import java.util.concurrent.CopyOnWriteArrayList;

import java.util.Iterator;

public class ConcurrentModificationResolved {

private static CopyOnWriteArrayList<Integer> sharedList = new CopyOnWriteArrayList<>();

static class Thread1 extends Thread {

@Override

public void run() {

try {

Iterator<Integer> iterator = sharedList.iterator();

while (iterator.hasNext()) {

Integer value = iterator.next();

System.out.println("Thread 1: " + value);

try { Thread.sleep(50); } catch (InterruptedException e) {}

}

} catch (Exception e) {

System.err.println("Thread 1: Exception caught.");

}

}

}

static class Thread2 extends Thread {

@Override

public void run() {

try {

for (int i = 0; i < 5; i++) {

sharedList.add(i);

System.out.println("Thread 2: Added " + i);

try { Thread.sleep(30); } catch (InterruptedException e) {}

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

public static void main(String[] args) {

sharedList.add(1);

sharedList.add(2);

sharedList.add(3);

sharedList.add(4);

Thread1 thread1 = new Thread1();

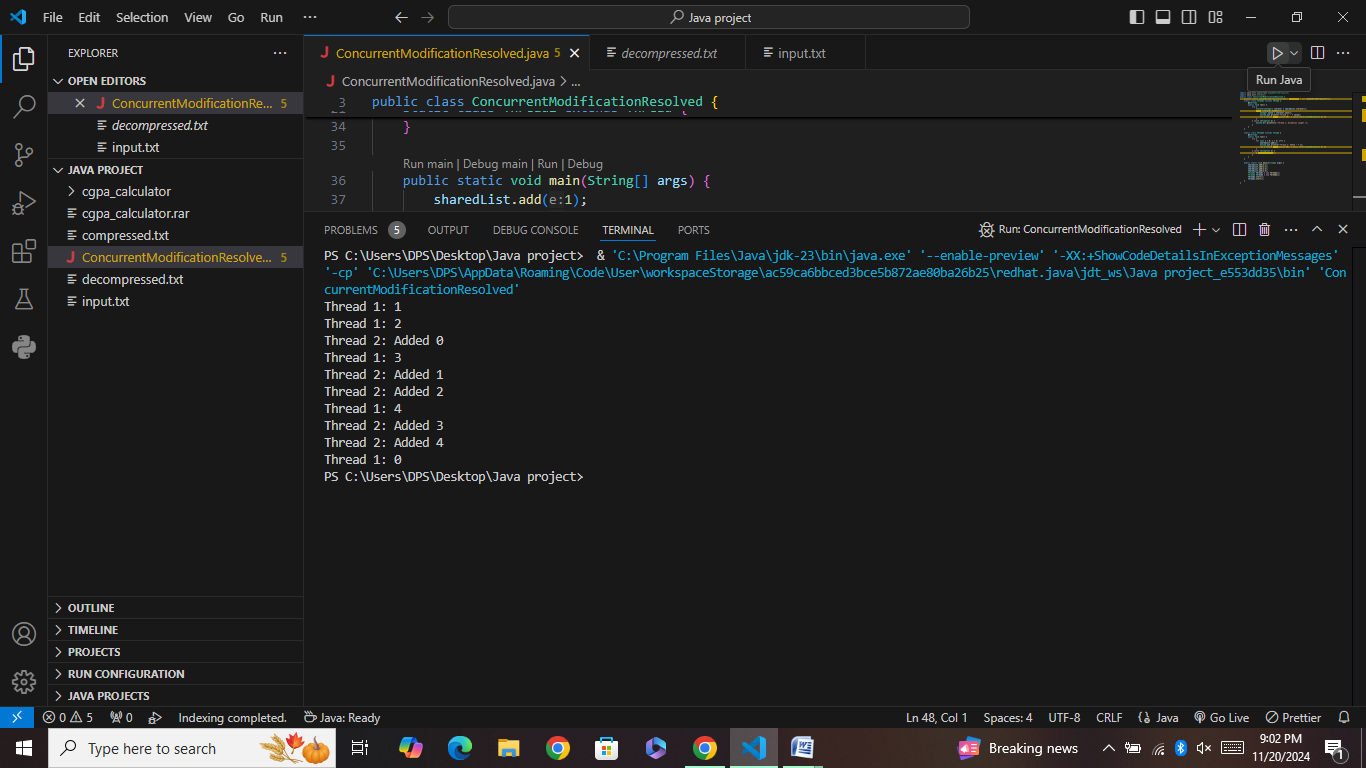
Thread2 thread2 = new Thread2();

thread2.start();

thread1.start();

}

}



Explanation ;

The Concurrent Modification Resolved program demonstrates safe concurrent access to a shared list using Copy On Write Array List.

Thread1: Iterates over the list and prints its elements, pausing for 50 milliseconds between prints.

Thread2: Adds integers (0-4) to the list, pausing for 30 milliseconds between additions.

The Copy On Write Array List ensures thread safety by creating a new copy of the list each time it is modified, preventing issues like Con current Modification Exception when Thread1 iterates while Thread2 modifies the list.