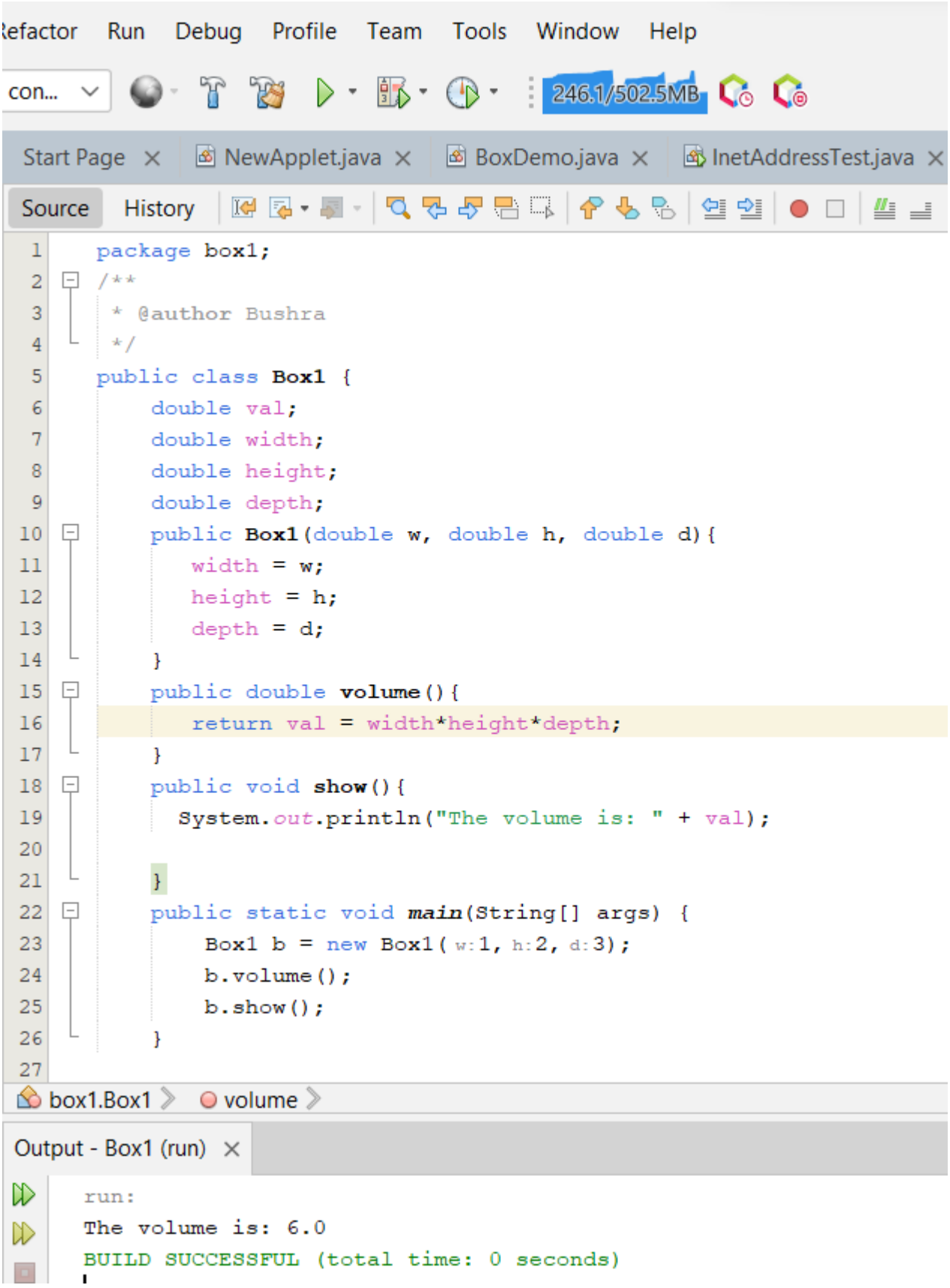
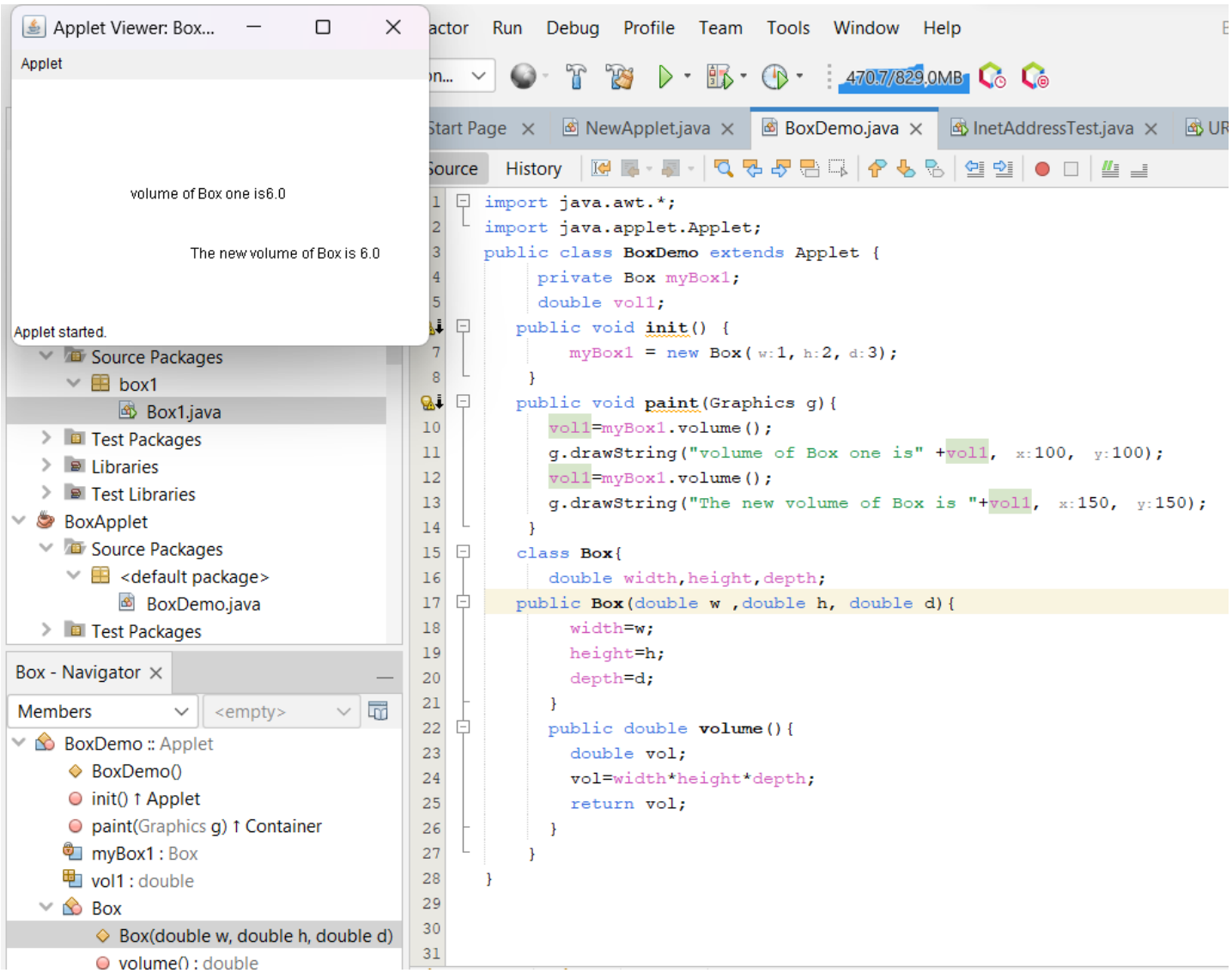
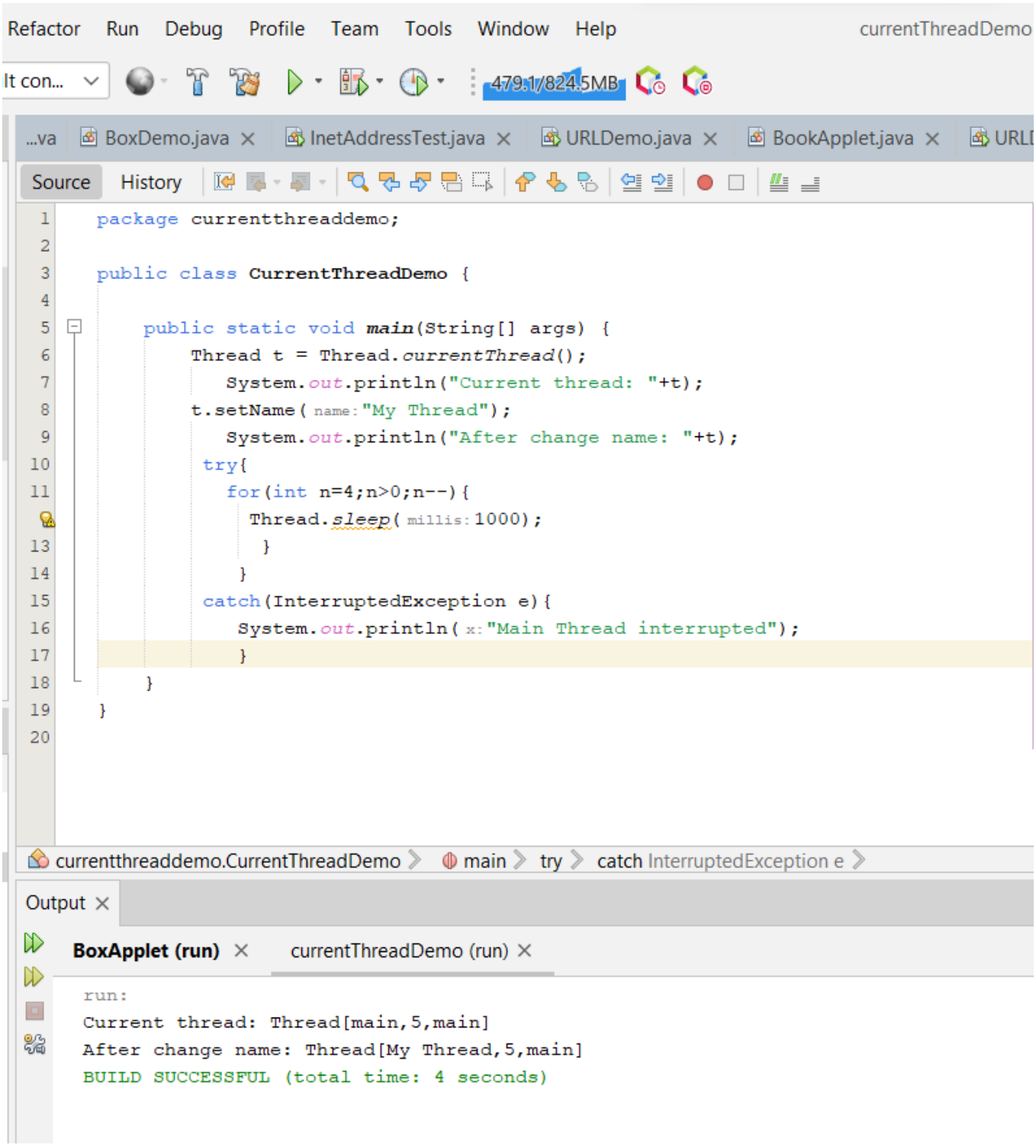

Refactor Run Debug Profile Team Tools Window Help
con...
246.1/502.5MB
Start Page
NewApplet.java
BoxDemo.java
InetAddressTest.java
Source
History
1 package box1;
2 /**
3 * @author Bushra
4 */
5 public class Box1 {
6 double val;
7 double width;
8 double height;
9 double depth;
10 public Box1(double w, double h, double d){
11 width = w;
12 height = h;
13 depth = d;
14 }
15 public double volume(){
16 return val = width*height*depth;
17 }
18 public void show(){
19 System.out.println("The volume is: " + val);
20
21 }
22 public static void main(String[] args) {
23 Box1 b = new Box1(w:1, h:2, d:3);
24 b.volume();
25 b.show();
26 }
27
box1.Box1
volume
Output - Box1 (run)
run:
The volume is: 6.0
BUILD SUCCESSFUL (total time: 0 seconds)

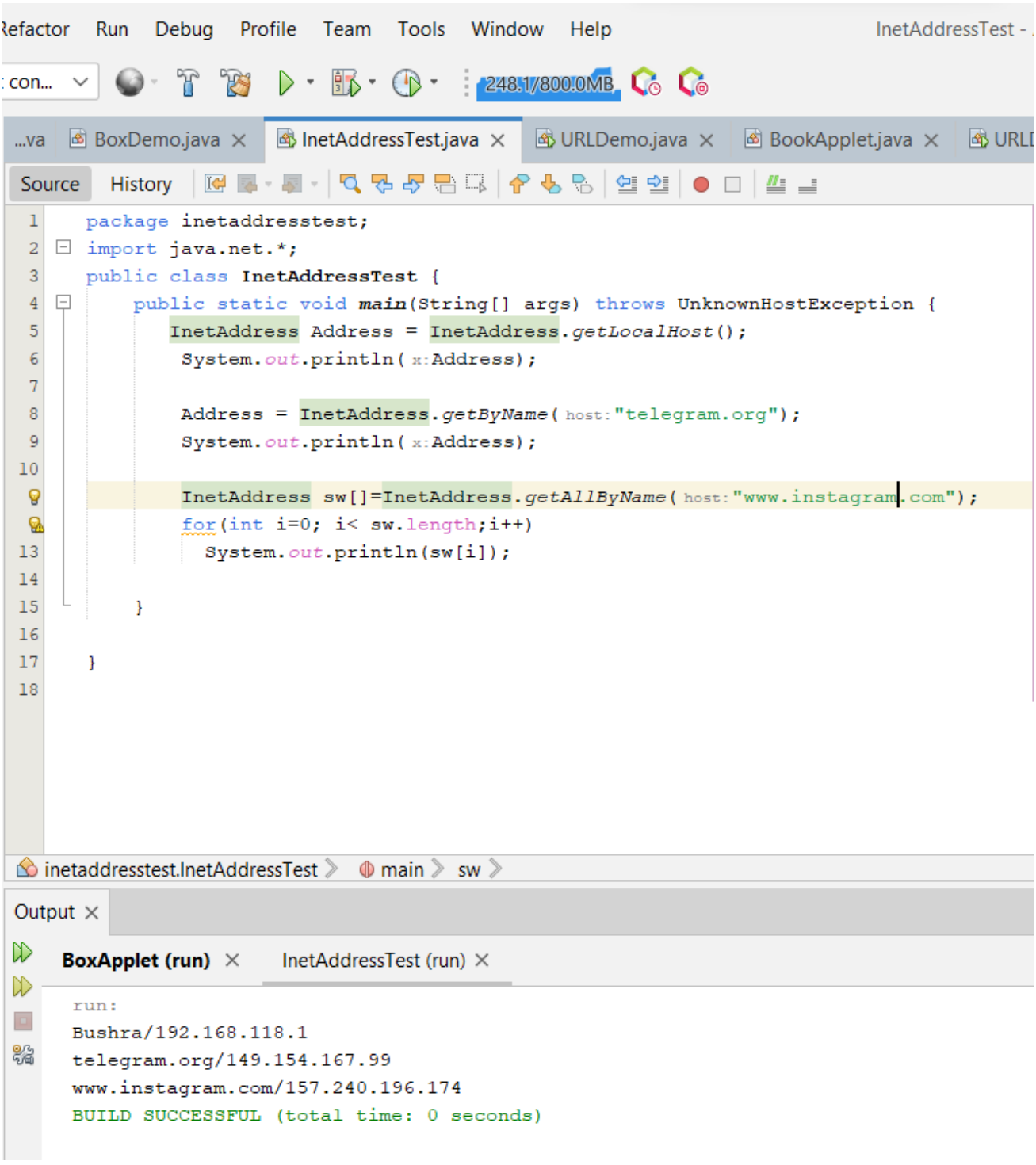
- برنامج بإستخدام ال applet لتعريف كلاس يسمى Box واستخدام دالة بناء واستخدام دالة تسمى volume



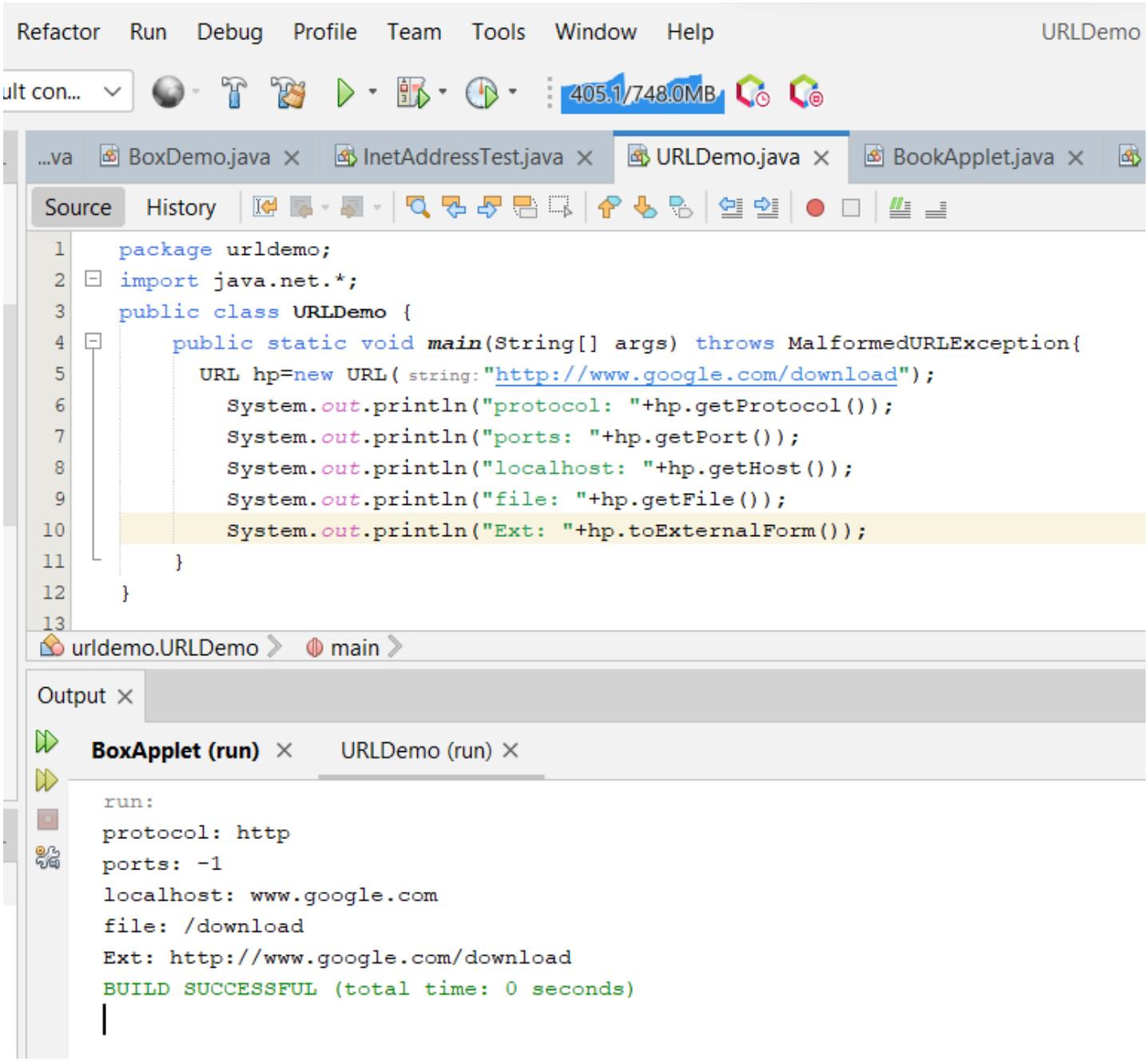
- برنامج يقوم بالتحكم في المسار الرئيسي main thread



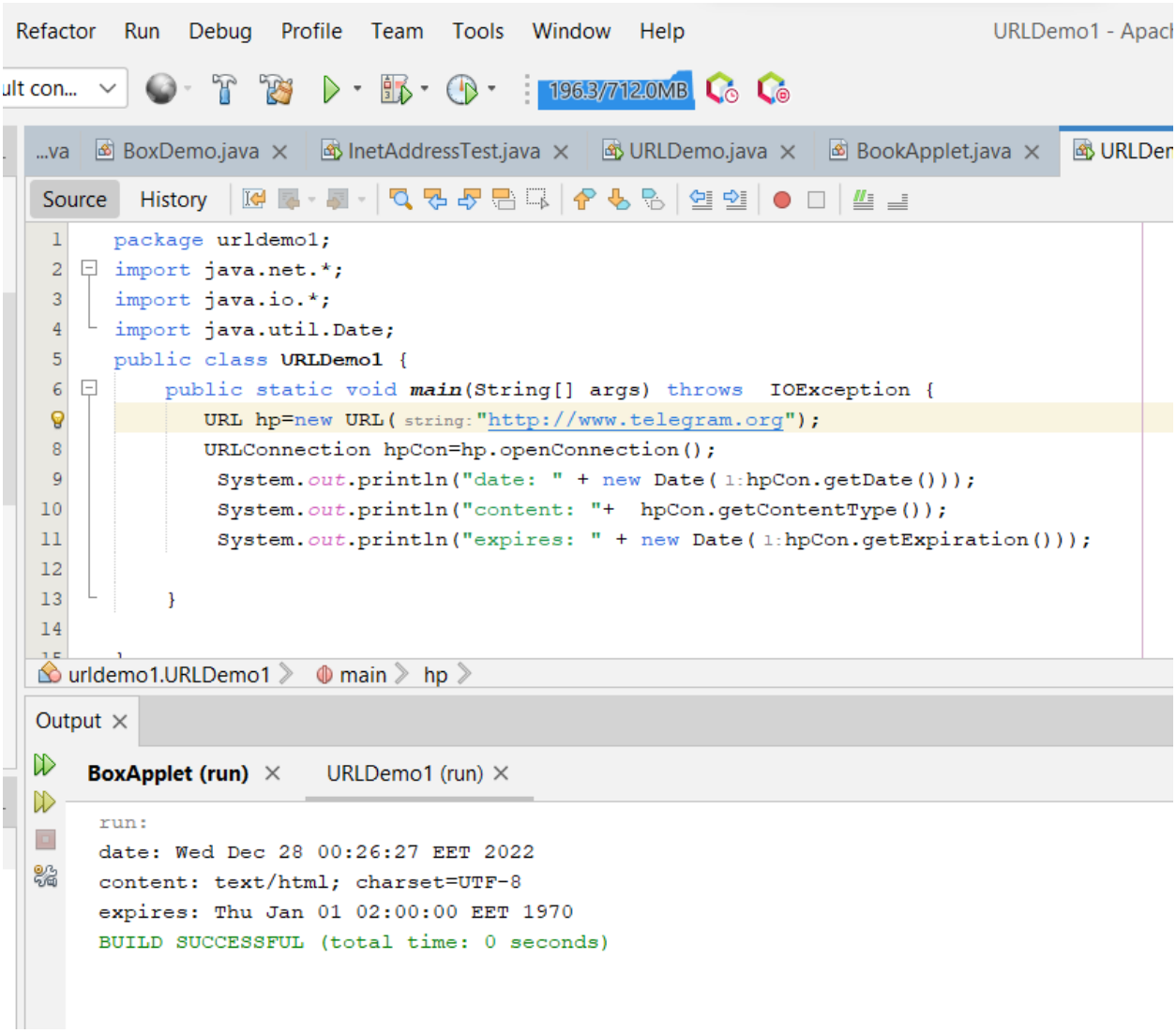
- برنامج يقوم بطباعة the address and names of the local machine لموقعين معروفين



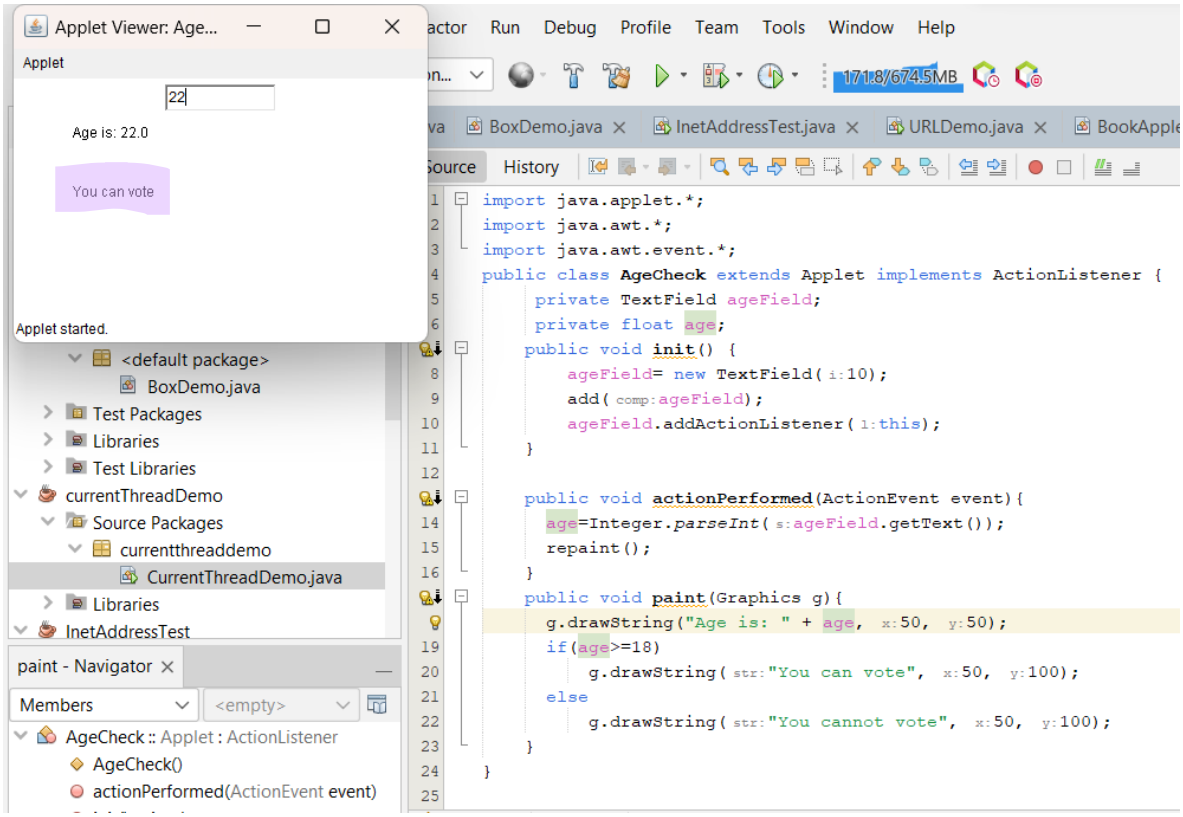
- دوال جاهزة في الURL

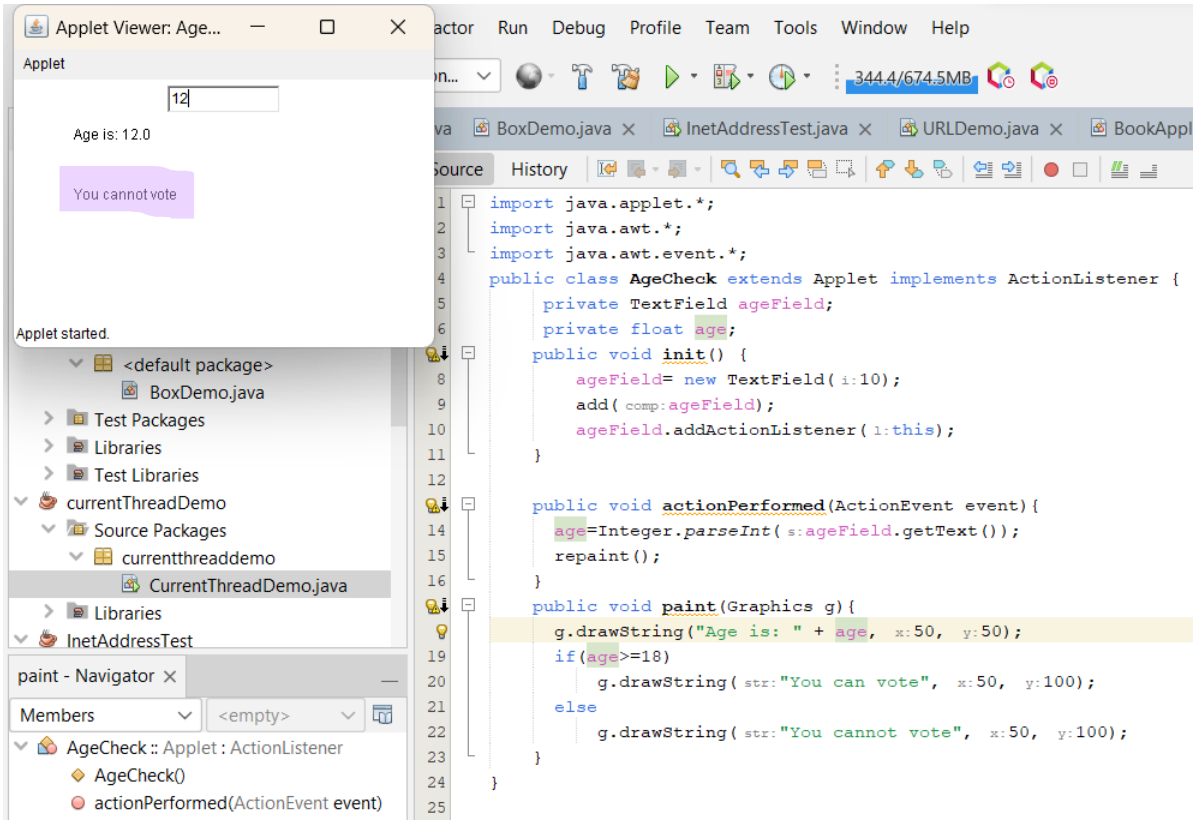


- برنامج بإستخدام ال URL Class والدالة openConnection()



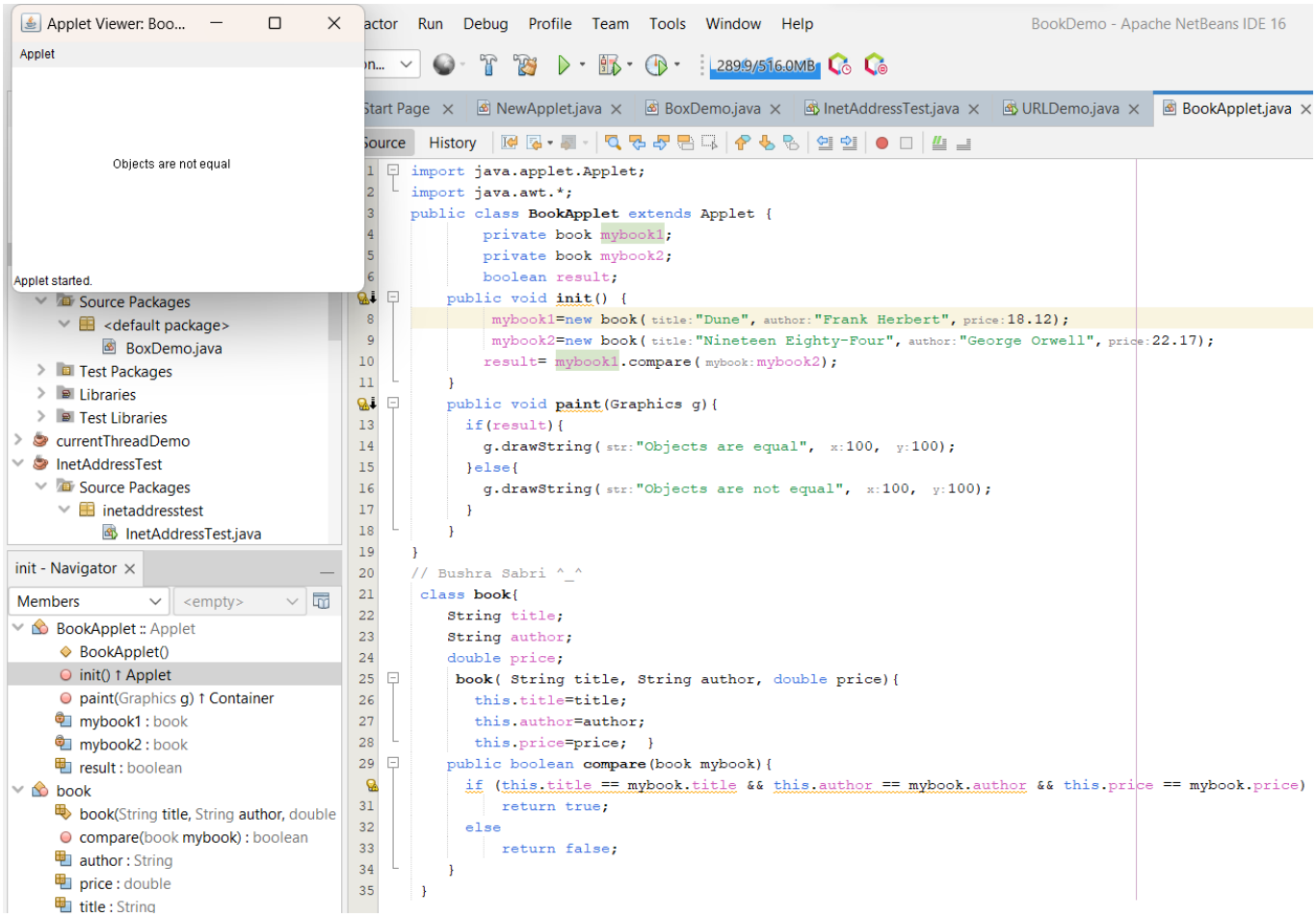
- برنامج يقوم بإدخال عمر الشخص بإستخدام ال applet



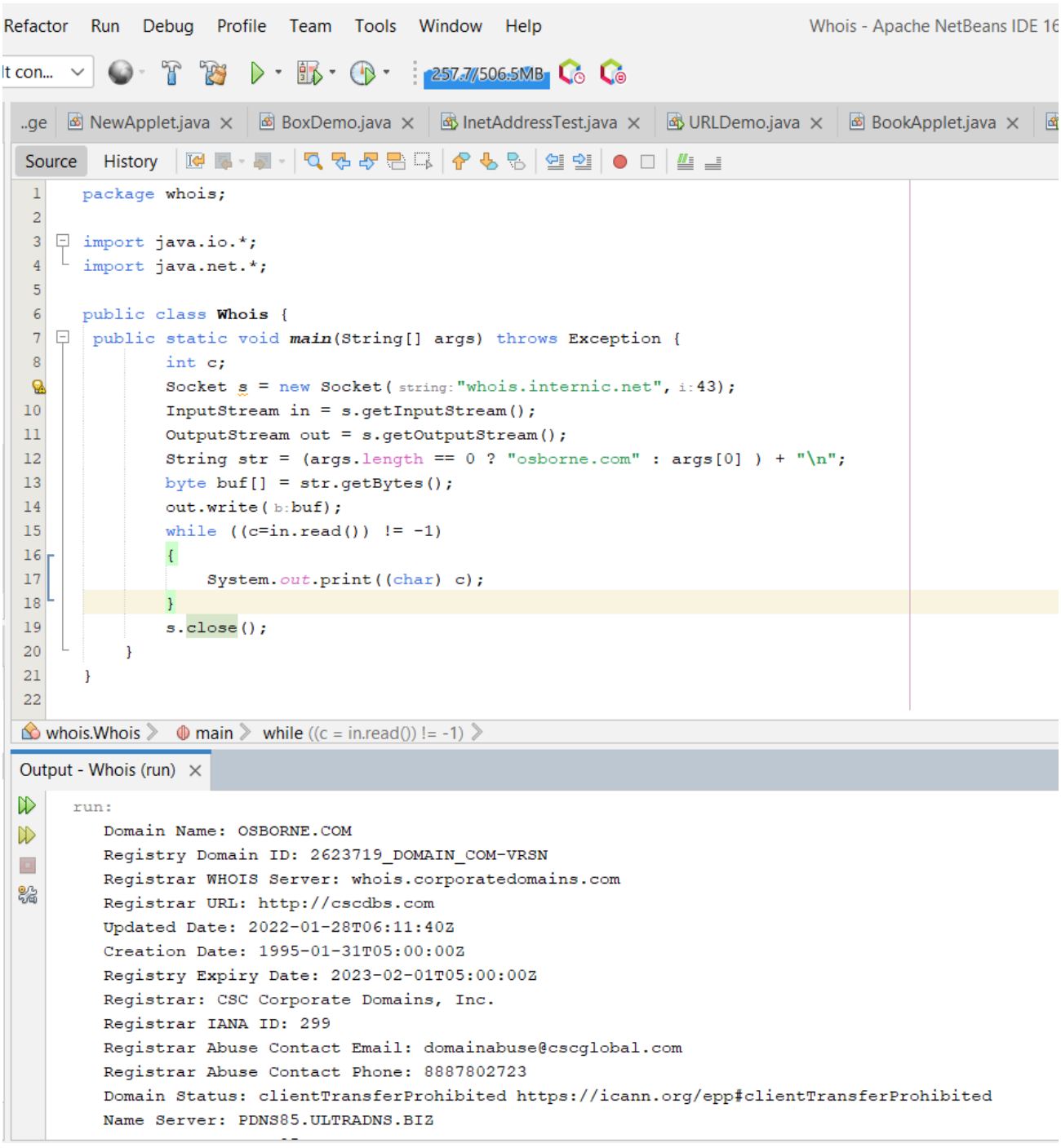


H.W

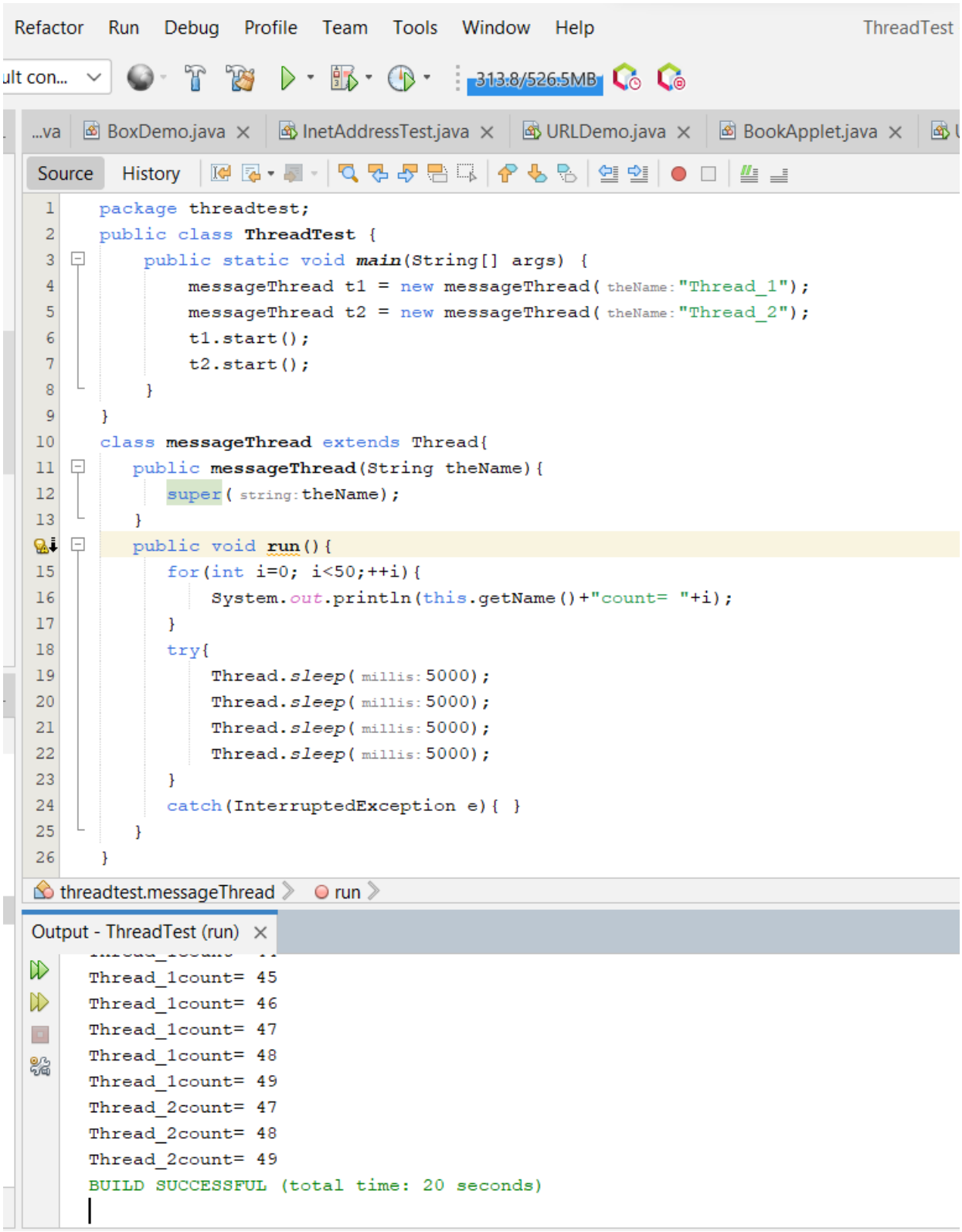
اكتب كلاس يسمى book يتكون من instance variables وهي title,author,price وإنشاء دالة بناء بإستخدام ال applet وكتابة دالة تقوم بالمقارنة بين ال object المُرسل وال object الحالي.



Demonstrate the socket

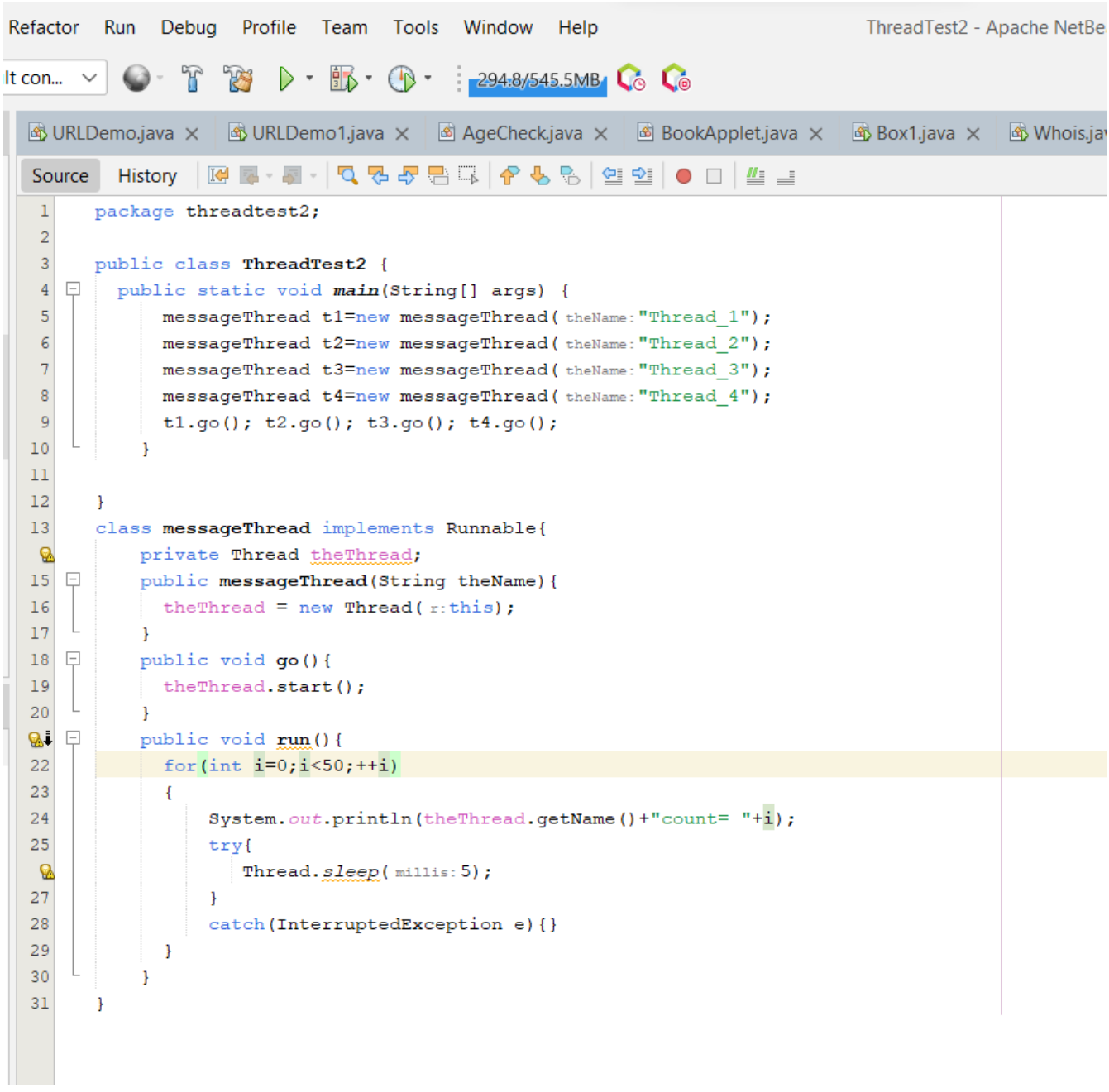


*اكتب البرنامج الآتـــي الذي يقوم بإستخدام المسارات المتعددة multiple threads



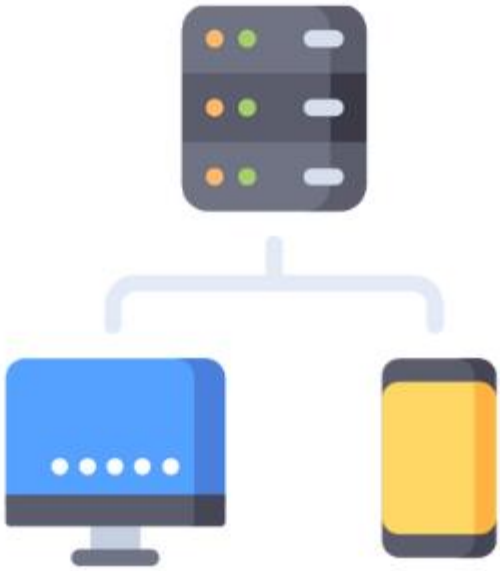
ال output للكود السابق عبارة عن two threads بيطبع كل واحد فيهم من الصفر لل49 (ترتيب عشوائي) -_-

* بإستخدام طريقة Runnable Interface

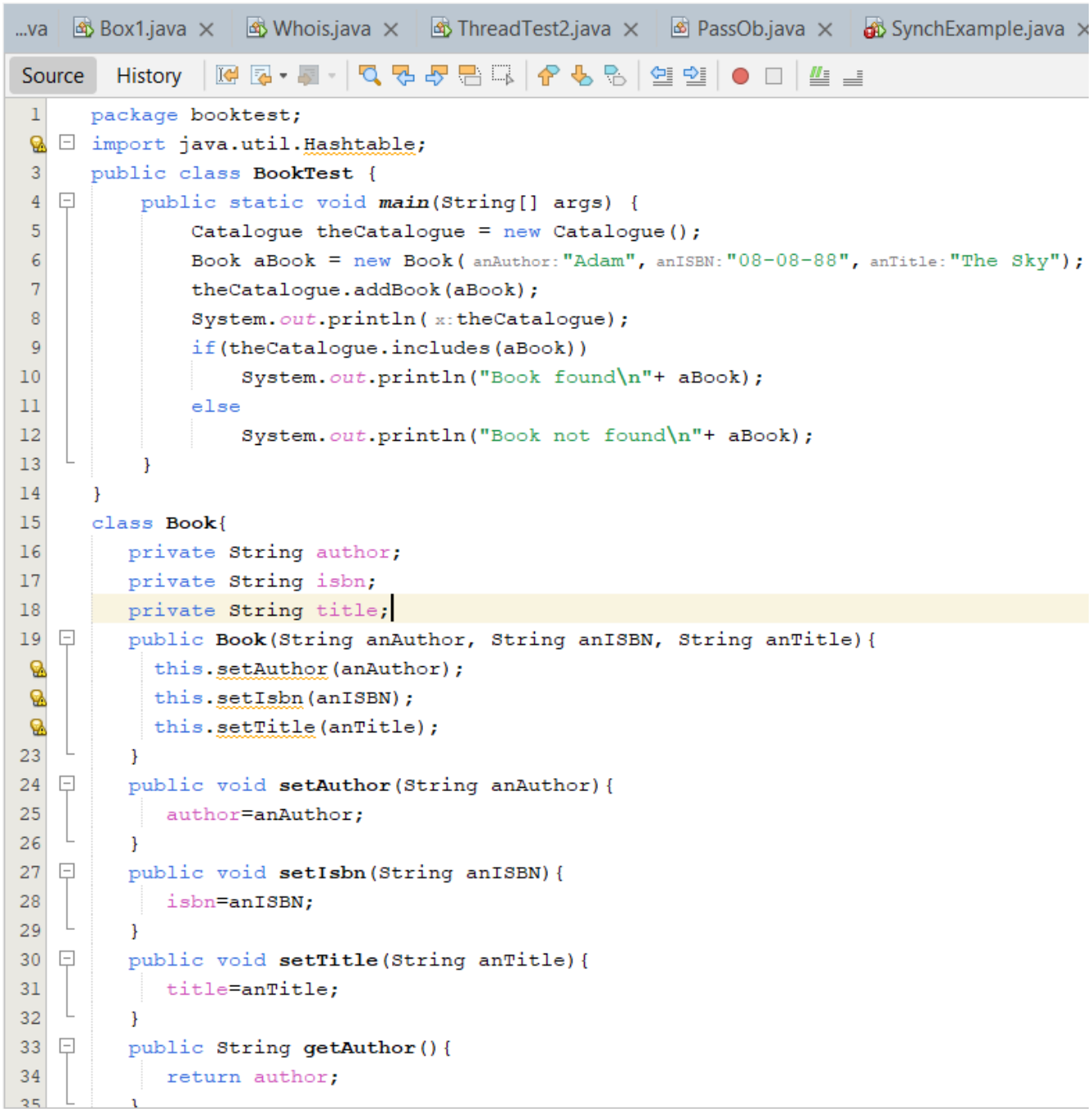


بعد النصفي

ITWT320



*اكتب برنامج بلغة الجافا لتعريف كلاس يسمى Book يتكون من العناصر التالية:اسم المؤلف، isbn، وtitle ودالة بناء ودوال استرجاع Getters وSetters ويتكون من كلاس يسمى Catalogue يتم من خلاله تعريف Hashtable ودالة تقوم بإضافة كتاب تسمى addBook ودالة تسمى include للبحث عن كتاب وكلاس يسمى BookTest وهذا هو الكلاس الرئيسي الذي يتم من خلاله تكوين ال objects وإستدعاء الدوال.



```
35      }
36      public String getISBN(){
37          return isbn;
38      }
39      public String getTitle(){
40          return title;
41      }
42  }
43  class Catalogue{
        private Hashtable theBookList;
45      public Catalogue(){
          theBookList = new Hashtable(); }
47      public void addBook(Book aBook){
48        theBookList.put( key:aBook.getISBN(), value:aBook); }
49      public boolean includes(Book aBook){
50        return theBookList.containsKey( key:aBook.getISBN()); }
51  }
```

*برنامج يتكون من multiple threads ويتم فيه إنهاء ال main thread أخيـرًا بإستخدام ال sleep()

```
1   package multithreaddemo;
2   public class MultiThreadDemo {
3       public static void main(String[] args) {
            new NewThread( threadName: "one"); //start the thread
            new NewThread( threadName: "two");
            new NewThread( threadName: "three");
7           try{ //wait for other threads to end
8             Thread.sleep( millis: 10000);
9           }catch(InterruptedException e){
10              System.out.println( x: "Main thread interrupted"); }
11              System.out.println( x: "Main thread exiting");   }
12  }
13  class NewThread implements Runnable{
14          String name; // name of thread
15          Thread t;
16          NewThread(String threadName){
17            name= threadName;
18            t = new Thread( task: this,name);
19              System.out.println("New Thread: "+t);
20              t.start(); //Start the thread
21          } //This is entry point for thread
22          public void run(){
23            try{
24              for(int i=5; i>0; i--){
25                  System.out.println(name +";"+i);
26                  Thread.sleep( millis: 1000);   }}
27            catch(InterruptedException e){
28                System.out.println(name +"Interrupted");}
29                System.out.println(name +"exiting");}
30  }
```

* برنامج يتكون من multiple threads ويتم فيه إنهاء ال main thread أخيـرًا بإستخدام الjoin()

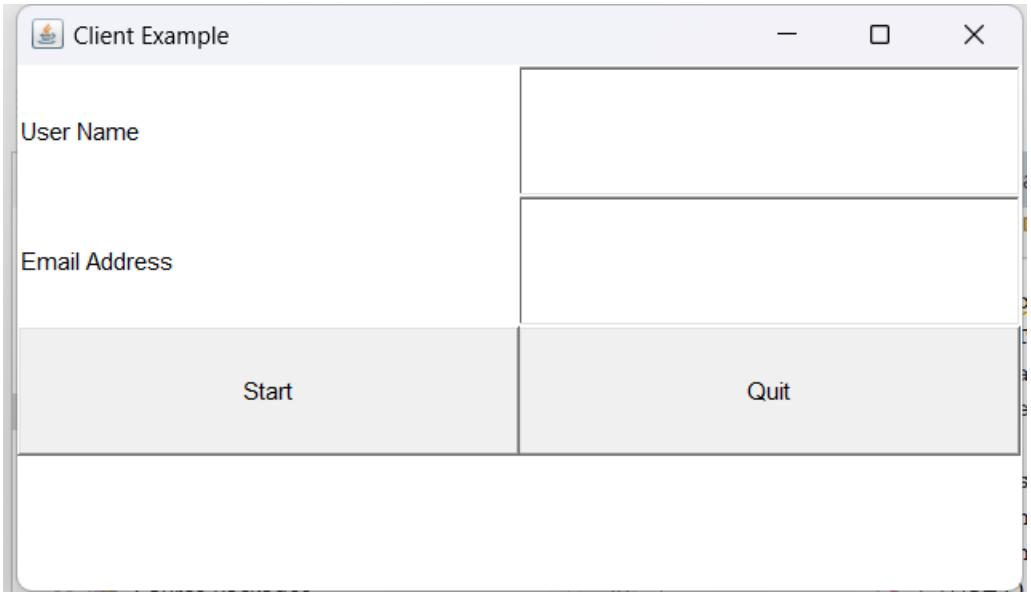
Using join() method to wait for threads to finish

```
1  package demojoin;
2  public class DemoJoin {
3      public static void main(String[] args) {
4          NewThread ob1 = new NewThread( threadName: "one");
5          NewThread ob2 = new NewThread( threadName: "two");
6          NewThread ob3 = new NewThread( threadName: "three");
7         System.out.println("Thread one is alive: "+ ob1.t.isAlive());
8          System.out.println("Thread two is alive: "+ ob2.t.isAlive());
9           System.out.println("Thread three is alive: "+ ob3.t.isAlive());
10          // wait to threads to finish
11          try{
12              System.out.println( x: "Waiting for threads to finish.");
13              ob1.t.join();
14              ob2.t.join();
15              ob3.t.join();
16          } catch(InterruptedException e){
17              System.out.println( x: "Main Thread Interrupted");
18          }
19             System.out.println("Thread one is alive: "+ ob1.t.isAlive());
20             System.out.println("Thread two is alive: "+ ob2.t.isAlive());
21             System.out.println("Thread three is alive: "+ ob3.t.isAlive());
22             System.out.println( x: "Main thread exiting.");
23      }
24 }
25 // using join() to wait for threads to finish
26 class NewThread implements Runnable{
27     String name; //name of thread
28     Thread t;
29     NewThread(String threadName){
30       name = threadName;
31       t = new Thread( task: this, name);
32          System.out.println("New thread: "+ t);
           t.start(); // Start the thread
34      }
35     // This is the entry point for thread
       public void run(){
37       try{
38           for(int i=5; i>0; i--){
39               System.out.println(name+ ":" + i);
                 Thread.sleep( millis: 1000);
41           }
42       } catch(InterruptedException e){
43          System.out.println(name+ "interrupted");
44       }
45          System.out.println(name+ "exiting");
46      }
47  }
48
```

Client-Server Model

```
1   package client;
2   import java.awt.*;
3   import java.awt.event.*;
4   import java.io.*;
5   import java.net.*;
6   public class Client extends Frame implements ActionListener {
7       TextField userName , emailAddress;
8       Button startProcessing , quit;
9       InputStream is = null;
10      OutputStream os = null;
11      PrintWriter pw = null;
12      BufferedReader br = null;
13      Socket s;
14      Client(String title){
15        super(title);
16        userName = new TextField( columns: 10);
17        emailAddress = new TextField( columns: 20);
18        startProcessing = new Button( label: "Start");
19        quit = new Button( label: "Quit");
20        setLayout(new GridLayout( rows: 4, cols: 1));
21        add(new Label( text: "User Name"));
22        add( comp: userName);
23        add(new Label( text: "Email Address"));
24        add( comp: emailAddress);
25        add( comp: startProcessing);
26        add( comp: quit);
27        setSize( width: 150, height: 300);
28        setVisible( b: true);
          startProcessing.addActionListener( l: this);
          quit.addActionListener( l: this);
31        try{
32          s = new Socket( host: "127.0.0.1", port: 2500);
33          is = s.getInputStream();
34          os = s.getOutputStream();
35          pw = new PrintWriter( out: os, autoFlush: true);
36          br = new BufferedReader(new InputStreamReader( in: is));
37        }
38        catch(IOException e){
39            System.out.println("Error Connection with the server "+e);
40        }
41      }
          public void actionPerformed(ActionEvent ae){
43        String buttonTest = ae.getActionCommand();
44        String typedName;
45        String receivedAddress;
```

```
46          try{
47              if(buttonTest.equals( anObject: quit)){
48                pw.println( x: "Exiting the server");
49                pw.println( x: "Exit");
50                is.close();
51                os.close();
52                pw.close();
53                br.close();
54                s.close();
55                System.exit( status: 0);//shut down
56              }
57              if(buttonTest.equals( anObject: "Start")){
58              typedName = userName.getText();
59              //send to server
60              pw.println( x: typedName);
61            //recived
62            receivedAddress=br.readLine();
63            emailAddress.setText( t: receivedAddress);}}
64             catch(IOException e){
65              System.out.println("Problem Connecting the server to send/recevied"+e);
66              }}
67            public static void main(String[] args){
68              Client c = new Client( title: "Client Example");
69            }}
```



*كـود الـserver

```
package server;
import java.io.*;
import java.net.*;
import java.util.*;
public class Server {
    public static void main(String[] args){
        InputStream is = null;
        OutputStream os = null;
        PrintWriter pw = null;
        BufferedReader br = null;

        int ConnectionCount = 0;
        String lineReader = "";
        Object o;
        String reply = "";
        System.out.println( x: "Server Starting");

        Hashtable names = new Hashtable();
        names.put( key: "Fred Smith",  value: "f.smith@br.uk");
        names.put( key: "Joe Bbgges",  value: "Joe.Bbgges@lei.uk");
        System.out.println( x: "Database done");

        try{
          ServerSocket ss = new ServerSocket( port: 2500);
          while(true){
           Socket s = ss.accept();
           ConnectionCount++;
           System.out.println("Connection "+ConnectionCount+" Once");

           is = s.getInputStream();
           os = s.getOutputStream();
           pw = new PrintWriter( out: os, autoFlush: true);
           br = new BufferedReader(new InputStreamReader( in: is));
           System.out.println( x: "System Setup");
           lineReader = "";

          while(true){
           lineReader=br.readLine();
           if (lineReader.equals( anObject: "Exit"))
            break;
              o=names.get( key: lineReader);
           if (o==null)
              reply = "User not Knows";
           else{
              reply = (String)o;
              pw.println( x: reply);
                }
               }
              pw.close();
              br.close();
              is.close();
              os.close();
              System.out.println( x: "Cloused Connection");
          }}
         catch(IOException e){
              System.out.println("Trouble "+e);
        }
    }
}
```