**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

**Chandubhai S Patel Institute of Technology**

**U & P U. Patel Department of Computer Engineering**

**Subject Name:** JAVA **Subject Code:** CE251

**Semester:** 3rd **Academic Year:** 2022-23

**Assignment 6**

**Github Repository Link :** **https://github.com/ronak-bhalgami/JAVA\_Practice\_Set\_6**

|  |  |
| --- | --- |
| **1.** | **Write a program to create thread which display “Hello World” message.**  **A. by extending Thread class**  **B. by using Runnable interface.** |
| **Code :** | **A. by extending Thread class**  //This program is done by Ronak Bhalgami(21CE009)  import java.lang.Thread;  class MyThreadA extends Thread  {      public void run()      {          for (int i = 0; i < 2; i++) {              System.out.println("Hello World");          }          System.out.println("End of MyThreadA");      }  }  public class Thread\_HelloWorld {      public static void main(String[] args) {          MyThreadA T1 = new MyThreadA();          T1.start();      }    }  **B. by using Runnable interface.**  class MyThreadRunable1 implements Runnable  {      public void run()      {          for (int i = 0; i < 2; i++) {              System.out.println("Hello World");          }          System.out.println("End of MyThreadRunable1");      }  }  public class Runalbe\_HelloWorld {      public static void main(String[] args) {          MyThreadRunable1 M1 = new MyThreadRunable1();          Thread T1 = new Thread(M1);          T1.start();      }  } |
| **Output :** | **A. by extending Thread class**    **B. by using Runnable interface.** |
| **2.** | **Generate 15 random numbers from 1 to 100 and store it in an int array. Write a program to display the numbers stored at odd indexes by thread1 and display numbers stored at even indexes by thread2.** |
| **Code:** | //This program is done by Ronak Bhalgami(21CE009)  import java.util.\*;  import javax.lang.model.util.ElementScanner14;  class ThreadSample implements Runnable{      int random[]=new int[15];      ThreadSample(int a[])      {          random=a;      }      void display()      {          if(Thread.currentThread().getName().equals("Thread-0"))          {              for(int i=0;i<random.length;i++)              {                  if(random[i]%2==0)                  {                      System.out.println("Even Number : "+random[i]+" Identify by "+Thread.currentThread().getName());                  }              }          }          else{              for(int i=0;i<random.length;i++)              {                  if(random[i]%2!=0)                  {                      System.out.println("Odd Number : "+random[i]+" Identify by "+Thread.currentThread().getName());                  }              }          }      }      public void run()      {          display();      }  }  public class PR2  {      public static void main(String[] args) {          Random r=new Random();          int random[]=new int[15];          for(int i=0;i<random.length;i++)          {              random[i]=r.nextInt(100);          }          ThreadSample s1=new ThreadSample(random);          Thread t1=new Thread(s1);          t1.start();          Thread t2=new Thread(s1);          t2.start();      }  } |
| **Output :** |  |
| **3.** | **Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.** |
| **Code :** | //This program is done by Ronak Bhalgami(21CE009)  class Mythread extends Thread {      public static int counter = 0;      public void run() {          System.out.println(Thread.currentThread().getName() + " is running");      }      static void increment() {          counter++;      }  }  class pra\_6\_3 {      public static void main(String[] args) {          Mythread t1 = new Mythread();          t1.start();          System.out.println("Before increment is called the value of counter is : " + t1.counter);          System.out.println("\nThread t1 sleep method called");          try {              t1.sleep(1000);          } catch (InterruptedException e) {              System.out.println(e);          }          t1.increment();          System.out.println("After increment is called the value of counter is : " + t1.counter);      }  } |
| **Output :** |  |
| **4.** | **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.** |
| **Code :** | //This program is done by Ronak Bhalgami(21CE009)  class Mythread extends Thread {      public void run() {          System.out.println("Thread " + Thread.currentThread().getName() + " is running");      }  }  public class pra\_6\_4 {      public static void main(String[] args) {          Mythread t1 = new Mythread();          Mythread t2 = new Mythread();          Mythread t3 = new Mythread();          t1.setName("First");          t2.setName("Second");          t3.setName("Third");          t1.setPriority(3);          t2.setPriority(5);          t3.setPriority(7);          t1.start();          t2.start();          t3.start();      }  } |
| **Output :** |  |
| **5.** | **Write a program to solve producer-consumer problem using thread Synchronization.** |
| **Code :** | //This program is done by Ronak Bhalgami(21CE009)  import java.util.LinkedList;  public class pra\_6\_5 {      public static void main(String[] args) throws InterruptedException {          // Object of a class that has both produce()          // and consume() methods          final PC pc = new PC();          // Create producer thread          Thread t1 = new Thread(new Runnable() {              @Override              public void run() {                  try {                      pc.produce();                  } catch (InterruptedException e) {                      e.printStackTrace();                  }              }          });          // Create consumer thread          Thread t2 = new Thread(new Runnable() {              @Override              public void run() {                  try {                      pc.consume();                  } catch (InterruptedException e) {                      e.printStackTrace();                  }              }          });          // Start both threads          t1.start();          t2.start();          // t1 finishes before t2          t1.join();          t2.join();      }      // This class has a list, producer (adds items to list      // and consumber (removes items).      public static class PC {          // Create a list shared by producer and consumer          // Size of list is 2.          LinkedList<Integer> list = new LinkedList<>();          int capacity = 2;          // Function called by producer thread          public void produce() throws InterruptedException {              int value = 0;              while (true) {                  synchronized (this) {                      // producer thread waits while list                      // is full                      while (list.size() == capacity)                          wait();                      System.out.println("Producer produced-" + value);                      // to insert the jobs in the list                      list.add(value++);                      // notifies the consumer thread that                      // now it can start consuming                      notify();                      // makes the working of program easier                      // to understand                      Thread.sleep(1000);                  }              }          }          // Function called by consumer thread          public void consume() throws InterruptedException {              while (true) {                  synchronized (this) {                      // consumer thread waits while list                      // is empty                      while (list.size() == 0)                          wait();                      // to retrive the ifrst job in the list                      int val = list.removeFirst();                      System.out.println("Consumer consumed-" + val);                      // Wake up producer thread                      notify();                      // and sleep                      Thread.sleep(1000);                  }              }          }      }  } |
| **Output :** |  |