|  |
| --- |
| Exp: 14 Date: 08/11/2023  **JAVA PROGRAMMING**  **CS6308** |

MULTI-THREADING

Name: VIJAI SURIA M

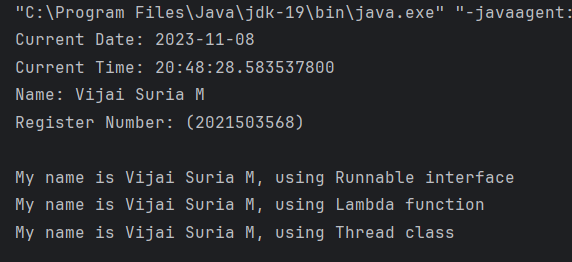
Reg No.: 2021503568

1. **Write a Java Program to display your name using thread with runnable interface**

**CODE**

|  |
| --- |
| import java.time.LocalDate; import java.time.LocalTime; class Name implements Runnable{  public void run(){  System.*out*.println("My name is Vijai Suria M, using Runnable interface");  } } class Names extends Thread {  public void run(){  System.*out*.println("My name is Vijai Suria M, using Thread class");  } } public class Name3568 {  public static void main(String[] args){  System.*out*.println("Current Date: " + LocalDate.*now*());  System.*out*.println("Current Time: " + LocalTime.*now*());  System.*out*.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");  Name obj = new Name();  Thread t1 = new Thread(obj);  Thread t2 = new Thread(() -> {System.*out*.println("My name is Vijai Suria M, using Lambda function");});  Thread t3 = new Names();  t1.start();  t2.start();  t3.start();  } } |

**OUTPUT:**

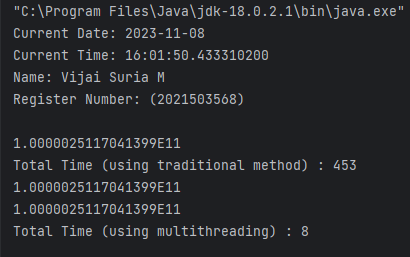


1. **Write a Java Program to show the difference between traditional and concurrent programming.**

**CODE**

|  |
| --- |
| import java.time.LocalDate; import java.time.LocalTime; import java.lang.Thread; class Myclass implements Runnable {  public void run()  {  double result=0;  for (int i = 0; i < 100000099; i++)  {  result = (i / 1.234566) \* 1234.567890988;  }  System.*out*.println(result);  } }  public class Concurrent3568 {  public static void main(String [] arg) throws Exception  {  System.*out*.println("Current Date: " + LocalDate.*now*());  System.*out*.println("Current Time: " + LocalTime.*now*());  System.*out*.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");  final long startTime=System.*currentTimeMillis*();  double result=0;  for(long i=0; i< 100000099; i++)  {  result=(i/1.234566)\*1234.567890988;  }  System.*out*.println(result);  System.*out*.println("Total Time (using traditional method) : " + (endTime-startTime));  final long sTime = System.*currentTimeMillis*();  Thread t1 = new Thread(new Myclass());  Thread t2 = new Thread(new Myclass());  t1.start();  t2.start();  if(t1.isAlive())  t1.join();  if(t2.isAlive())  t2.join();  final long eTime = System.*currentTimeMillis*();  System.*out*.println("Total Time (using multithreading) : " + (eTime-sTime));  } } |

**OUTPUT:**

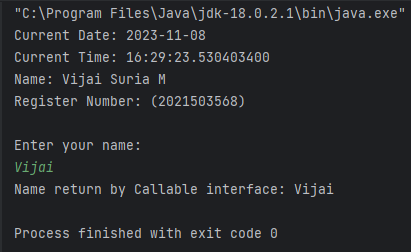


1. **Write a Java Program using callable interface to show that return value of a thread can be caught and displayed.**

**CODE**

|  |
| --- |
| import java.util.Scanner; import java.util.concurrent.\*; import java.time.\*;  class CallableInterface implements Callable<String> {  public String call(){  Scanner in = new Scanner(System.*in*);  System.*out*.println("Enter your name: ");  return in.nextLine();  } } public class Callable3568 {  public static void main(String[] args) throws Exception{  System.*out*.println("Current Date: " + LocalDate.*now*());  System.*out*.println("Current Time: " + LocalTime.*now*());  System.*out*.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");  ExecutorService task = Executors.*newSingleThreadScheduledExecutor*();  Future<String> name = task.submit(new CallableInterface());  System.*out*.println("Name returned by Callable interface: " + name.get());  task.shutdown();  } } |

**OUTPUT:**

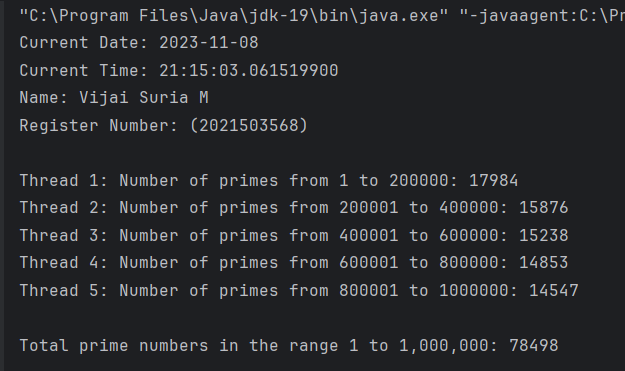


1. **Write a Java a program to implement the thread pool concept to find the prime numbers in 10 million numbers.  
   Create a 5 threads to find prime numbers, separate the number ranges as five entity.**

**CODE**

|  |
| --- |
| import java.util.concurrent.ExecutorService; import java.util.concurrent.Executors; import java.util.concurrent.Future; import java.util.concurrent.Callable; import java.time.\*;  public class PrimeFind3568 {  public static void main(String[] args) throws Exception {  System.*out*.println("Current Date: " + LocalDate.*now*());  System.*out*.println("Current Time: " + LocalTime.*now*());  System.*out*.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");  int totalNumbers = 1000000;  int threadCount = 5;  int rangeSize = totalNumbers / threadCount;  ExecutorService executorService = Executors.*newFixedThreadPool*(threadCount);  Future<Integer>[] results = new Future[threadCount];  for (int i = 0; i < threadCount; i++) {  final int startRange = i \* rangeSize + 1;  final int endRange = (i + 1) \* rangeSize;  results[i] = executorService.submit(new PrimeCounter(startRange, endRange));  System.*out*.printf("Thread %d: Number of primes from %d to %d: %d", i+1, startRange, endRange, results[i].get());  System.*out*.println();  }  int totalPrimes = 0;  for (int i = 0; i < threadCount; i++) {  totalPrimes += results[i].get();  }  executorService.shutdown();  System.*out*.println("\nTotal prime numbers in the range 1 to 1,000,000: " + totalPrimes);  }  static class PrimeCounter implements Callable<Integer> {  private final int start;  private final int end;  PrimeCounter(int start, int end) {  this.start = start;  this.end = end;  }  @Override  public Integer call() {  int count = 0;  for (int num = start; num <= end; num++) {  if (isPrime(num)) {  count++;  }  }  return count;  }  // Helper function to check if a number is prime  private boolean isPrime(int number) {  if (number <= 1) {  return false;  }  for (int i = 2; i <= Math.*sqrt*(number); i++) {  if (number % i == 0) {  return false;  }  }  return true;  }  } } |

**OUTPUT:**

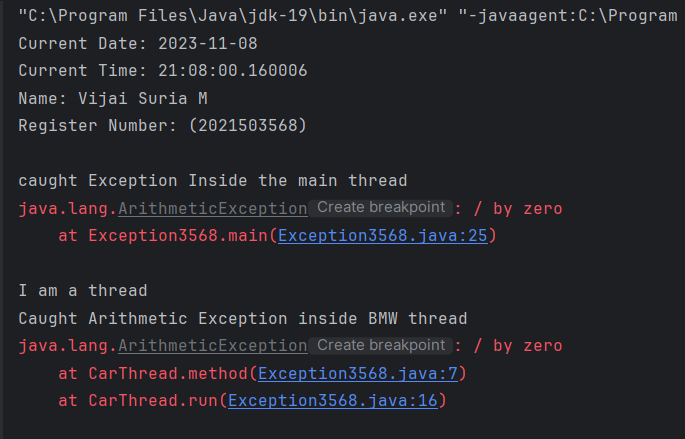


1. **Write a Java program to show the exception in main thread and user defined thread.**

**CODE**

|  |
| --- |
| import java.time.LocalDate; import java.time.LocalTime;  class CarThread extends Thread{  private void method(){  try {  System.*out*.println(5/0);  }  catch(ArithmeticException e){  System.*out*.println("Caught Arithmetic Exception inside " + Thread.*currentThread*().getName() + " thread");  e.printStackTrace();  }  }  public void run(){  System.*out*.println("\nI am a thread");  method();  } } public class Exception3568 {  public static void main(String[] args) throws InterruptedException{  System.*out*.println("Current Date: " + LocalDate.*now*());  System.*out*.println("Current Time: " + LocalTime.*now*());  System.*out*.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");  try {  System.*out*.println(5/0);  }  catch (ArithmeticException e){  System.*out*.println("caught Exception Inside the " + Thread.*currentThread*().getName() + " thread");  e.printStackTrace();  }  Thread.*sleep*(1000);  CarThread BMW = new CarThread();  BMW.setName("BMW");  BMW.start();  } } |

**OUTPUT:**

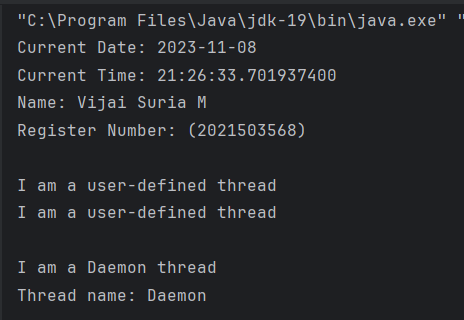


1. **Write a program to create a thread as Daemon and display the non-Daemon thread accordingly**

**CODE**

|  |
| --- |
| import java.time.LocalDate; import java.time.LocalTime;  public class Daemon3568 extends Thread{  public void run(){  if(Thread.*currentThread*().isDaemon()){//checking for daemon thread  System.*out*.println("\nI am a Daemon thread\nThread name: " + Thread.*currentThread*().getName());  }  else{  System.*out*.println("I am a user-defined thread");  }  }  public static void main(String[] args){  System.*out*.println("Current Date: " + LocalDate.*now*());  System.*out*.println("Current Time: " + LocalTime.*now*());  System.*out*.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");  Daemon3568 t1=new Daemon3568();//creating thread  Daemon3568 t2=new Daemon3568();  Daemon3568 t3=new Daemon3568();   t1.setDaemon(true);//now t1 is daemon thread  t1.setName("Daemon");   t1.start();//starting threads  t2.start();  t3.start();  } } |

**OUTPUT:**



1. **Write a program to access the file by more than one thread and display the content using synchronization.  
   A single file is being accessed by both threads t1 and t2.**

**CODE**

|  |
| --- |
| import java.nio.file.\*; import java.io.IOException; import java.time.LocalDate; import java.time.LocalTime;  public class FileThread3568 {  private static Path *filePath* = Paths.*get*("C:\\Users\\vijai\\Documents\\ASSIGNMENTS\\JAVA-PROGRAMMING/Threads/src/demo.txt");  private static final Object *lock* = new Object();   public static void main(String[] args) {  System.*out*.println("Current Date: " + LocalDate.*now*());  System.*out*.println("Current Time: " + LocalTime.*now*());  System.*out*.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");  Thread t1 = new Thread(() -> {  synchronized (*lock*) {  try {  byte[] fileContent = Files.*readAllBytes*(*filePath*);  String content = new String(fileContent);  System.*out*.println("Thread t1: File Content -\n" + content);  } catch (IOException e) {  e.printStackTrace();  }  }  });   Thread t2 = new Thread(() -> {  synchronized (*lock*) {  try {  byte[] fileContent = Files.*readAllBytes*(*filePath*);  String content = new String(fileContent);  System.*out*.println("Thread t2: File Content -\n" + content);  } catch (IOException e) {  e.printStackTrace();  }  }  });   t1.start();  t2.start();  } } |

**OUTPUT:**

