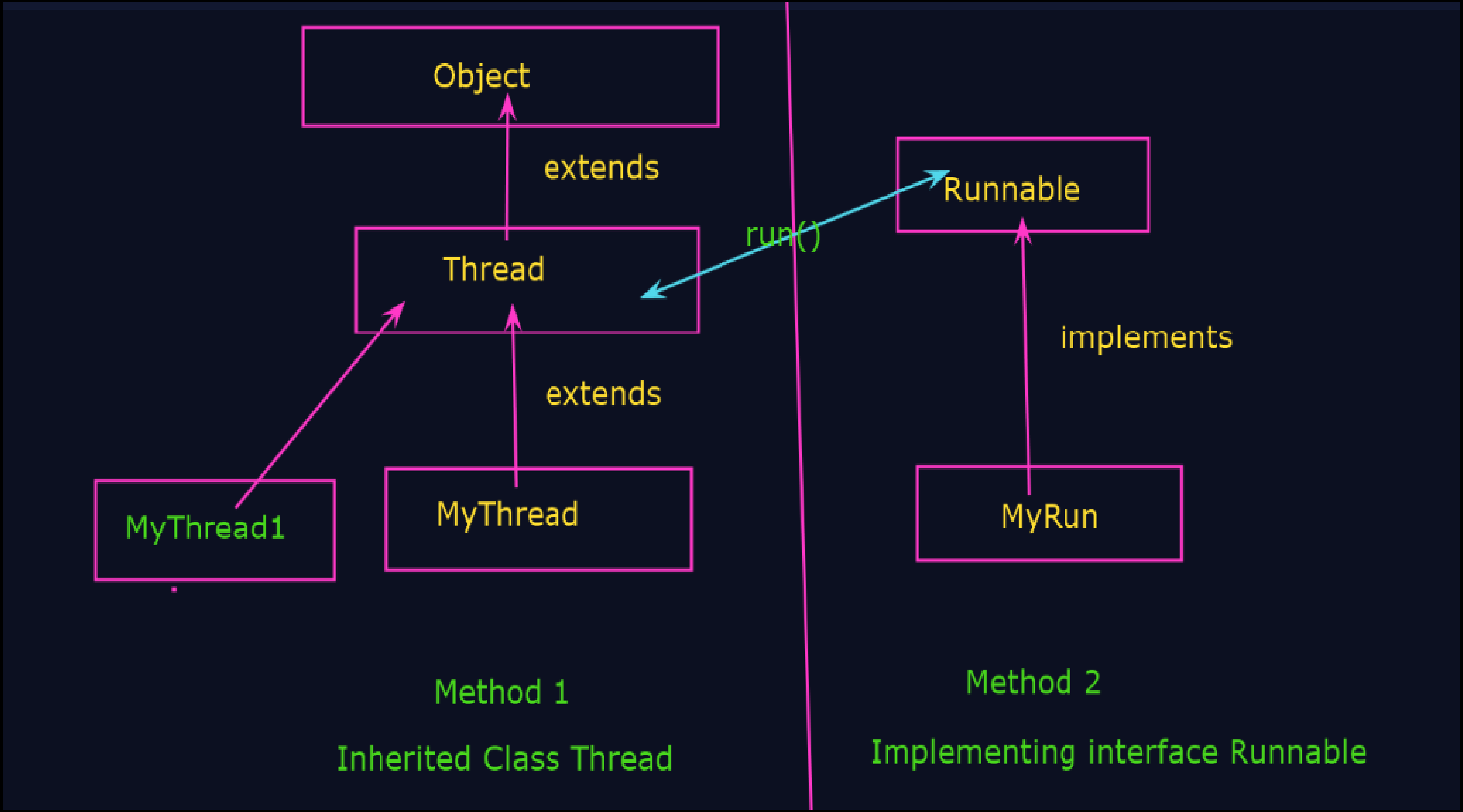
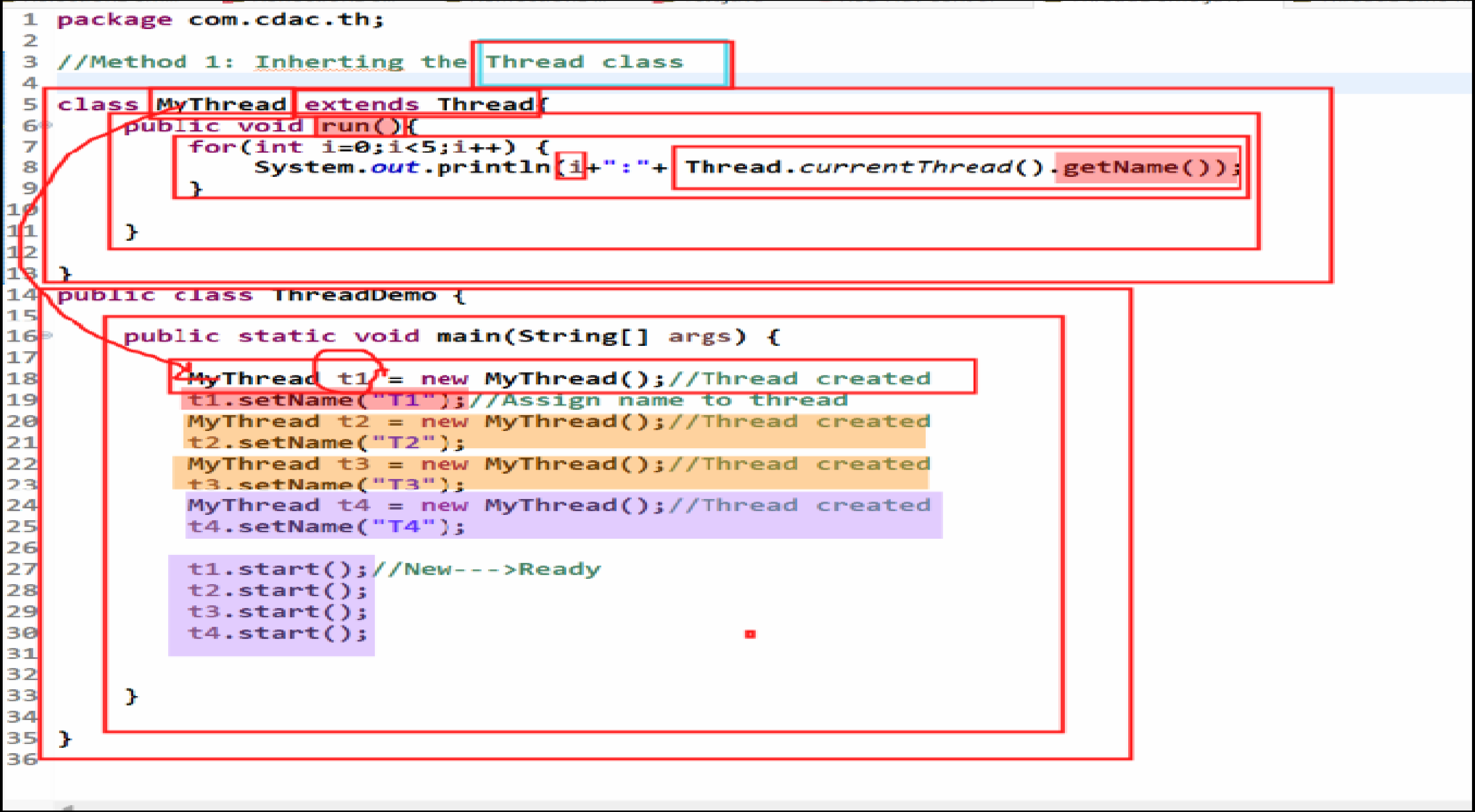
Object Oriented Programming with Java (OOPJ)

Session 17: Multithreading

Kiran Waghmare


Object
extends
Thread
run()
Runnable
implements
extends
MyThread1
MyThread
MyRun
Method 1
Inherited Class Thread
Method 2
Implementing interface Runnable

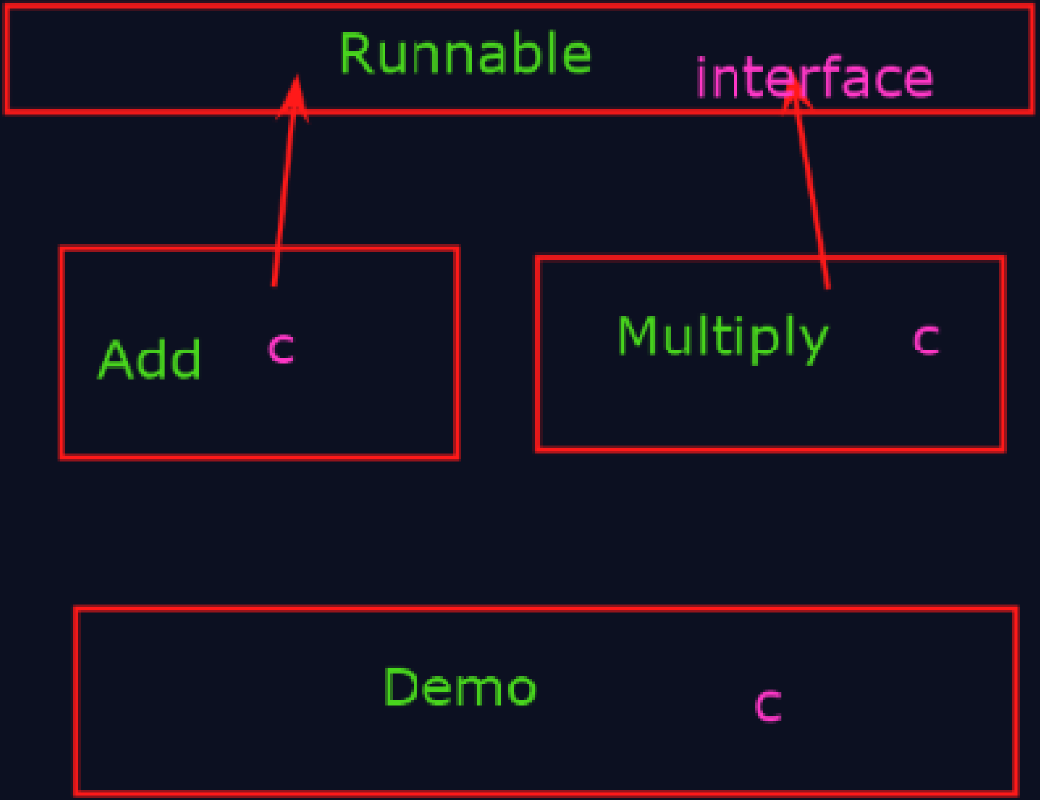


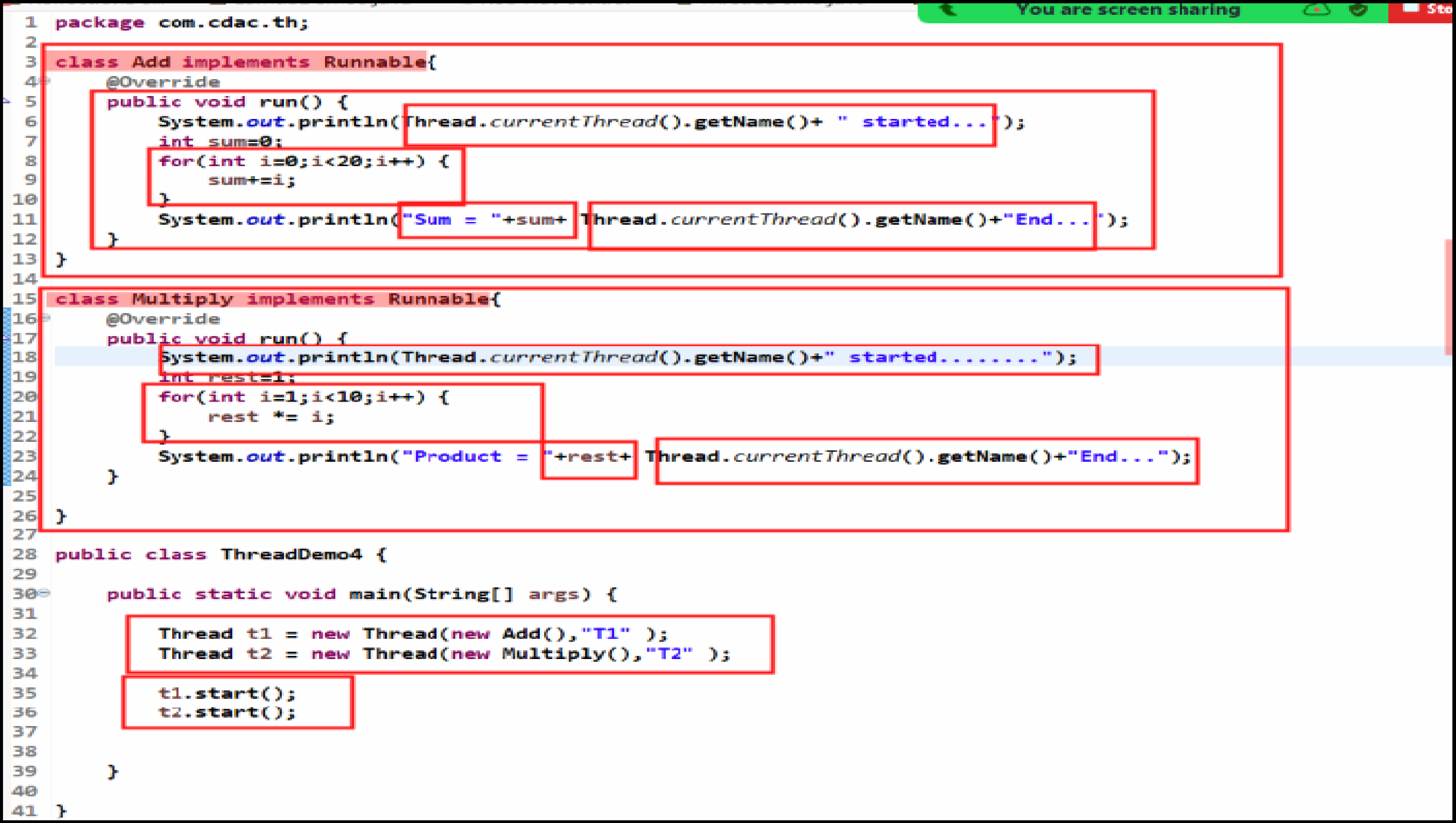
```java
package com.cdac.th;

//Method 1: Inherting the Thread class

class MyThread extends Thread{
    public void run(){
        for(int i=0;i<5;i++) {
            System.out.println(i+":"+ Thread.currentThread().getName());
        }

    }

}
public class ThreadDemo {

    public static void main(String[] args) {

        MyThread t1 = new MyThread();//Thread created
        t1.setName("T1");//Assign name to thread
        MyThread t2 = new MyThread();//Thread created
        t2.setName("T2");
        MyThread t3 = new MyThread();//Thread created
        t3.setName("T3");
        MyThread t4 = new MyThread();//Thread created
        t4.setName("T4");

        t1.start();//New--->Ready
        t2.start();
        t3.start();
        t4.start();


    }

}
```

Thread class inherit : Class refersence

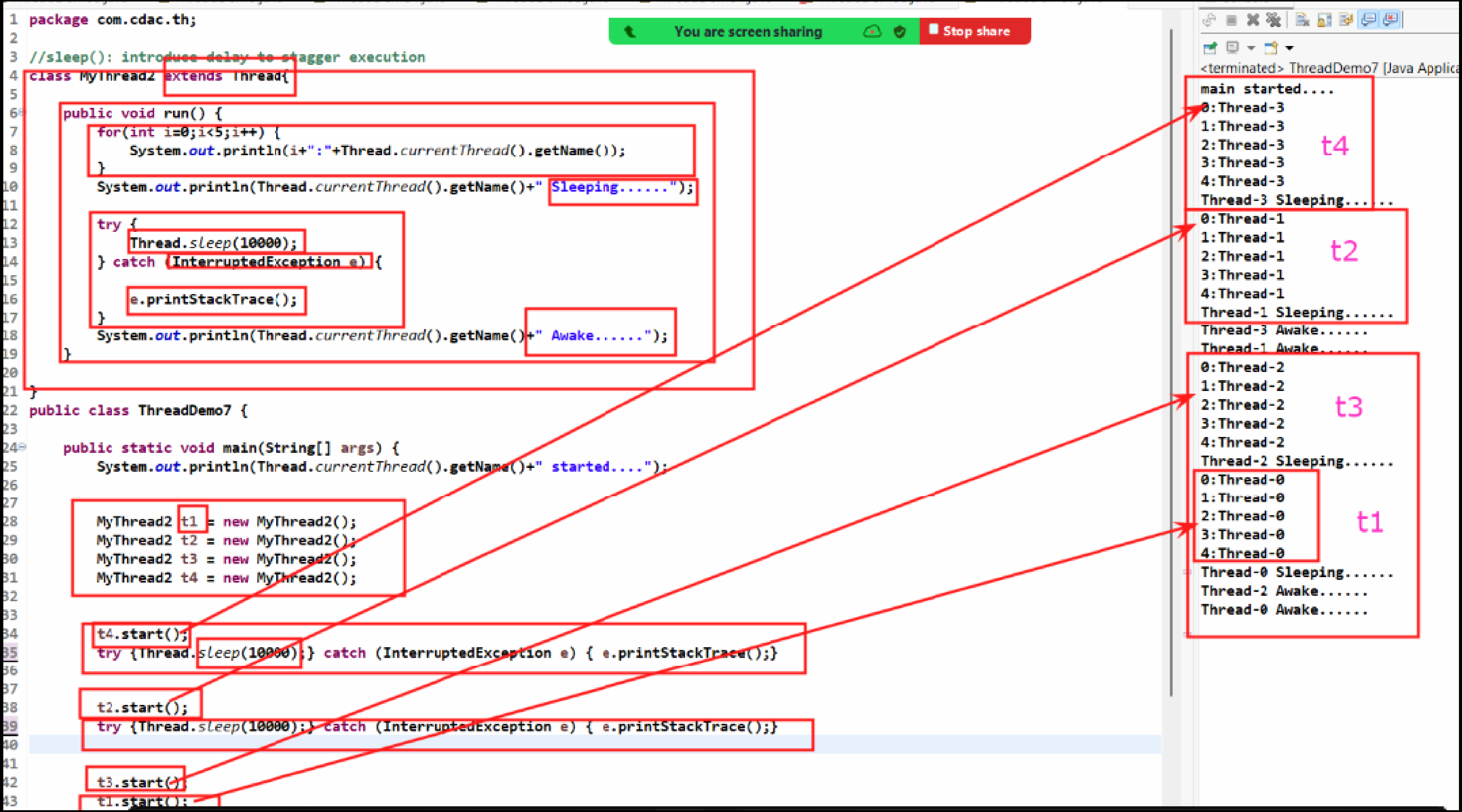
Runnable interface implement : class reference

Runnable interface with lambda function and Anonymous class:



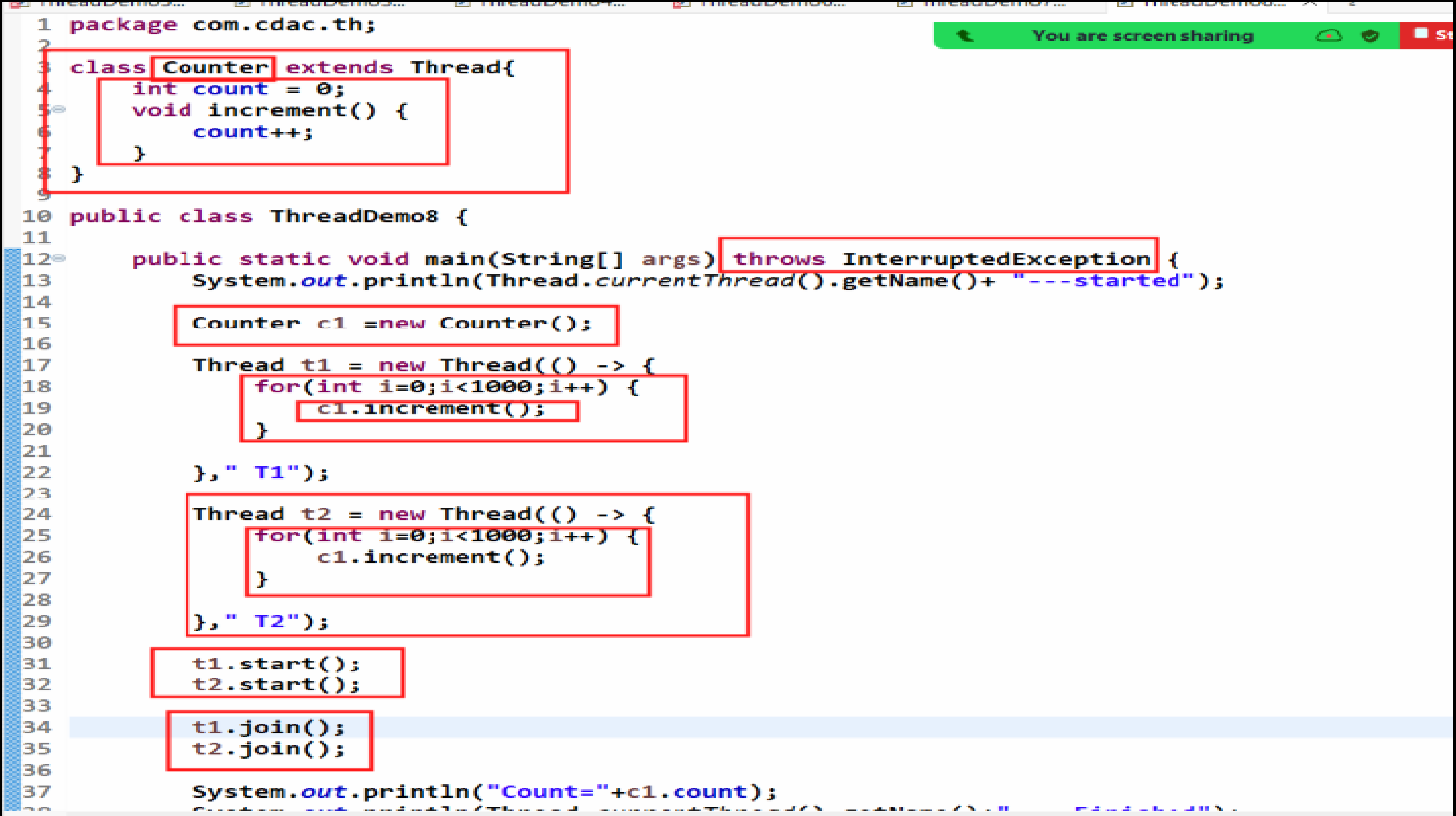


```java
package com.cdac.th;

class Add implements Runnable{
    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName()+ " started...");
        int sum=0;
        for(int i=0;i<20;i++) {
            sum+=i;
        }
        System.out.println("Sum = "+sum+ Thread.currentThread().getName()+"End...");
    }
}

class Multiply implements Runnable{
    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName()+" started........");
        int rest=1;
        for(int i=1;i<10;i++) {
            rest *= i;
        }
        System.out.println("Product = "+rest+ Thread.currentThread().getName()+"End...");
    }

}

public class ThreadDemo4 {

    public static void main(String[] args) {

        Thread t1 = new Thread(new Add(),"T1" );
        Thread t2 = new Thread(new Multiply(),"T2" );

        t1.start();
        t2.start();


    }

}
```

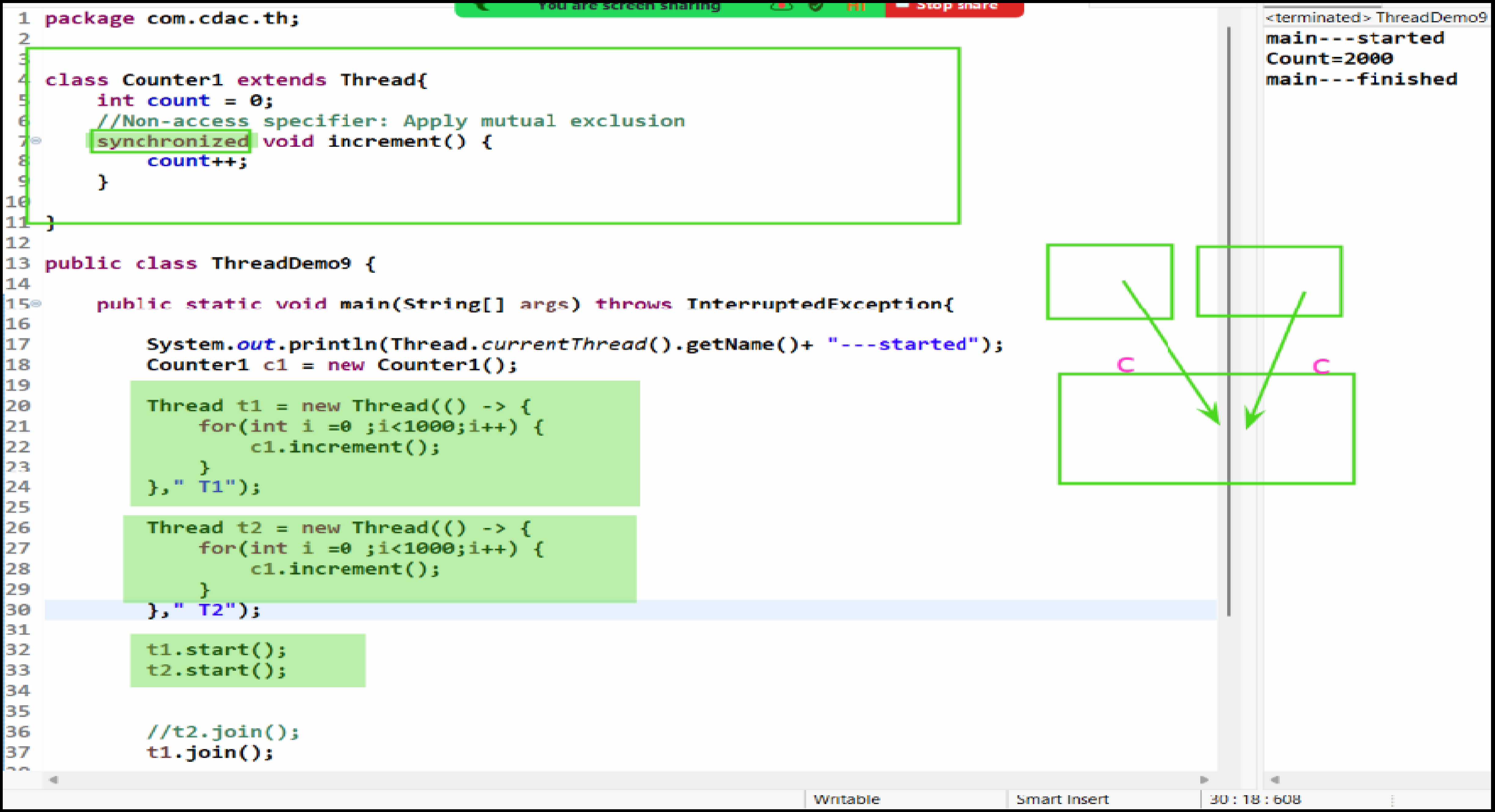


```java
package com.cdac.th;

//sleep(): introduce delay to stagger execution
class MyThread2 extends Thread{

    public void run() {
        for(int i=0;i<5;i++) {
            System.out.println(i+":"+Thread.currentThread().getName());
        }
        System.out.println(Thread.currentThread().getName()+" Sleeping......");

        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {

            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName()+" Awake......");
    }

}
public class ThreadDemo7 {

    public static void main(String[] args) {
        System.out.println(Thread.currentThread().getName()+" started....");


        MyThread2 t1 = new MyThread2();
        MyThread2 t2 = new MyThread2();
        MyThread2 t3 = new MyThread2();
        MyThread2 t4 = new MyThread2();


        t4.start();
        try {Thread.sleep(10000);} catch (InterruptedException e) { e.printStackTrace();}

        t2.start();
        try {Thread.sleep(10000);} catch (InterruptedException e) { e.printStackTrace();}


        t3.start();
        t1.start();
```

```
main started....
0:Thread-3
1:Thread-3
2:Thread-3
3:Thread-3
4:Thread-3
Thread-3 Sleeping......
0:Thread-1
1:Thread-1
2:Thread-1
3:Thread-1
4:Thread-1
Thread-1 Sleeping......
Thread-3 Awake......
Thread-1 Awake......
0:Thread-2
1:Thread-2
2:Thread-2
3:Thread-2
4:Thread-2
Thread-2 Sleeping......
0:Thread-0
1:Thread-0
2:Thread-0
3:Thread-0
4:Thread-0
Thread-0 Sleeping......
Thread-2 Awake......
Thread-0 Awake......
```

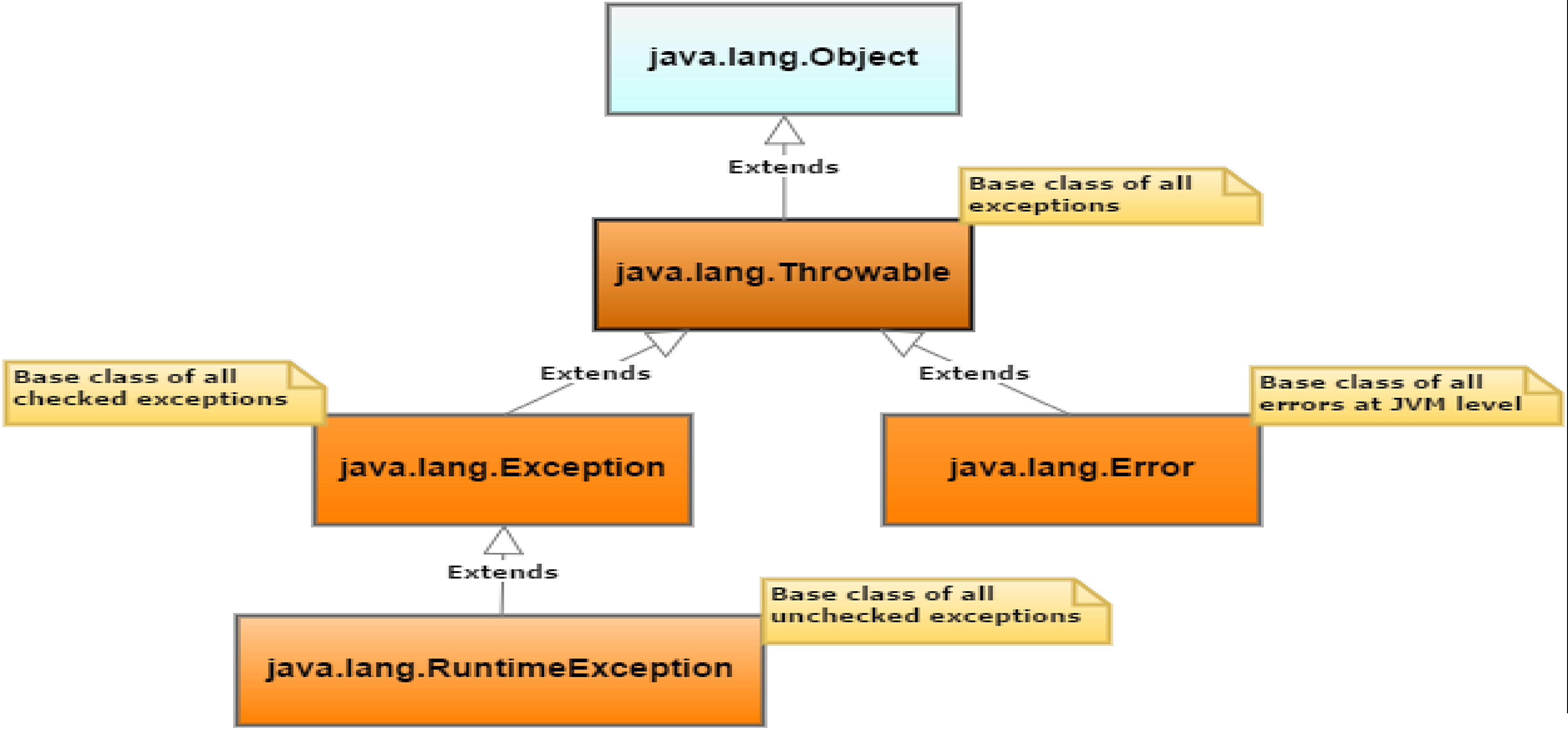


```java
package com.cdac.th;

class Counter extends Thread{
    int count = 0;
    void increment() {
        count++;
    }
}

public class ThreadDemo8 {

    public static void main(String[] args) throws InterruptedException {
        System.out.println(Thread.currentThread().getName()+ "---started");

        Counter c1 =new Counter();

        Thread t1 = new Thread(() -> {
            for(int i=0;i<1000;i++) {
                c1.increment();
            }

        }," T1");

        Thread t2 = new Thread(() -> {
            for(int i=0;i<1000;i++) {
                c1.increment();
            }

        }," T2");

        t1.start();
        t2.start();

        t1.join();
        t2.join();

        System.out.println("Count="+c1.count);
```



```java
package com.cdac.th;


class Counter1 extends Thread{
    int count = 0;
    //Non-access specifier: Apply mutual exclusion
    synchronized void increment() {
        count++;
    }

}

public class ThreadDemo9 {

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

        System.out.println(Thread.currentThread().getName()+ "---started");
        Counter1 c1 = new Counter1();

        Thread t1 = new Thread(() -> {
            for(int i =0 ;i<1000;i++) {
                c1.increment();
            }
        }," T1");

        Thread t2 = new Thread(() -> {
            for(int i =0 ;i<1000;i++) {
                c1.increment();
            }
        }," T2");

        t1.start();
        t2.start();


        //t2.join();
        t1.join();
```

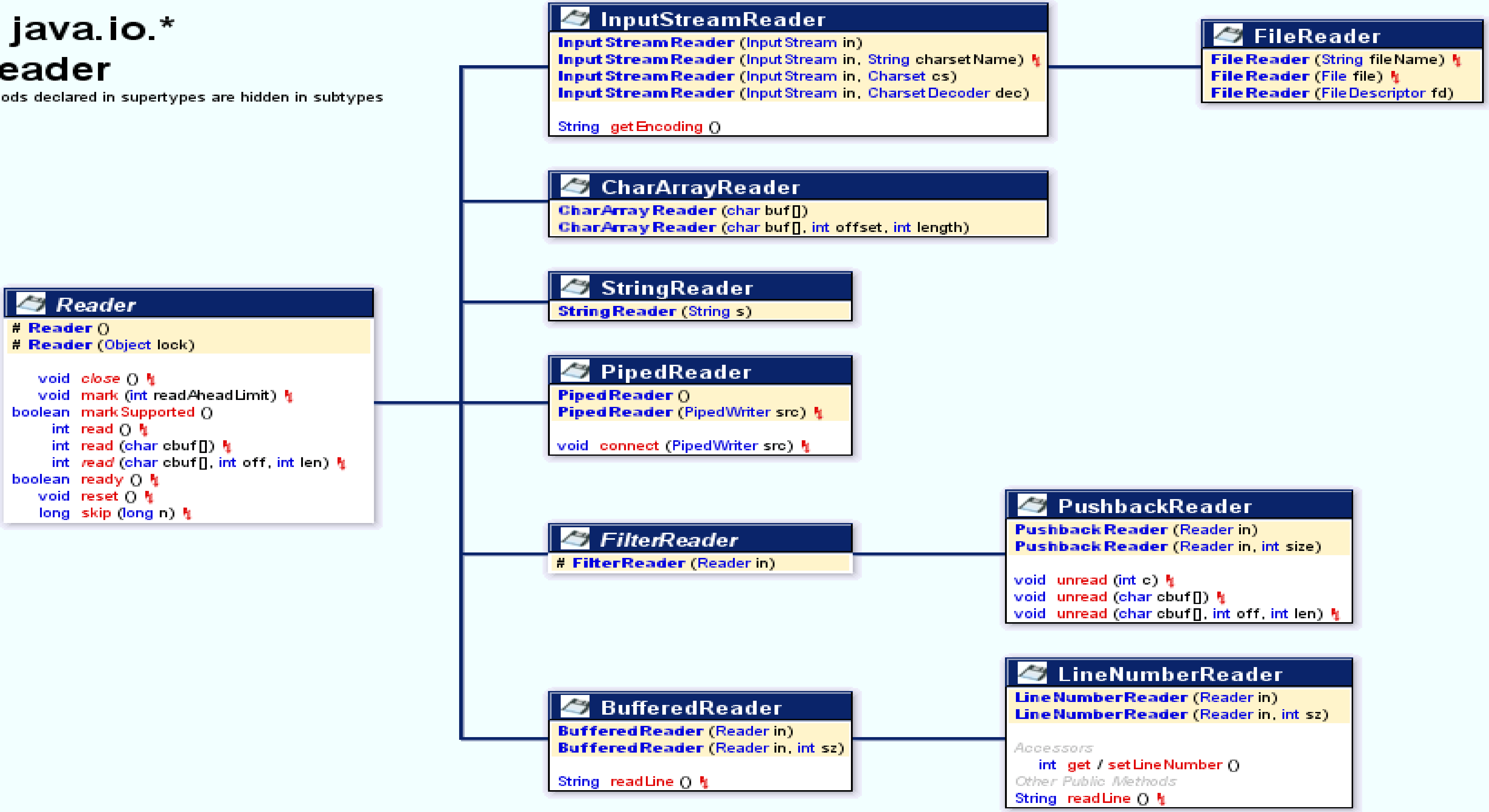
```
<terminated> ThreadDemo9
main---started
Count=2000
main---finished
```


java.lang.Object
Extends
java.lang.Throwable
Base class of all exceptions
Extends
Extends
java.lang.Exception
Base class of all checked exceptions
java.lang.Error
Base class of all errors at JVM level
Extends
java.lang.RuntimeException
Base class of all unchecked exceptions

Exception Hierarchy in Java

java.io.* Reader

Methods declared in supertypes are hidden in subtypes



```
Reader
# Reader ()
# Reader (Object lock)

   void  close ()
   void  mark (int readAheadLimit)
boolean  markSupported ()
    int  read ()
    int  read (char cbuf[])
    int  read (char cbuf[], int off, int len)
boolean  ready ()
   void  reset ()
   long  skip (long n)
```

```
InputStreamReader
InputStreamReader (InputStream in)
InputStreamReader (InputStream in, String charsetName)
InputStreamReader (InputStream in, Charset cs)
InputStreamReader (InputStream in, CharsetDecoder dec)

String  getEncoding ()
```

```
FileReader
FileReader (String fileName)
FileReader (File file)
FileReader (FileDescriptor fd)
```

```
CharArrayReader
CharArrayReader (char buf[])
CharArrayReader (char buf[], int offset, int length)
```

```
StringReader
StringReader (String s)
```

```
PipedReader
PipedReader ()
PipedReader (PipedWriter src)

void  connect (PipedWriter src)
```

```
FilterReader
# FilterReader (Reader in)
```

```
PushbackReader
PushbackReader (Reader in)
PushbackReader (Reader in, int size)

void  unread (int c)
void  unread (char cbuf[])
void  unread (char cbuf[], int off, int len)
```

```
BufferedReader
BufferedReader (Reader in)
BufferedReader (Reader in, int sz)

String  readLine ()
```

```
LineNumberReader
LineNumberReader (Reader in)
LineNumberReader (Reader in, int sz)

Accessors
   int  get / setLineNumber ()
Other Public Methods
String  readLine ()
```