 10; i++) {              e = new even(i);              o = new odd(i);              try {                  if (i % 2 == 0) {                      e.start();                  } else {                      o.start();                  }                  Thread.sleep(1000);              } catch (Exception ex) {                  ex.printStackTrace();              }          }      }  }  **OUTPUT:**    **CONCLUSION:** Program print square if number is even and cube if number is odd. |
| 35 | Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.  **PROGRAM CODE:**  class MyThread extends Thread {      public void run() {          for (int i = 0; i < 3; i++) {              System.out.println("Call No : " + i);              try {                  sleep(1000);              } catch (InterruptedException e) {                  e.printStackTrace();              }          }      }  }  public class test3 {      public static void main(String[] args) {          MyThread t1 = new MyThread();          t1.start();          for (int i = 0; i < 3; i++) {              System.out.println("Main Thread");          }      }  }  **OUTPUT:**    **CONCLUSION:** sleep() method of class thread allow user to interrupt the execution for given interval. |
| 36 | Write a program to create three threads ‘FIRST’, ‘SECOND’, ‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’ thread to 5(default) and the ‘THIRD’ thread to 7.  **PROGRAM CODE:**  class first extends Thread {      public void run() {          System.out.println("First");      }  }  class Second extends Thread {      public void run() {          System.out.println("Second");      }  }  class Third extends Thread {      public void run() {          System.out.println("Third");      }  }  public class prec36 {      public static void main(String[] args) {          first f = new first();            Second s = new Second();          Third t = new Third();          f.setPriority(3);          s.setPriority(5);          t.setPriority(7);          f.start();          s.start();          t.start();      }  }  **OUTPUT:**    **CONCLUSION:** priority function allow user to prioritize thread. |
| 37 | Write a program to solve producer-consumer problem using thread synchronization.  **PROGRAM CODE:**  class Produce extends Thread {      int n;      boolean produced = false;      Produce(int n) {          this.n = n;      }      public synchronized void run() {          for (int i = 1; i <= n; i++) {              while (produced) {                  try {                      wait();                  } catch (InterruptedException e) {                      Thread.currentThread().interrupt();                  }              }              System.out.println("Produced: " + i);              produced = true;              notify();          }      }  }  class Consume extends Thread {      int n;      Produce producer;      Consume(int n, Produce producer) {          this.n = n;          this.producer = producer;      }      public synchronized void run() {          for (int i = 1; i <= n; i++) {              synchronized (producer) {                  while (!producer.produced) {                      try {                          producer.wait();                      } catch (InterruptedException e) {                          Thread.currentThread().interrupt();                      }                  }                  System.out.println("Consumed: " + i);                  producer.produced = false;                  producer.notify();              }          }      }  }  public class prec37 {      public static void main(String[] args) {          int n = 10;          Produce p = new Produce(n);          Consume c = new Consume(n, p);          p.start();          c.start();      }  }  **OUTPUT:**    **CONCLUSION:** Synchronized function allows to execute thread in a synchronized way. |