|  |  |
| --- | --- |
| Assignment 3  Module 2: Object Oriented Programming in JAVA | Abstract  In this assignment, I have completed 13 questions on java basic topics like Garbage Collector, File Handling, Serialization and De-Serialization, Collection Framework, Generic assignment, Multi-Threading and Exception Handling. Every question starts from new page along with its code and output. Also, I have connected the screenshot of text file wherever required.  sarang deodhar  220350320026 |

**1. Override finalize method to understand the behavior of JVM garbage collector.**

**Code:**

package assignment3;

public class Que1 {

static int *i* = 0;

static int *j* = 0;

*@Override*

protected void finalize() {

*i*++;

System.***out***.println(*i*+" GC is invoked");

}

public static void main(String[] args) {

int n = 20;

Que1 q[] = new Que1[n];

for(int i = 0 ; i < n ; i++)

q[i] = new Que1();

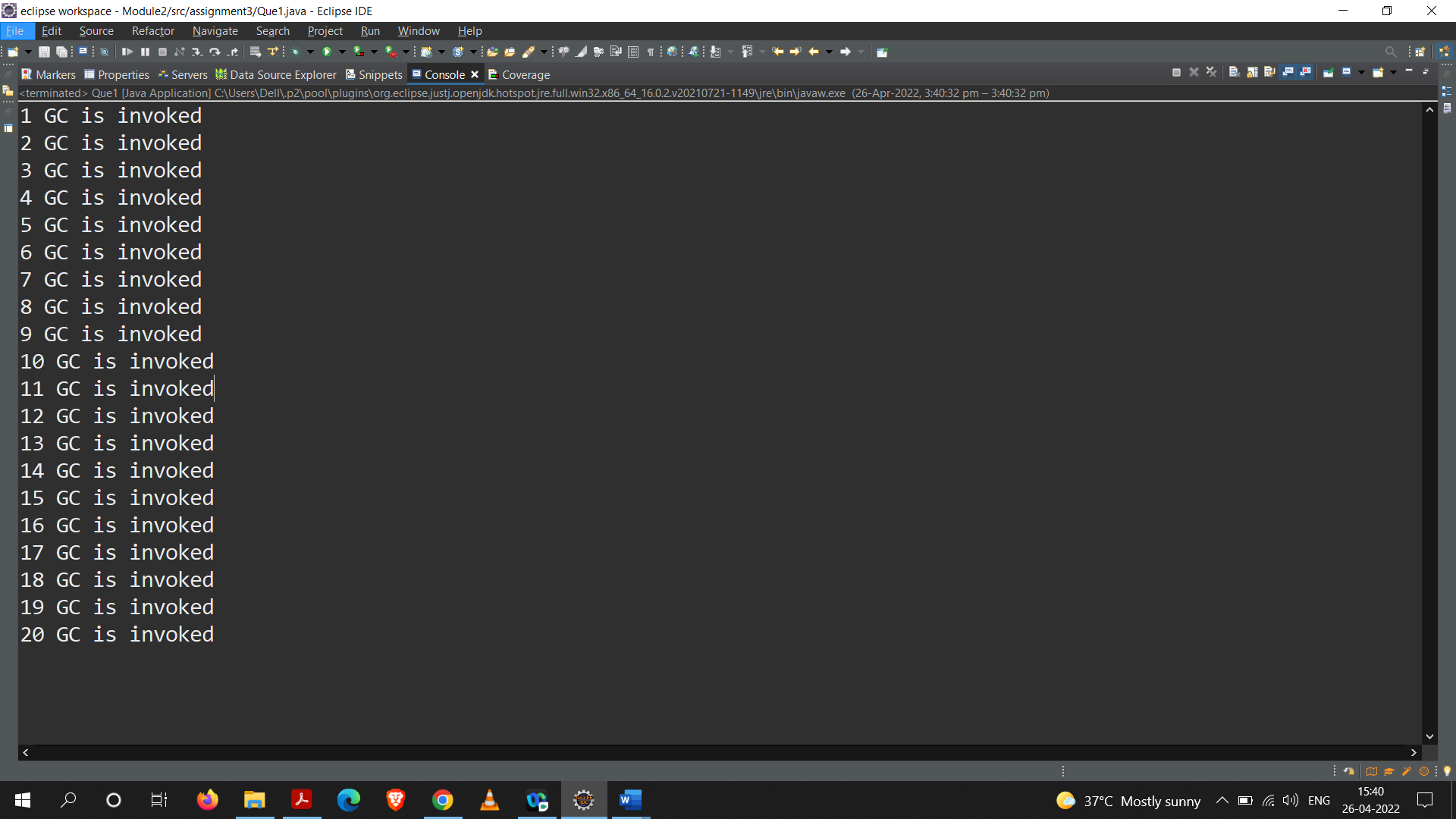
q = null;

System.*gc*();

}

}

**Output:**



**2. Create a Demo class to Read & write image/text files.**

**Code:**

package assignment3;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

public class Que2 {

public static void main(String[] args) throws Exception {

BufferedReader fr = new BufferedReader(new FileReader("E:\\eclipse workspace\\Module2\\assignment3\_Q2\\Read.txt"));

FileWriter fw = new FileWriter("E:\\eclipse workspace\\Module2\\assignment3\_Q2\\Write.txt",false);

String ch;

while((ch=fr.readLine())!=null) {

fw.write(ch+"\n");

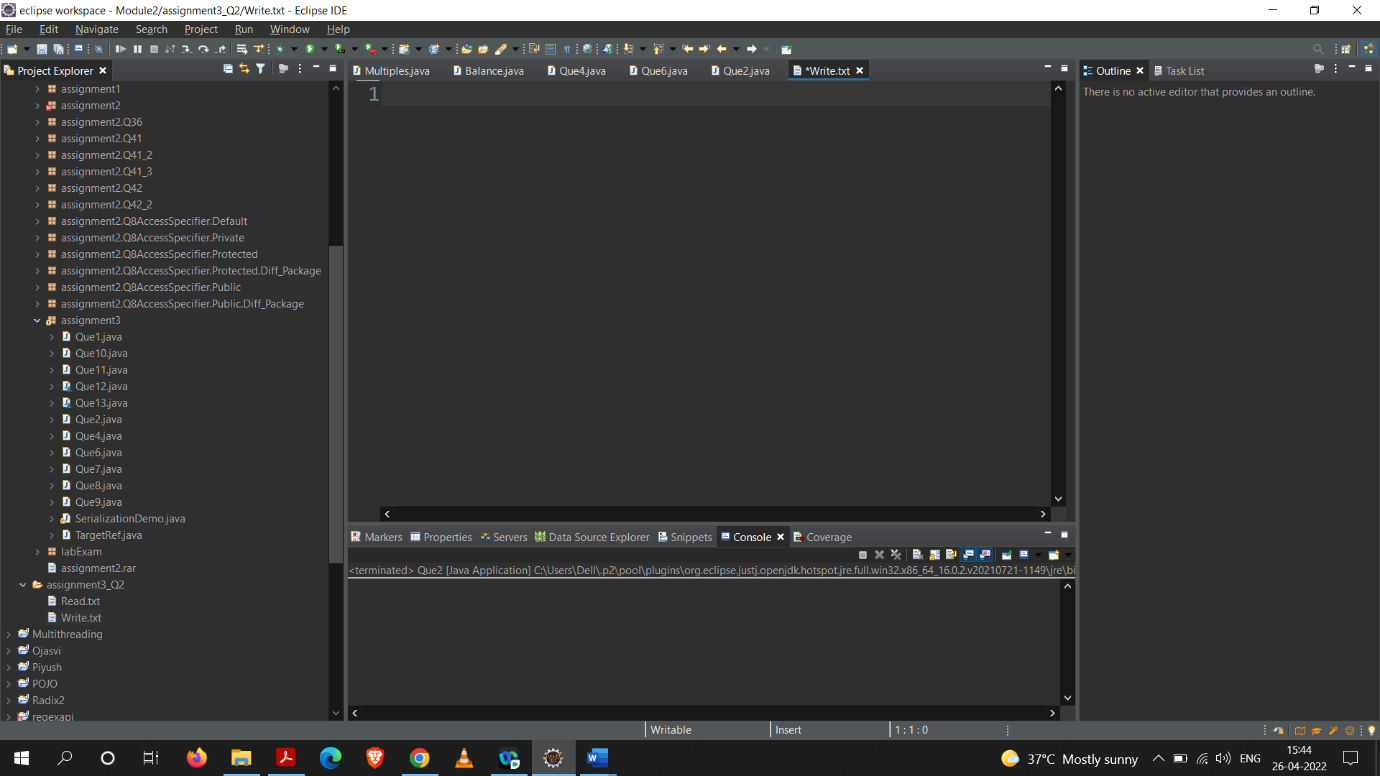
}

fr.close();

fw.close();

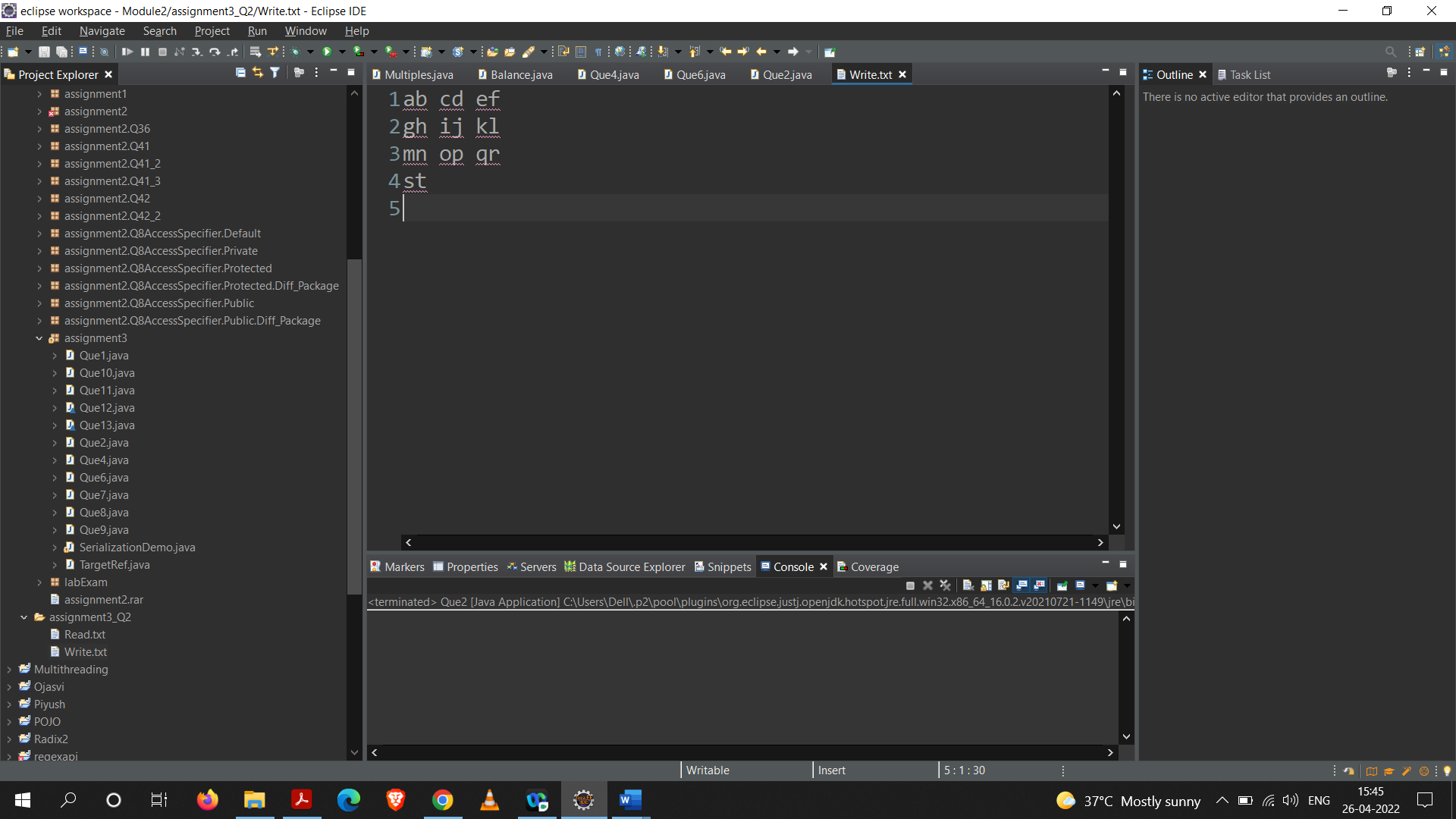
}

}

**Output: (state of write.txt before and after executing program)**

**Before**

**After**



**3. Create SerializationDemo class to illustrate serialization and de-serialization process.**

**Code:**

package assignment3;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

*@SuppressWarnings*("serial")

class Person implements Serializable{

String name;

int age;

transient String password;

public Person(String n, int a, String p){

name=n;

age=a;

password=p;

}

*@Override*

public String toString(){

return name+" "+age+" "+password;

}

}

public class SerializationDemo{

*@SuppressWarnings*("resource")

public static void main(String[] args) throws Exception {

Person p1=new Person("Java",27,"java");

// serialization

FileOutputStream fos=new FileOutputStream("person.ser");

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(p1);

// de-serialize

FileInputStream fis=new FileInputStream("person.ser");

ObjectInputStream ois=new ObjectInputStream(fis);

Person p2=(Person)ois.readObject();

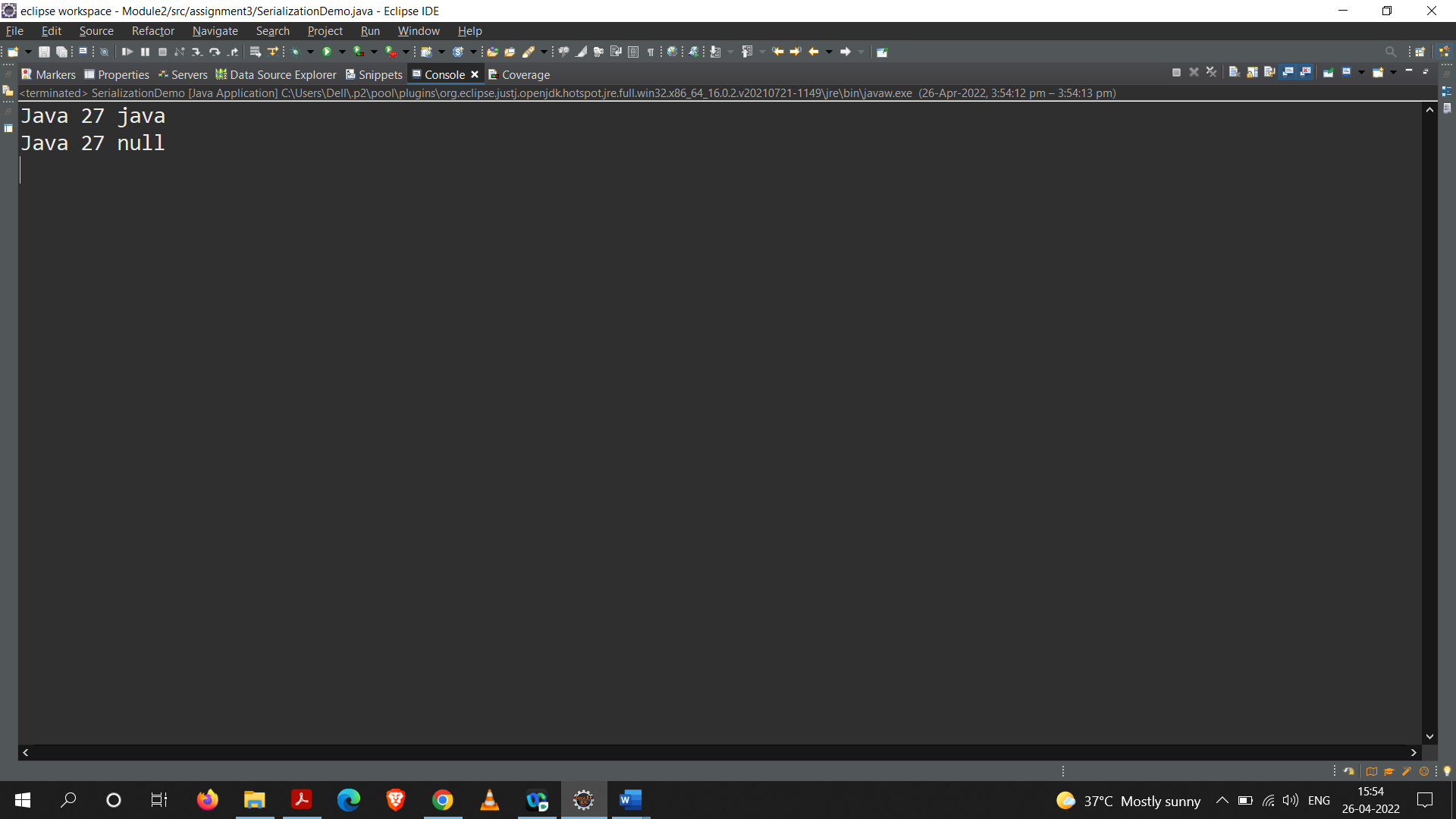
System.***out***.println(p1);

System.***out***.println(p2);

}

}

**Output:**



**4. Create an Employee HashSet collection and override equals & hashCode methods to understand how the set maintains uniqueness using these methods.**

**Code:**

package assignment3;

import java.util.\*;

class Employee {

private String name;

private int id;

public Employee(int id, String name){

this.id = id;

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

*@Override*

public String toString() {

return "Employee [name=" + name + ", id=" + id + "]";

}

*@Override*

public int hashCode() {

return Objects.*hash*(id, name);

}

*@Override*

public boolean equals(Object obj) {

if (this == obj)

return true;

else if (obj == null)

return false;

else if (getClass() != obj.getClass())

return false;

Employee other = (Employee) obj;

return id == other.id && Objects.*equals*(name, other.name);

}

}

public class Que4

{

public static void main(String[] args)

{

HashSet<Employee> hs = new HashSet<Employee>();

Employee e1 = new Employee(1,"Sarang");

Employee e2 = new Employee(2,"Deepak");

hs.add(e1);

hs.add(e2);

for(Employee e : hs)

System.***out***.println(e);

Employee e3=e2;

System.***out***.println(e3);

System.***out***.println("HashCode e1 = "+e1.hashCode());

System.***out***.println("Hashcode e2 = "+e2.hashCode());

System.***out***.println("Hashcode e3 = "+e3.hashCode());

System.***out***.println("e1 and e3 : "+e1.equals(e3));

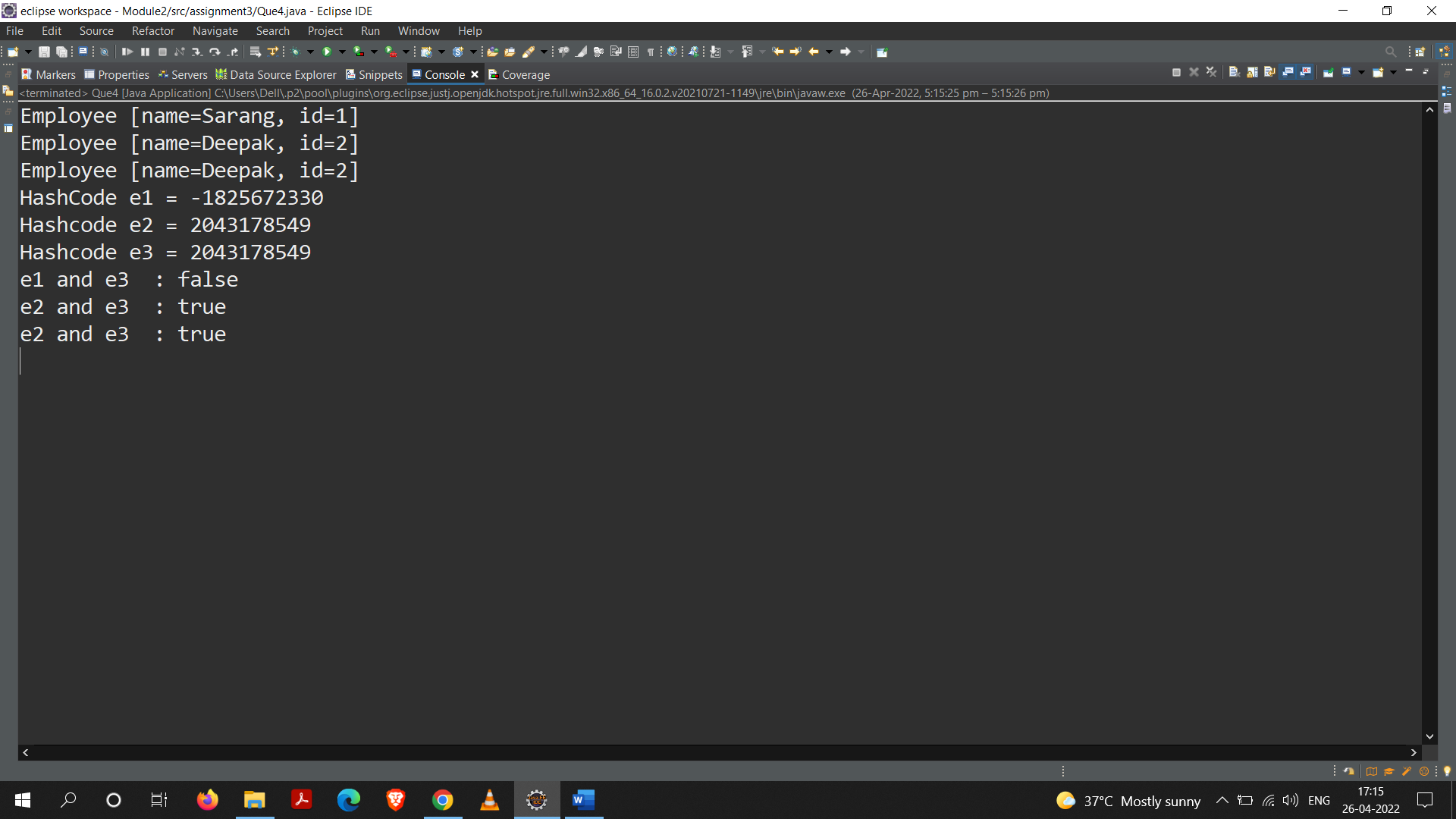
System.***out***.println("e2 and e3 : "+e2.equals(e3));

System.***out***.println("e2 and e3 : "+(e2==e3));

}

}

**Output:**



**5. Create a Sample class to understand generic assignments using “? extends SomeClass”, “? super someclass ” and “?”.**

**Code:**

package assignment3;

import java.util.ArrayList;

class Sample

{

public void print(ArrayList<? extends Number> list)

{

for(Object element:list)

{

System.***out***.println(element);

}

}

}

class Sample2

{

public void print(ArrayList<? super Double> d)

{

for(Object ele : d)

System.***out***.println(ele);

}

}

class Sample3

{

public void print(ArrayList<?>al)

{

for(Object ob : al)

System.***out***.println(ob);

}

}

public class Que5

{

public static void main(String[] args)

{

ArrayList<Integer> intList = new ArrayList<Integer>();

intList.add(100);

intList.add(200);

intList.add(300);

intList.add(400);

ArrayList<Double> dlist = new ArrayList<Double>();

dlist.add(1.1);

dlist.add(2.1);

dlist.add(3.1);

dlist.add(4.1);

ArrayList<String> str = new ArrayList<String>();

str.add("ABC");

str.add("DEF");

str.add("GHI");

str.add("JKL");

System.***out***.println(intList);

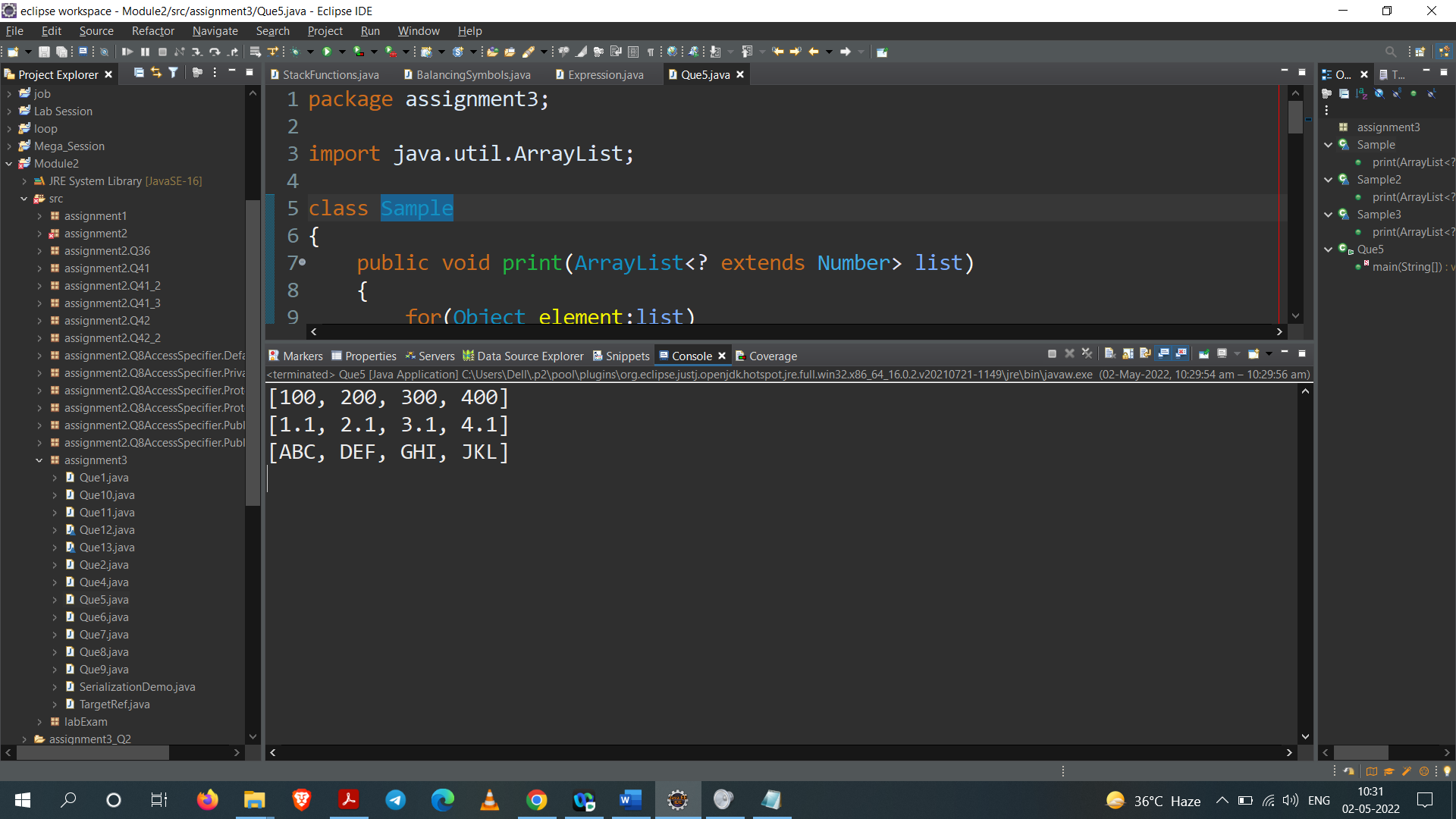
System.***out***.println(dlist);

System.***out***.println(str);

}

}

**Output:**



**6. Invoke private methods of some other class using reflection.**

**Code:**

**Que6**

package assignment3;

import java.lang.reflect.Method;

public class Que6

{

public static void main(String[] args) {

try {

TargetRef t = new TargetRef();

Method[] m = TargetRef.class.getDeclaredMethods();

for(int i = 0 ; i < m.length ; i++) {

m[i].setAccessible(true);

m[i].invoke(t);

}

}

catch(Exception e) {

e.printStackTrace();

}

}

}

**TargetRef**

package assignment3;

import java.lang.reflect.Method;

public class Que6

{

public static void main(String[] args) {

try {

TargetRef t = new TargetRef();

Method[] m = TargetRef.class.getDeclaredMethods();

for(int i = 0 ; i < m.length ; i++) {

m[i].setAccessible(true);

m[i].invoke(t);

}

}

catch(Exception e) {

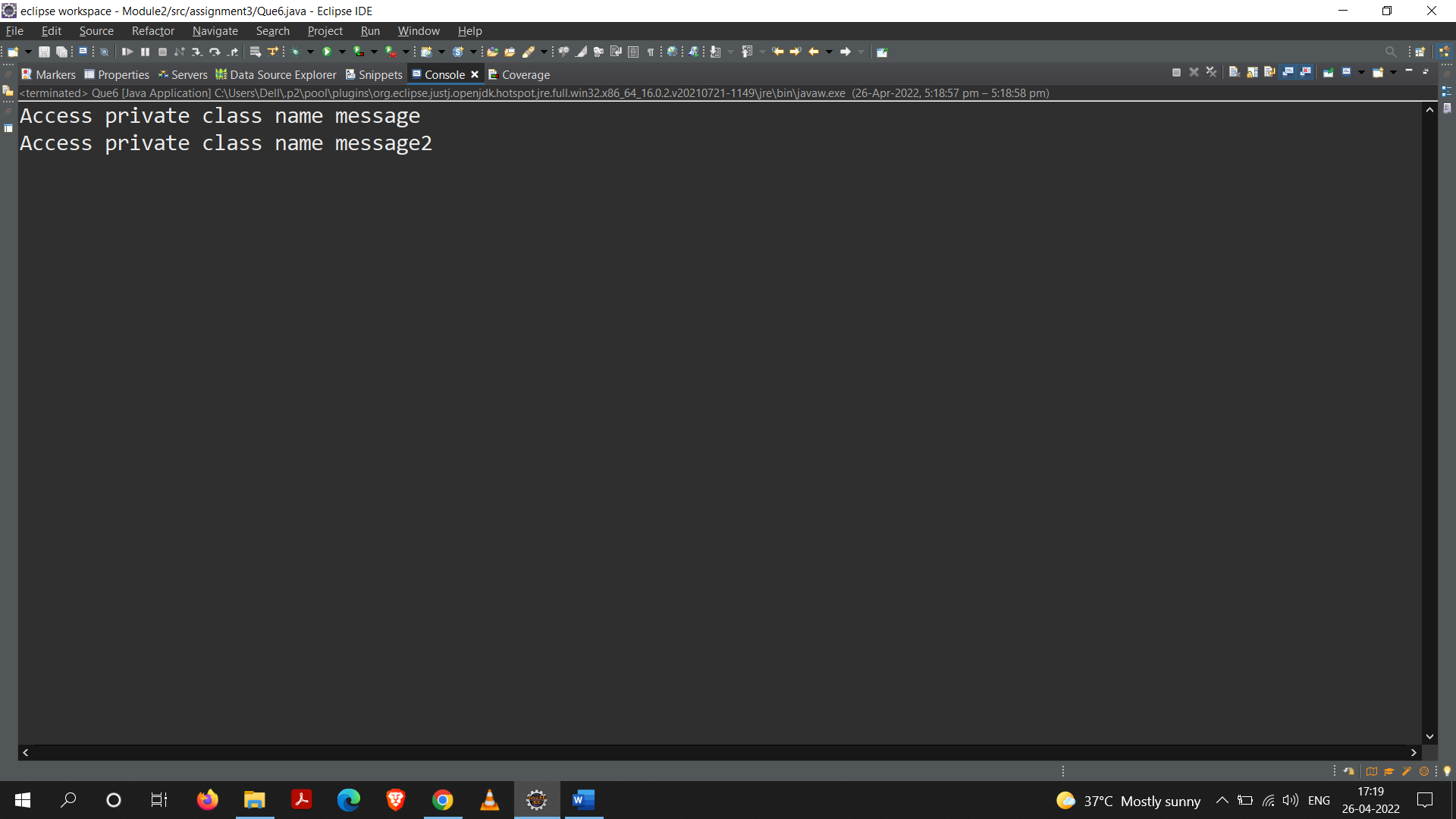
e.printStackTrace();

}

}

}

**Output:**



**7. Create multiple threads using Thread class and Runnable interfaces.**

**Code:**

package assignment3;

class MyThread1 extends Thread

{

public void run()

{

for(int i = 0 ; i < 10 ; i++)

System.***out***.println(Thread.*currentThread*());

}

}

class MyThread2 implements Runnable

{

public void run()

{

for(int i = 0 ; i < 10 ; i++)

System.***out***.println(Thread.*currentThread*());

}

}

public class Que7

{

public static void main(String[] args)

{

MyThread1 t1 = new MyThread1();

MyThread1 t2 = new MyThread1();

MyThread1 t3 = new MyThread1();

t1.start();

t2.start();

t3.start();

MyThread2 m1 = new MyThread2();

MyThread2 m2 = new MyThread2();

MyThread2 m3 = new MyThread2();

Thread b1 = new Thread(m1);

Thread b2 = new Thread(m2);

Thread b3 = new Thread(m3);

b1.start();

b2.start();

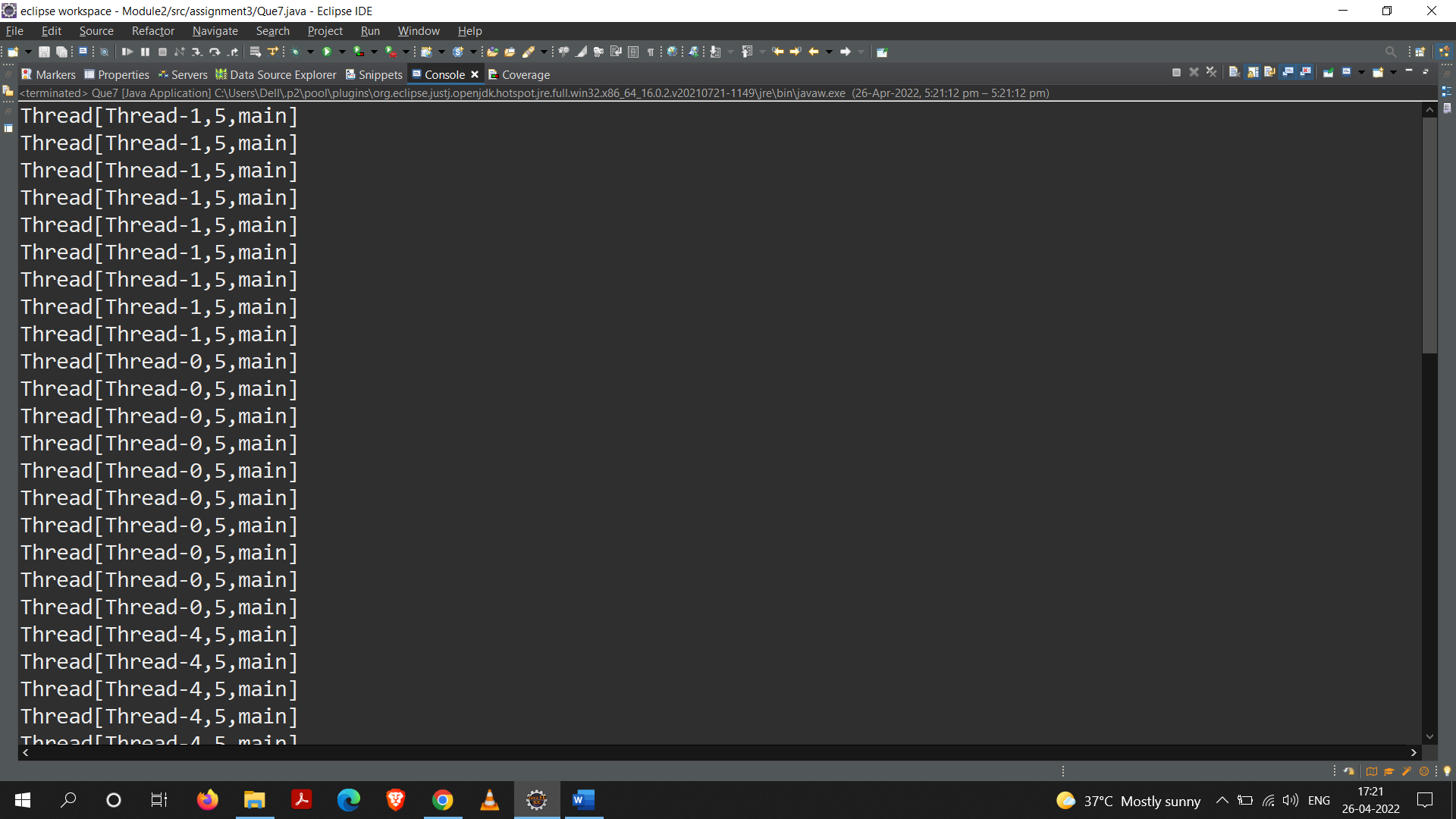
b3.start();

}

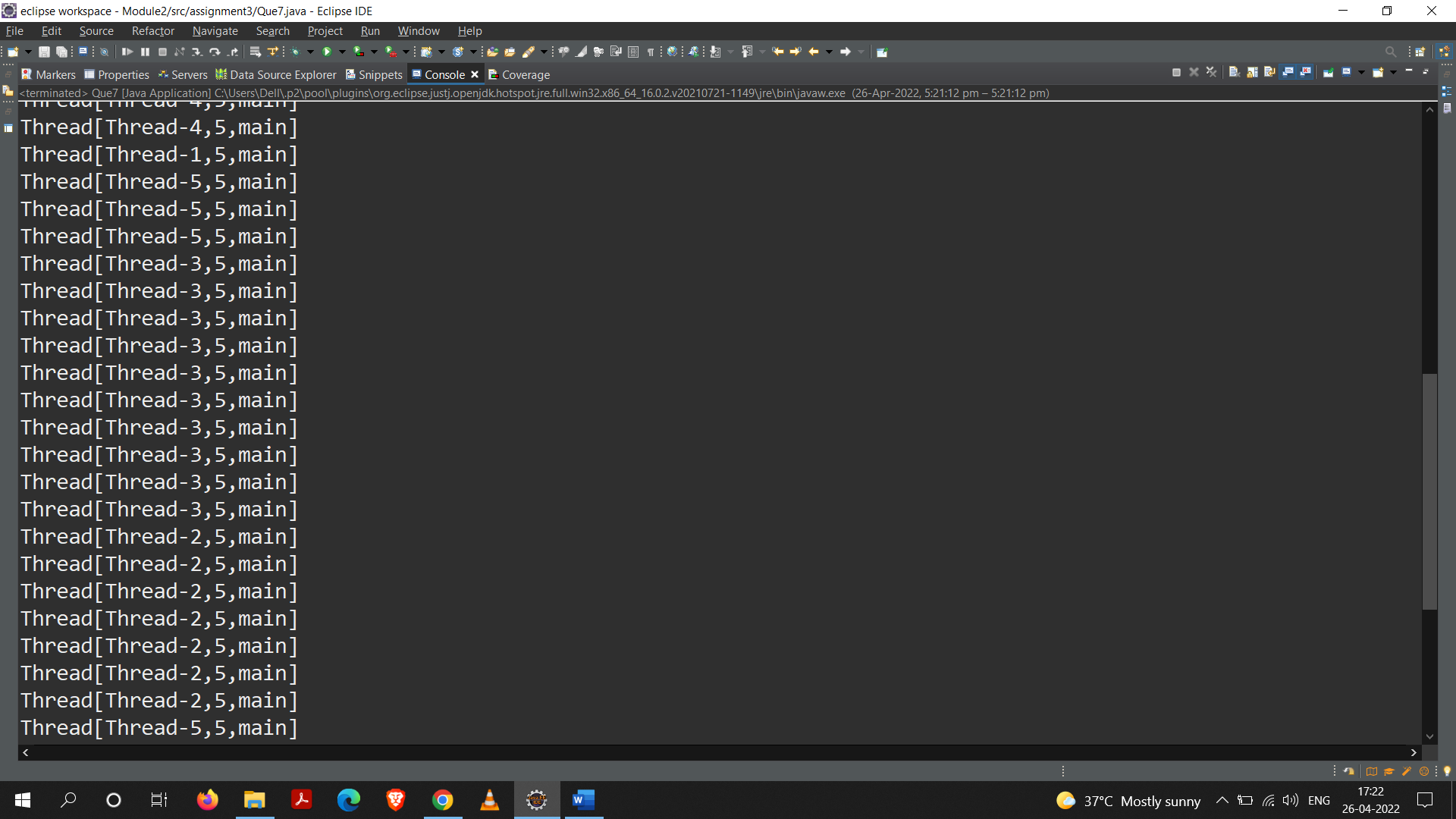
}

**Output:**

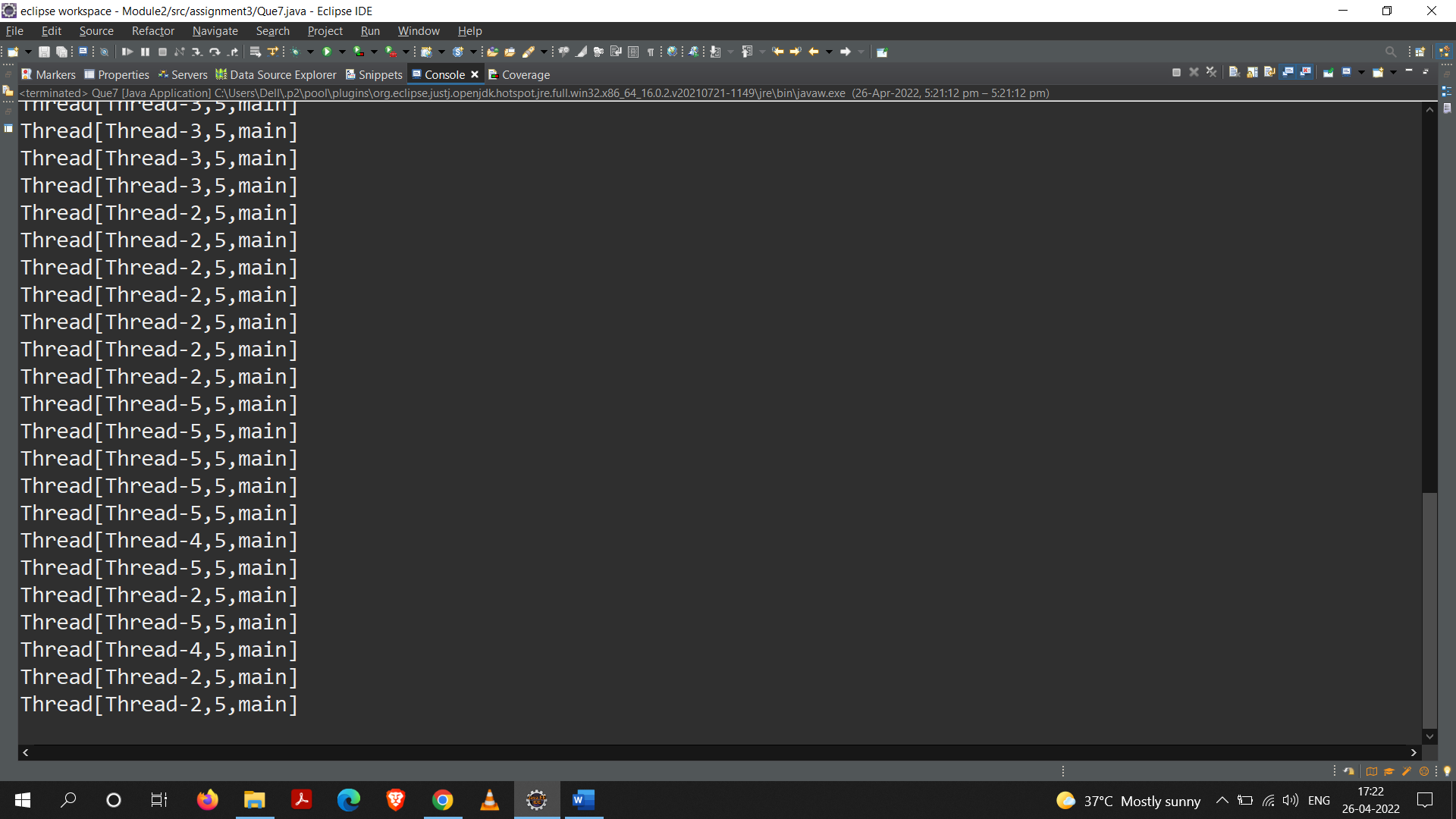
**Part – A**



**Part – B**



**Part – C**



**8. Assign same task and different task to multiple threads.**

**Code:**

package assignment3;

class MultiDemo extends Thread {

public void run() {

for(int i=1;i<=5;i++) {

System.***out***.println(Thread.*currentThread*().getName()+" : "+i+" ");

try{

Thread.*sleep*(100);

}

catch(Exception e){}

}

}

}

class Demo extends Thread {

void even() {

for(int i=0;i<=10;i++)

if(i%2==0)

System.***out***.println("EVEN :"+i);

}

void odd(){

for(int i=1;i<=10;i++)

if(i%2!=0)

System.***out***.println("ODD :"+i);

}

public void run() {

if(Thread.*currentThread*().getName().equals("Even"))

even();

else

odd();

}

}

public class Que8{

public static void main(String[] args)

{

MultiDemo t1=new MultiDemo();

MultiDemo t2=new MultiDemo();

t1.start();

try{

t1.join();

}catch(Exception e){}

t2.start();

try{

t2.join();

}catch(Exception e){}

System.***out***.println();

Demo a=new Demo();

Demo b=new Demo();

a.setName("Even");

b.setName("Odd");

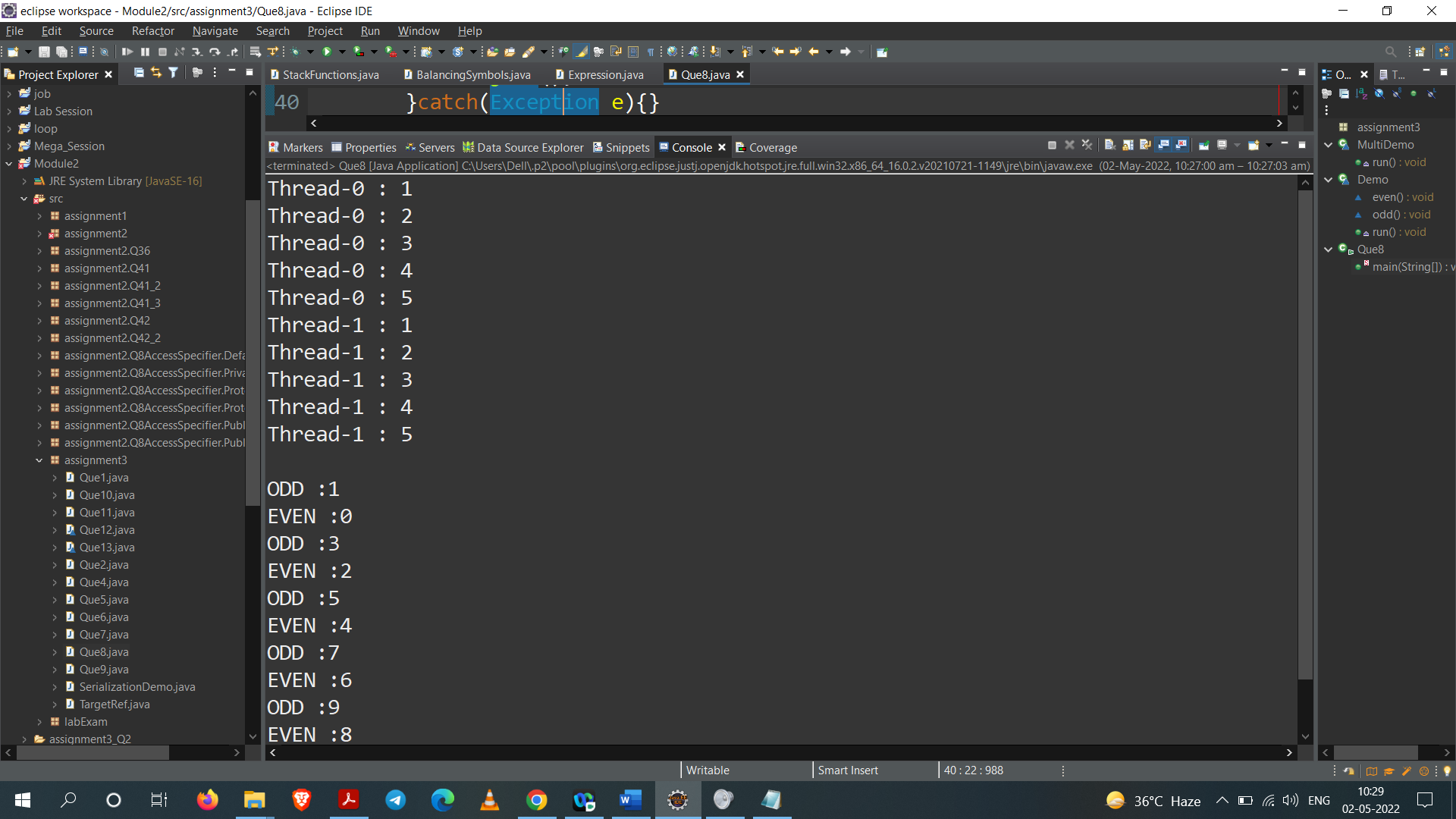
a.start();

b.start();

}

}

**Output:**



**9. Create a Deadlock class to demonstrate deadlock in multithreading environment.**

**Code:**

package assignment3;

class Util{

static void sleep(long millis) {

try {

Thread.*sleep*(millis);

}catch (InterruptedException e) {

e.printStackTrace();

}

}

}

class Shared {

synchronized void test1(Shared s2) {

System.***out***.println("test1-begin");

Util.*sleep*(1000);

s2.test2();

System.***out***.println("test1-end");

}

synchronized void test2() {

System.***out***.println("test2-begin");

Util.*sleep*(1000);

System.***out***.println("test2-end");

}

}

class Thread1 extends Thread {

private Shared s1;

private Shared s2;

public Thread1(Shared s1, Shared s2) {

this.s1 = s1;

this.s2 = s2;

}

*@Override*

public void run() {

s1.test1(s2);

}

}

class Thread2 extends Thread {

private Shared s1;

private Shared s2;

public Thread2(Shared s1, Shared s2) {

this.s1 = s1;

this.s2 = s2;

}

*@Override*

public void run() {

s2.test1(s1);

}

}

public class Que9 {

public static void main(String[] args) {

Shared s1 = new Shared();

Shared s2 = new Shared();

Thread1 t1 = new Thread1(s1, s2);

t1.start();

Thread2 t2 = new Thread2(s1, s2);

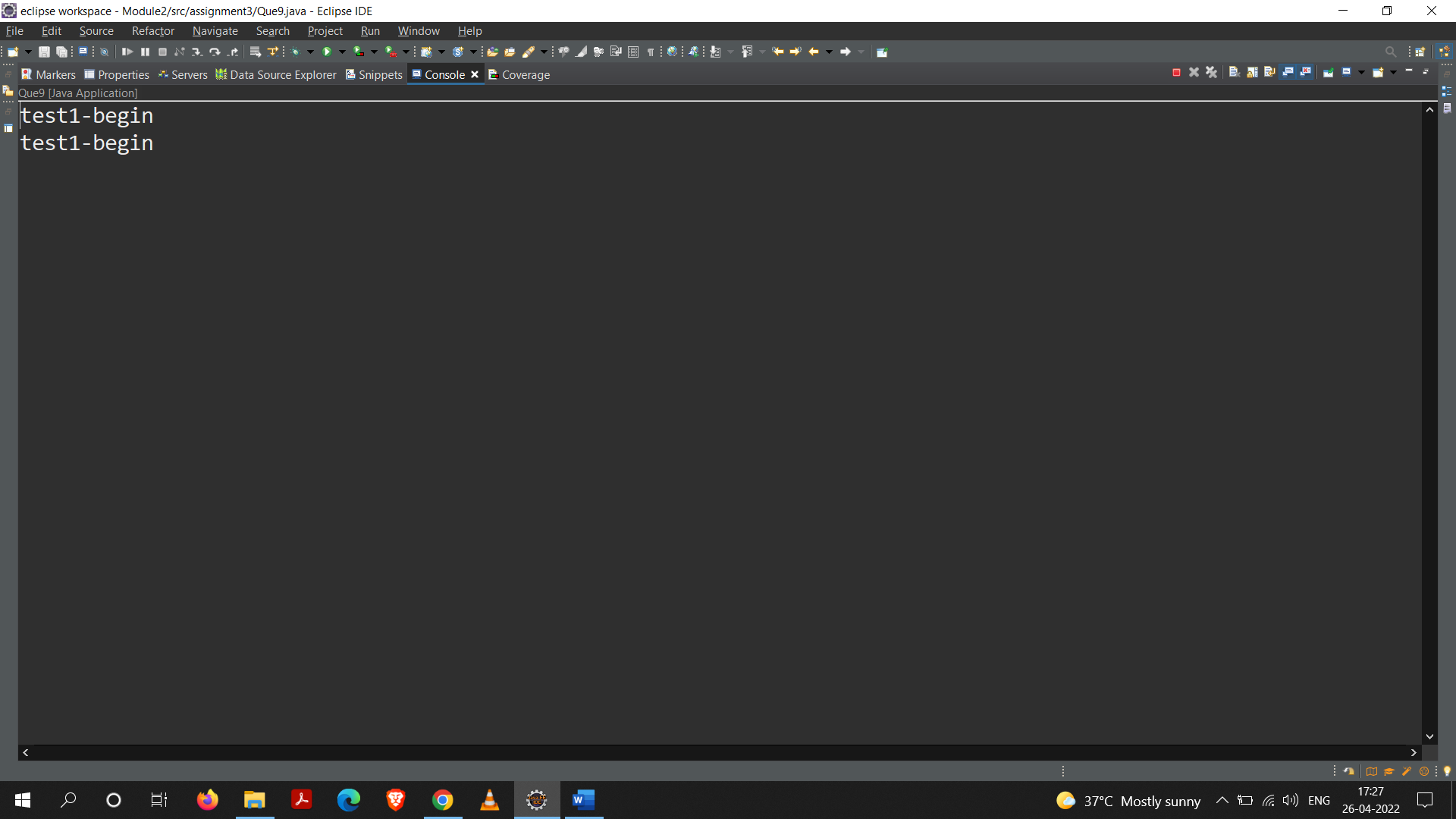
t2.start();

Util.*sleep*(2000);

}

}

**Output:**



**10. Implement wait, notify and notifyAll methods.**

**Code:**

package assignment3;

public class Que10 {

private static final long ***SLEEP\_INTERVAL*** = 3000;

private boolean running = true;

private Thread thread;

public void start() {

print("Inside start() method");

thread = new Thread(new Runnable() {

*@Override*

public void run() {

print("Inside run() method");

try {

Thread.*sleep*(***SLEEP\_INTERVAL***);

} catch(InterruptedException e) {

Thread.*currentThread*().interrupt();

}

synchronized(Que10.this) {

running = false;

Que10.this.notify();

}

}

});

thread.start();

}

public void join() throws InterruptedException {

print("Inside join() method");

synchronized(this) {

while(running) {

print("Waiting for the peer thread to finish.");

wait(); //waiting, not running

}

print("Peer thread finished.");

}

}

private void print(String s) {

System.***out***.println(s);

}

public static void main(String[] args) throws InterruptedException {

Que10 test = new Que10();

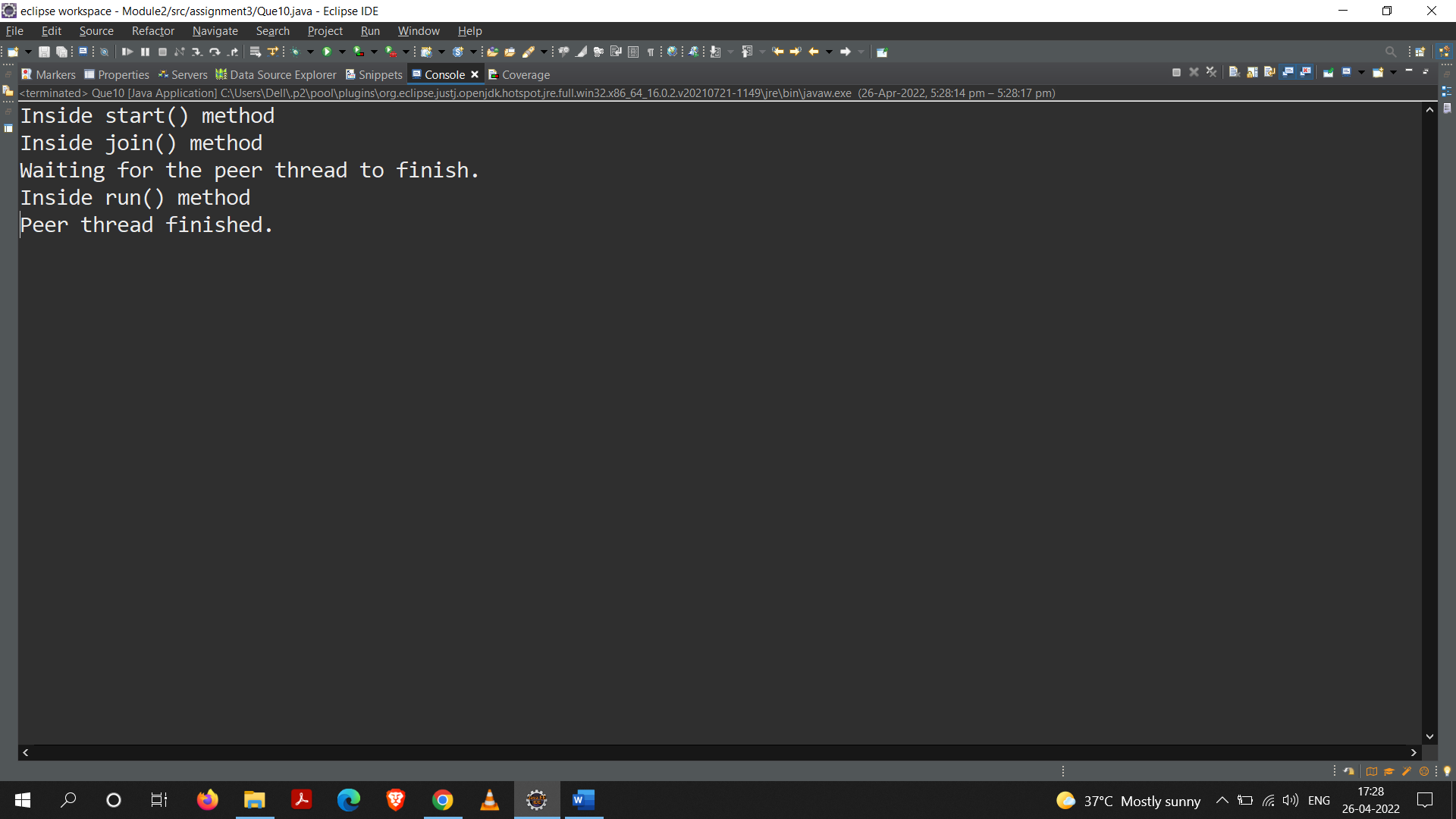
test.start();

test.join();

}

}

**Output:**



**11. Create multiple threads using anonymous inner classes.**

**Code:**

package assignment3;

class Conditions{

synchronized public void dispNos(){

System.***out***.println("Even Numbers in First 250 No :");

for(int i=0;i<=250;i=i+2){

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

try {

Thread.*sleep*(50);

} catch (Exception e) { }

}

System.***out***.println();

}

synchronized public void PrintTable(int n){

System.***out***.println("\nPrinting Table : ");

for (int i=1;i<=10;i++)

System.***out***.println(n+" \* "+i+" = "+(n\*i));

try {

Thread.*sleep*(50);

} catch (Exception e) {}

if(i%30==0)

System.***out***.println();

}

synchronized public void allchar(){

System.***out***.println("\nAll Characters form 0 to 127 : ");

for (int i = 0; i <=127; i++) {

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

try {

Thread.*sleep*(50);

} catch (Exception e) {}

if(i%10==0)

System.***out***.println();

}

}

}

public class Que11 {

public static void main(String[] args) {

Conditions c=new Conditions();

new Thread(){

public void run(){

c.dispNos();

}

}.start();

new Thread(){

public void run(){

c.PrintTable(2);

}

}.start();

new Thread(){

public void run(){

c.allchar();

}

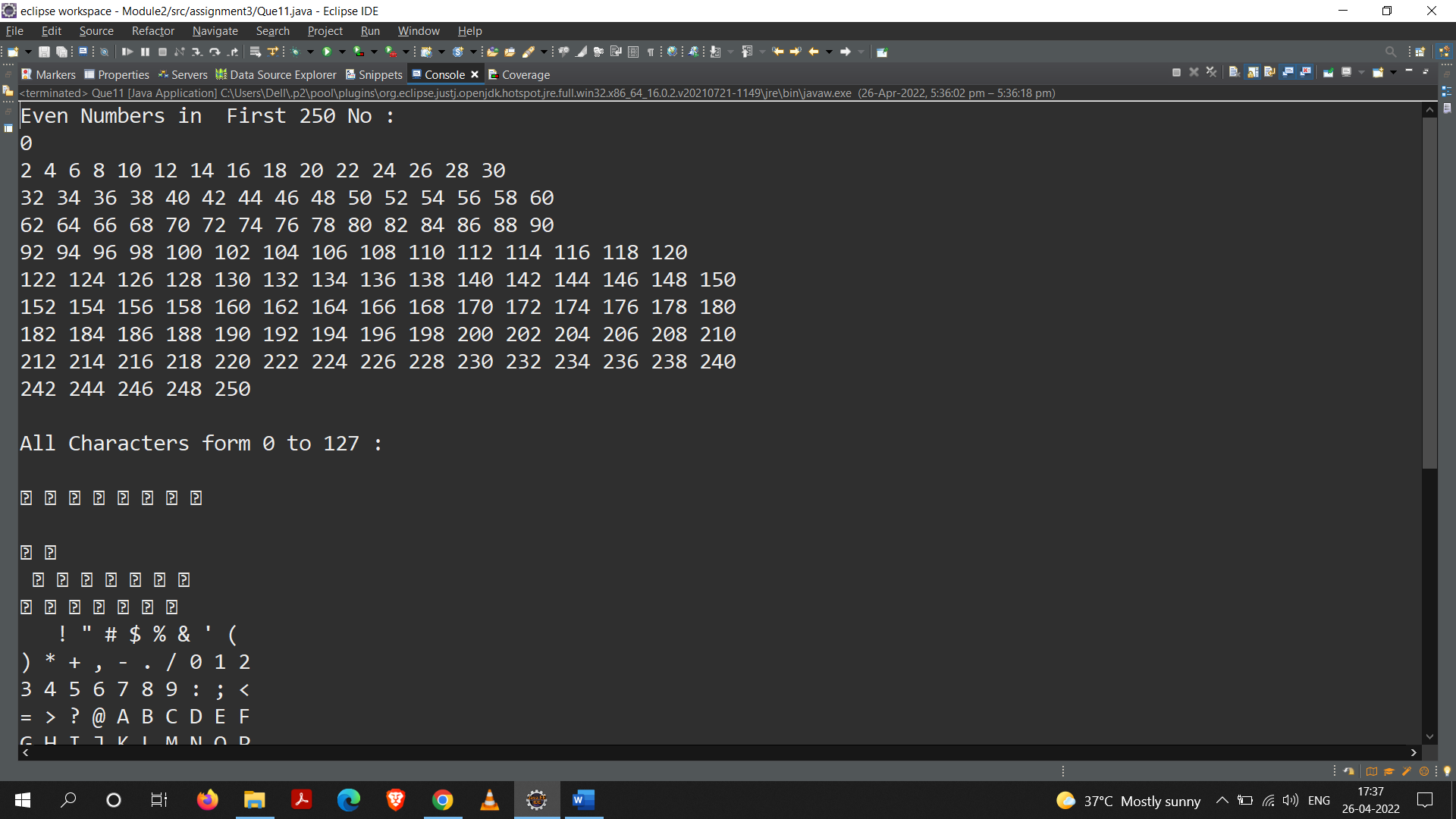
}.start();

}

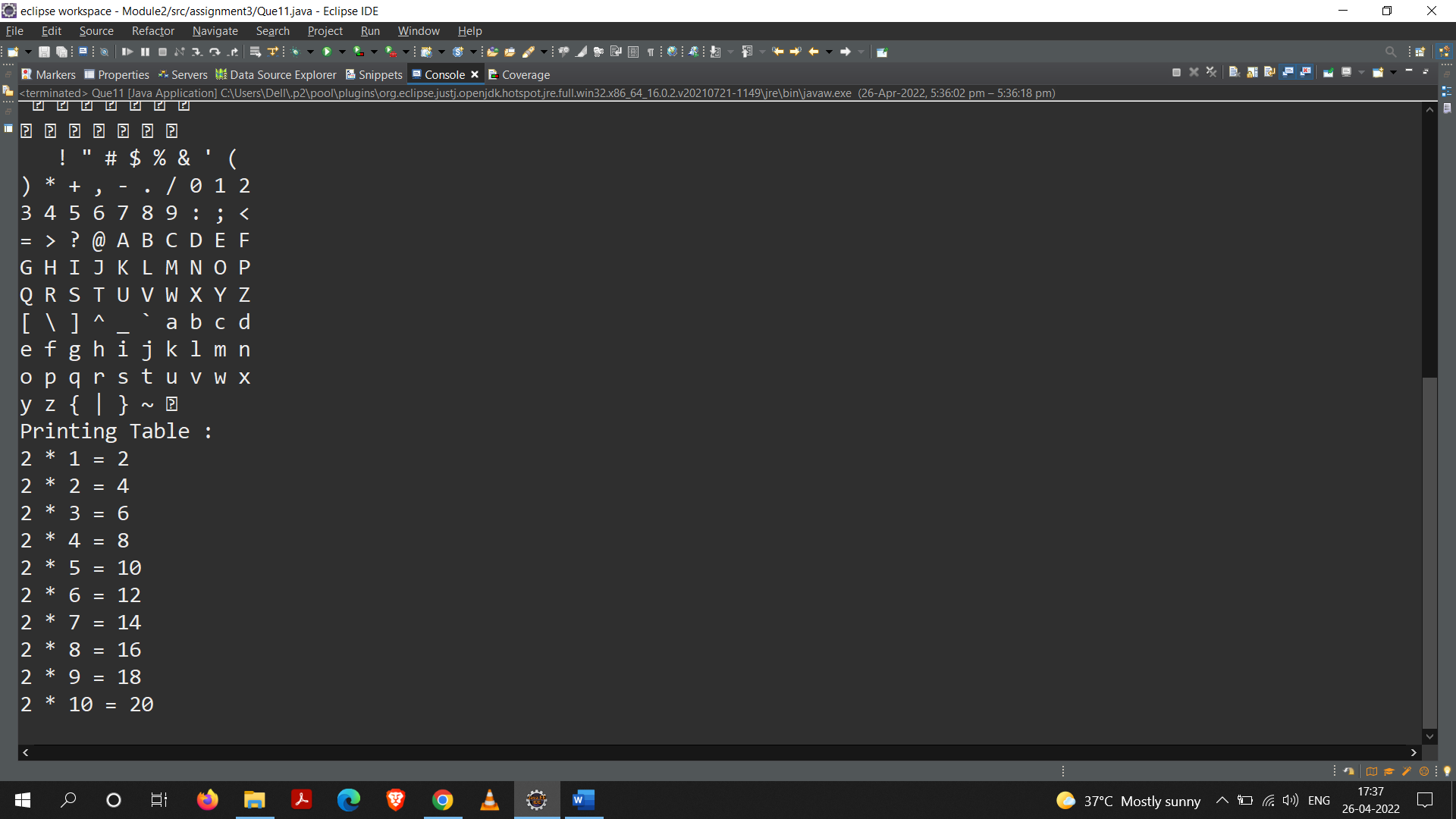
}

**Output:**

**Part – A**



**Part – B**



**12.Write a program for example of try and catch block. In this check whether the given array size is negative or not**

**Code:**

package assignment3;

class Que12 {

*@SuppressWarnings*("unused")

public static void main(String a[])

{

try

{

int array[] = new int[-2];

}

catch(NegativeArraySizeException n)

{

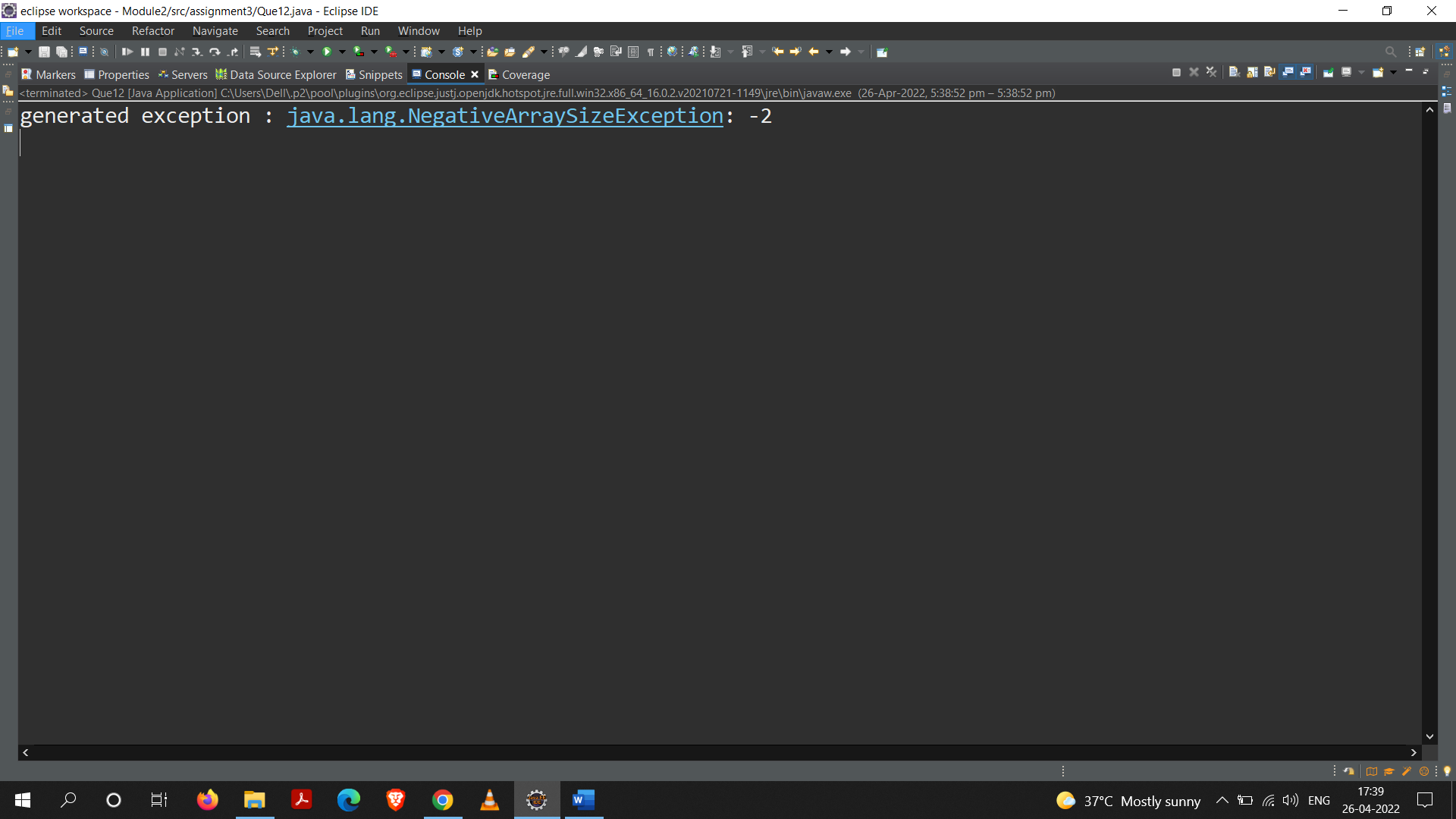
System.***out***.println("generated exception : " + n);

}

}

}

**Output:**



**13.Write a program to create a class MyThread in this class a constructor, call the base class constructor, using super and starts the thread. The run method of the class starts after this. It**

**can be observed that both main thread and created child thread are executed concurrently.**

**Code:**

package assignment3;

class MyThread extends Thread

{

MyThread()

{

super("Using Thread class");

System.***out***.println ("Child thread:" + this);

start();

}

public void run()

{

try

{

int i;

for (i = 5; i > 0; i--)

{

System.***out***.println ("Child thread" + i);

Thread.*sleep* (500);

}

} catch (InterruptedException e) {

System.***out***.println(e);

}

}

}

class Que13 {

public static void main(String args[])

{

new MyThread();

try {

int k;

for (k = 5; k < 0; k--)

{

Thread.*sleep*(1000);

}

}catch (InterruptedException e) {

System.***out***.println(e);

}

}

}

**Output:**

