

The screenshot displays an IDE interface with the following components:

- Menu Bar:** refactor, Run, Debug, Profile, Team, Tools, Window, Help
- Toolbar:** Includes a dropdown menu with 'con...', a globe icon, a wrench icon, a play button, a refresh icon, a memory usage indicator showing '246.1/502.5MB', and two circular icons with red 'X' marks.
- Tab Bar:** Start Page x, NewApplet.java x, BoxDemo.java x, InetAddressTest.java x
- Source Editor:** Contains the following Java code:

```
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
```
- Navigation Bar:** box1.Box1 > volume >
- Output Console:** Output - Box1 (run) x. It shows the following output:

```
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

The screenshot displays an IDE with the following components:

- Applet Viewer:** A window titled "Applet Viewer: Box..." showing the output of the applet. It contains two lines of text: "volume of Box one is6.0" and "The new volume of Box is 6.0". Below the text, it says "Applet started." and shows a project tree with "Source Packages" and "box1" containing "Box1.java".
- Source Editor:** The main window showing the Java code for the applet and the Box class. The code is as follows:

```
1 import java.awt.*;
2 import java.applet.Applet;
3 public class BoxDemo extends Applet {
4     private Box myBox1;
5     double vol1;
6     public void init() {
7         myBox1 = new Box(w:1, h:2, d:3);
8     }
9     public void paint(Graphics g) {
10        vol1=myBox1.volume();
11        g.drawString("volume of Box one is" +vol1, x:100, y:100);
12        vol1=myBox1.volume();
13        g.drawString("The new volume of Box is "+vol1, x:150, y:150);
14    }
15    class Box{
16        double width,height,depth;
17        public Box(double w ,double h, double d){
18            width=w;
19            height=h;
20            depth=d;
21        }
22        public double volume() {
23            double vol;
24            vol=width*height*depth;
25            return vol;
26        }
27    }
28 }
29
30
31
```
- Box - Navigator:** A window showing the class hierarchy and members. It lists "BoxDemo :: Applet" with members "BoxDemo()", "init() ↑ Applet", "paint(Graphics g) ↑ Container", "myBox1 : Box", and "vol1 : double". It also lists "Box" with members "Box(double w, double h, double d)" and "volume() : double".

برنامج يقوم بالتحكم في المسار الرئيسي main thread

The screenshot shows an IDE window with the following content:

Menu: Refactor Run Debug Profile Team Tools Window Help

currentThreadDemo

479.1/824.5MB

Source History

```
1 package currentthreaddemo;
2
3 public class CurrentThreadDemo {
4
5     public static void main(String[] args) {
6         Thread t = Thread.currentThread();
7         System.out.println("Current thread: "+t);
8         t.setName( name:"My Thread");
9         System.out.println("After change name: "+t);
10        try{
11            for(int n=4;n>0;n--){
12                Thread.sleep( millis:1000);
13            }
14        }
15        catch(InterruptedException e){
16            System.out.println( x:"Main Thread interrupted");
17        }
18    }
19 }
20
```

currentthreaddemo.CurrentThreadDemo > main > try > catch InterruptedException e >

Output ×

BoxApplet (run) × currentThreadDemo (run) ×

```
run:
Current thread: Thread[main,5,main]
After change name: Thread[My Thread,5,main]
BUILD SUCCESSFUL (total time: 4 seconds)
```

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

The screenshot shows an IDE window with the following code in `InetAddressTest.java`:

```

1 package inetaddresstest;
2 import java.net.*;
3 public class InetAddressTest {
4     public static void main(String[] args) throws UnknownHostException {
5         InetAddress Address = InetAddress.getLocalHost();
6         System.out.println("x:Address");
7
8         Address = InetAddress.getByName(host:"telegram.org");
9         System.out.println("x:Address");
10
11         InetAddress sw[]=InetAddress.getAllByName(host:"www.instagram.com");
12         for(int i=0; i< sw.length;i++)
13             System.out.println(sw[i]);
14
15     }
16
17 }
18

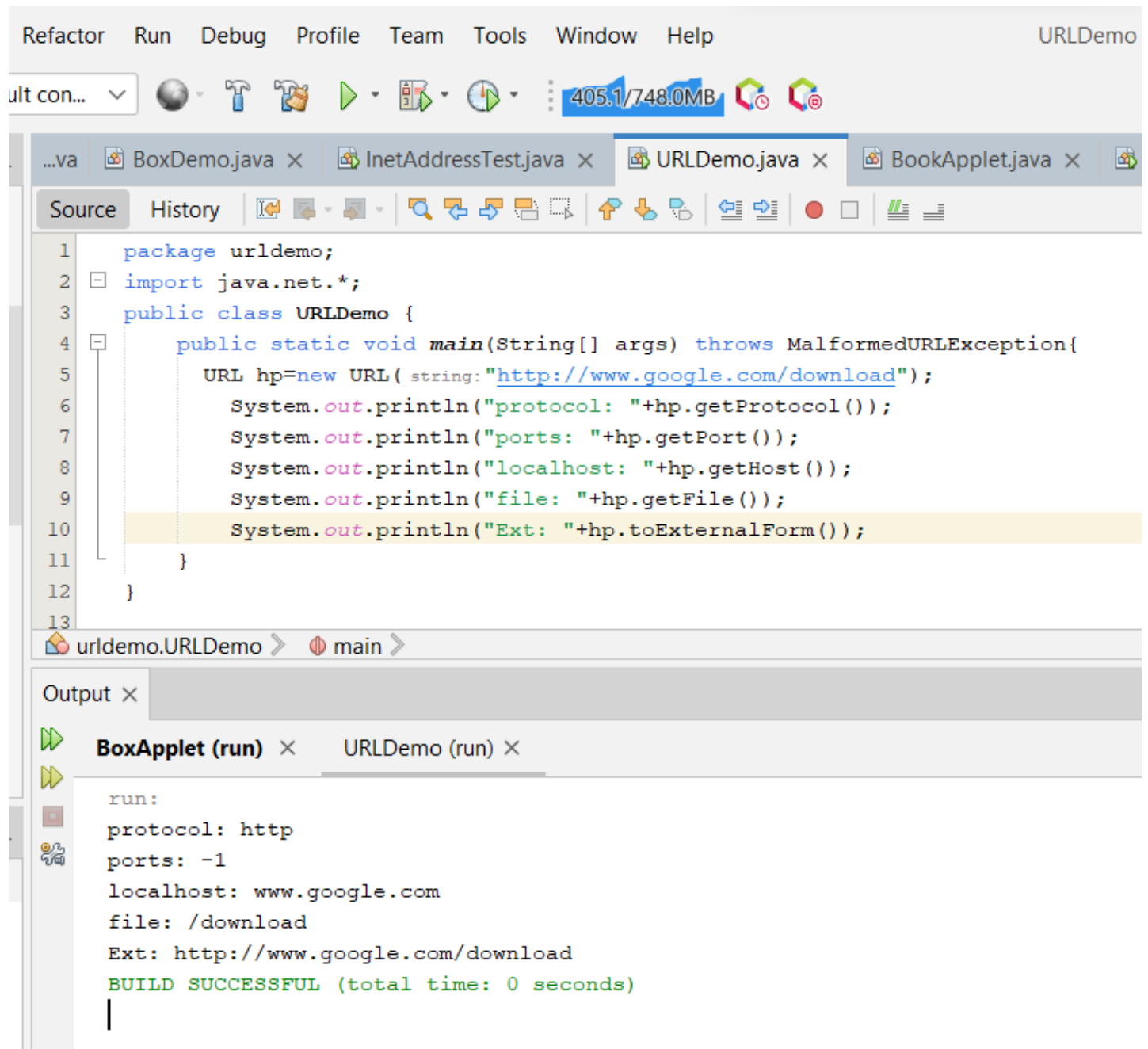
```

The output window shows the following results:

```

run:
Bushra/192.168.118.1
telegram.org/149.154.167.99
www.instagram.com/157.240.196.174
BUILD SUCCESSFUL (total time: 0 seconds)

```



The screenshot shows an IDE window titled "URLDemo" with a menu bar (Refactor, Run, Debug, Profile, Team, Tools, Window, Help) and a toolbar. The main editor displays the source code for `URLDemo.java`:

```
1 package urldemo;
2 import java.net.*;
3 public class URLDemo {
4     public static void main(String[] args) throws MalformedURLException{
5         URL hp=new URL( string: "http://www.google.com/download");
6         System.out.println("protocol: "+hp.getProtocol());
7         System.out.println("ports: "+hp.getPort());
8         System.out.println("localhost: "+hp.getHost());
9         System.out.println("file: "+hp.getFile());
10        System.out.println("Ext: "+hp.toExternalForm());
11    }
12 }
13
```

The IDE also shows the execution context: `urldemo.URLDemo > main >`. The Output window displays the following output:

```
run:
protocol: http
ports: -1
localhost: www.google.com
file: /download
Ext: http://www.google.com/download
BUILD SUCCESSFUL (total time: 0 seconds)
```

برنامج بإستخدام ال URL Class والدالة openConnection()

The screenshot displays an IDE window titled "URLDemo1 - Apache". The source code editor shows the following Java code:

```
1 package urldemo1;
2 import java.net.*;
3 import java.io.*;
4 import java.util.Date;
5 public class URLDemo1 {
6     public static void main(String[] args) throws IOException {
7         URL hp=new URL( string: "http://www.telegram.org");
8         URLConnection hpCon=hp.openConnection();
9         System.out.println("date: " + new Date( 1:hpCon.getDate()));
10        System.out.println("content: "+ hpCon.getContentType());
11        System.out.println("expires: " + new Date( 1:hpCon.getExpiration()));
12    }
13 }
14
```

The IDE's breadcrumb navigation shows the path: `urldemo1.URLDemo1 > main > hp >`. The Output window shows the following execution results:

```
run:
date: Wed Dec 28 00:26:27 EET 2022
content: text/html; charset=UTF-8
expires: Thu Jan 01 02:00:00 EET 1970
BUILD SUCCESSFUL (total time: 0 seconds)
```

برنامج يقوم بإدخال عمر الشخص باستخدام ال applet

The screenshot shows an IDE with an applet viewer window titled 'Applet Viewer: Age...' and a source code editor. The applet viewer displays a text field containing '22', the text 'Age is: 22.0', and a purple button labeled 'You can vote'. The source code editor shows the following code:

```

1 import java.applet.*;
2 import java.awt.*;
3 import java.awt.event.*;
4 public class AgeCheck extends Applet implements ActionListener {
5     private TextField ageField;
6     private float age;
7     public void init() {
8         ageField= new TextField(10);
9         add( ageField);
10        ageField.addActionListener( this);
11    }
12
13    public void actionPerformed(ActionEvent event){
14        age=Integer.parseInt( ageField.getText());
15        repaint();
16    }
17
18    public void paint(Graphics g){
19        g.drawString("Age is: " + age, 50, 50);
20        if(age>=18)
21            g.drawString( str:"You can vote", 50, 100);
22        else
23            g.drawString( str:"You cannot vote", 50, 100);
24    }
25 }

```

The screenshot shows the same IDE environment as above, but with the applet viewer window displaying an input field containing '12', the text 'Age is: 12.0', and a purple button labeled 'You cannot vote'. The source code editor shows the same code as in the previous screenshot, but with the 'else' branch highlighted in yellow, indicating it is the active state.

H.W

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

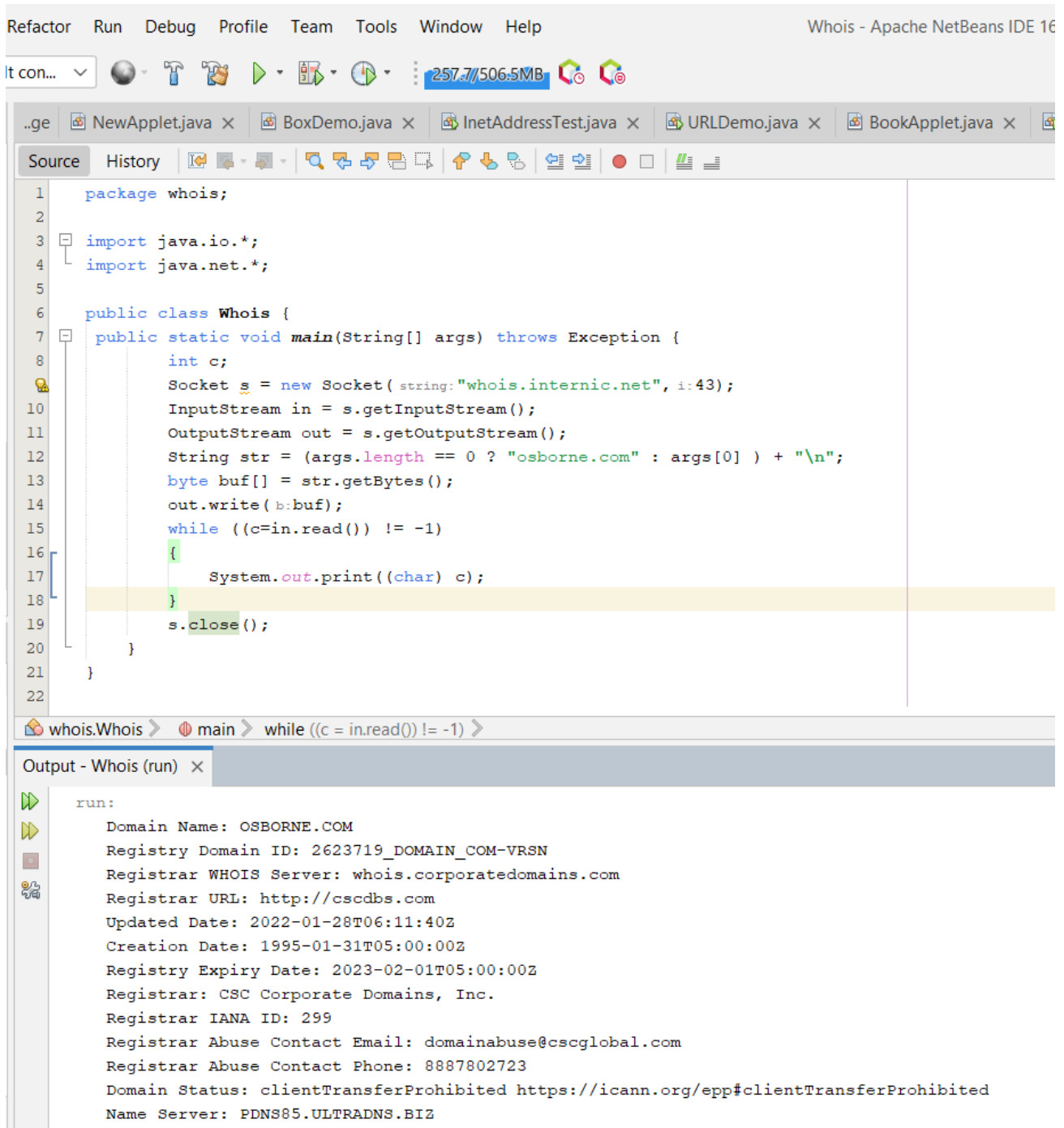
The screenshot displays the Apache NetBeans IDE environment. The main editor shows the source code for `BookApplet.java` and `book.java`. The `BookApplet` class extends `Applet` and implements the `init()` and `paint()` methods. In `init()`, two `book` objects are created: `mybook1` (title: "Dune", author: "Frank Herbert", price: 18.12) and `mybook2` (title: "Nineteen Eighty-Four", author: "George Orwell", price: 22.17). The `compare` method of `mybook1` is called, and the result is stored in `result`. The `paint` method checks the value of `result` and draws the text "Objects are equal" or "Objects are not equal" on the applet's canvas.

The variable navigator on the left shows the state of the applet after execution. It lists the `BookApplet` object and its methods (`init()`, `paint()`). It also shows the state of the `book` class, including the `mybook1` and `mybook2` instances, and the `result` variable, which is currently `boolean`.

```

1  import java.applet.Applet;
2  import java.awt.*;
3  public class BookApplet extends Applet {
4      private book mybook1;
5      private book mybook2;
6      boolean result;
7      public void init() {
8          mybook1=new book (title:"Dune", author:"Frank Herbert", price:18.12);
9          mybook2=new book (title:"Nineteen Eighty-Four", author:"George Orwell", price:22.17);
10         result= mybook1.compare ( mybook:mybook2);
11     }
12     public void paint(Graphics g){
13         if(result){
14             g.drawString( str:"Objects are equal", x:100, y:100);
15         }else{
16             g.drawString( str:"Objects are not equal", x:100, y:100);
17         }
18     }
19 }
20 // Bushra Sabri ^_^
21 class book{
22     String title;
23     String author;
24     double price;
25     book( String title, String author, double price){
26         this.title=title;
27         this.author=author;
28         this.price=price; }
29     public boolean compare(book mybook){
30         if (this.title == mybook.title && this.author == mybook.author && this.price == mybook.price)
31             return true;
32         else
33             return false;
34     }
35 }

```


Demonstrate the socket

The screenshot displays the Apache NetBeans IDE interface. The main window shows the source code for a Java class named `Whois`. The code uses a `Socket` to connect to `whois.internic.net` on port 43. It reads the response from the socket and prints it to the console. The code is as follows:

```
1 package whois;
2
3 import java.io.*;
4 import java.net.*;
5
6 public class Whois {
7     public static void main(String[] args) throws Exception {
8         int c;
9         Socket s = new Socket("whois.internic.net", 43);
10        InputStream in = s.getInputStream();
11        OutputStream out = s.getOutputStream();
12        String str = (args.length == 0 ? "osborne.com" : args[0]) + "\n";
13        byte buf[] = str.getBytes();
14        out.write(buf);
15        while ((c=in.read()) != -1)
16        {
17            System.out.print((char) c);
18        }
19        s.close();
20    }
21 }
22
```

The 'Output' window shows the results of running the program:

```
run:
Domain Name: OSBORNE.COM
Registry Domain ID: 2623719_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.corporatedomains.com
Registrar URL: http://cscdns.com
Updated Date: 2022-01-28T06:11:40Z
Creation Date: 1995-01-31T05:00:00Z
Registry Expiry Date: 2023-02-01T05:00:00Z
Registrar: CSC Corporate Domains, Inc.
Registrar IANA ID: 299
Registrar Abuse Contact Email: domainabuse@cscglobal.com
Registrar Abuse Contact Phone: 8887802723
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Name Server: PDNS85.ULTRADNS.BIZ
```

*اكتب البرنامج الآتي يقوم بإستخدام المسارات المتعددة multiple threads

The screenshot shows an IDE window titled 'ThreadTest'. The code defines a package 'threadtest' containing a public class 'ThreadTest' and a class 'messageThread' that extends 'Thread'. The 'main' method in 'ThreadTest' creates two 'messageThread' objects, 't1' and 't2', and starts them. The 'run' method in 'messageThread' prints the thread's name and a count from 0 to 49, with a 5000ms sleep between each iteration. The output window shows the execution results for both threads.

```

1 package threadtest;
2 public class ThreadTest {
3     public static void main(String[] args) {
4         messageThread t1 = new messageThread( theName: "Thread_1");
5         messageThread t2 = new messageThread( theName: "Thread_2");
6         t1.start();
7         t2.start();
8     }
9 }
10 class messageThread extends Thread{
11     public messageThread(String theName){
12         super( string:theName);
13     }
14     public void run(){
15         for(int i=0; i<50;++i){
16             System.out.println(this.getName()+"count= "+i);
17         }
18         try{
19             Thread.sleep( millis: 5000);
20             Thread.sleep( millis: 5000);
21             Thread.sleep( millis: 5000);
22             Thread.sleep( millis: 5000);
23         }
24         catch(InterruptedException e){ }
25     }
26 }

```

threadtest.messageThread > run >

Output - ThreadTest (run) x

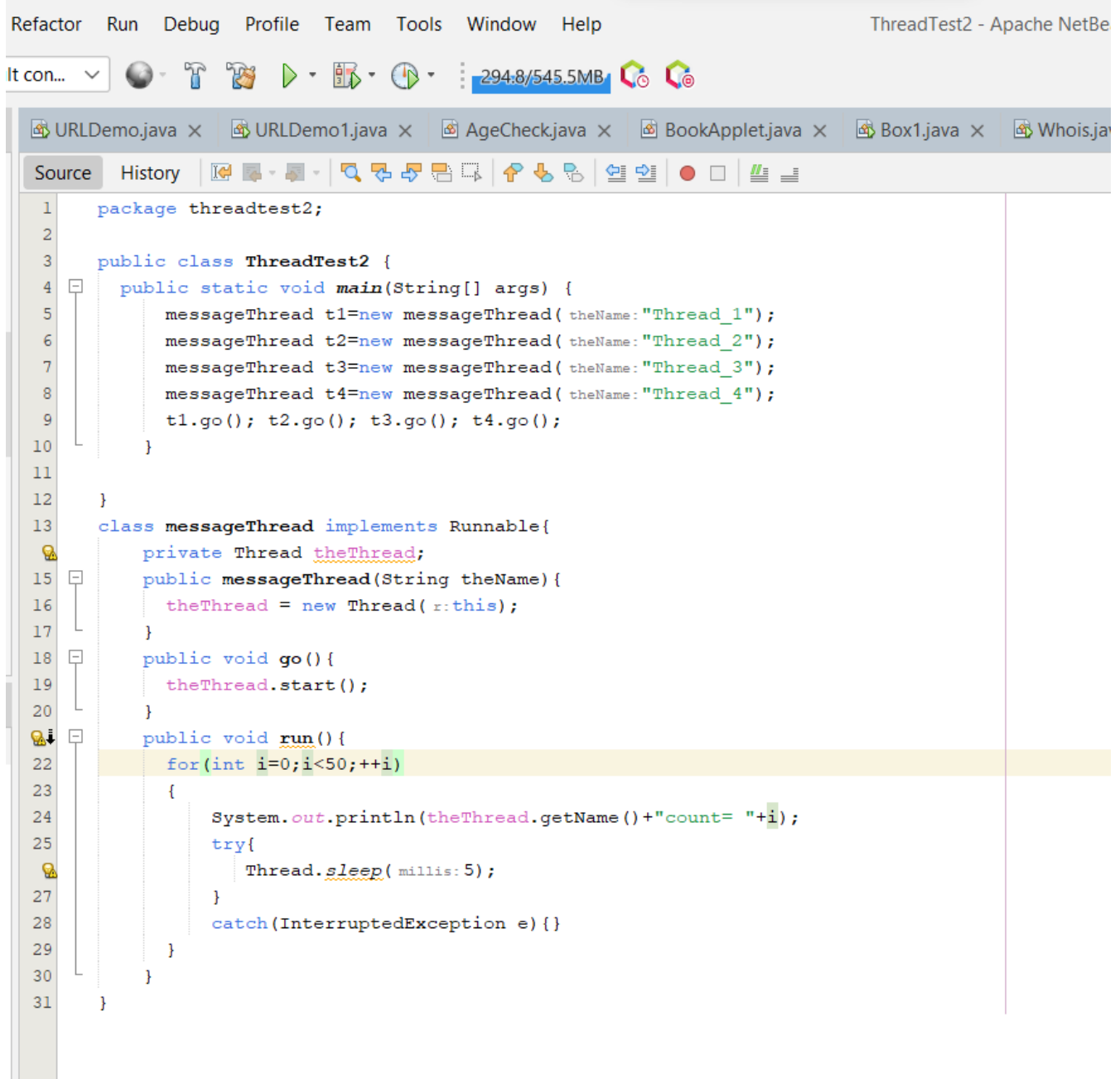
```

Thread_1count= 45
Thread_1count= 46
Thread_1count= 47
Thread_1count= 48
Thread_1count= 49
Thread_2count= 47
Thread_2count= 48
Thread_2count= 49
BUILD SUCCESSFUL (total time: 20 seconds)

```

```
Output - ThreadTest (run) ×
run:
Thread_1count= 0
Thread_1count= 1
Thread_1count= 2
Thread_1count= 3
Thread_1count= 4
Thread_1count= 5
Thread_1count= 6
Thread_1count= 7
Thread_1count= 8
Thread_1count= 9
Thread_1count= 10
Thread_1count= 11
Thread_1count= 12
Thread_1count= 13
Thread_1count= 14
Thread_1count= 15
Thread_1count= 16
Thread_1count= 17
Thread_2count= 0
Thread_2count= 1
Thread_2count= 2
Thread_2count= 3
Thread_2count= 4
Thread_2count= 5
Thread_2count= 6
Thread_2count= 7
Thread_2count= 8
Thread_2count= 9
Thread_2count= 10
Thread_2count= 11
Thread_2count= 12
Thread_1count= 18
Thread_1count= 19
Thread_1count= 20
Thread_1count= 21
Thread_1count= 22
```

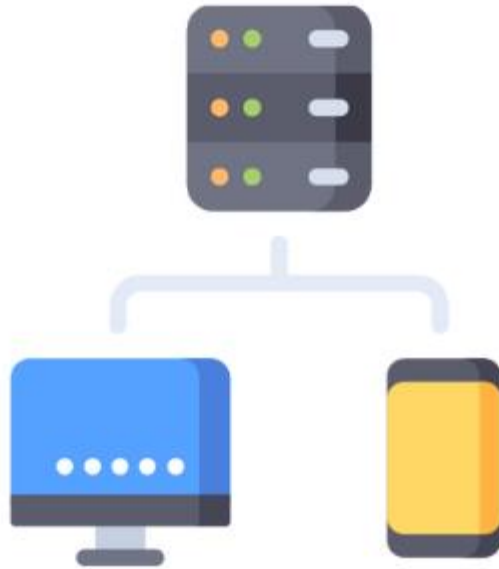
ال output للكود السابق عبارة عن two threads بيطلع كل واحد فيهم من الصفر لل 49 (ترتيب عشوائي) -_-



```
1 package threadtest2;
2
3 public class ThreadTest2 {
4     public static void main(String[] args) {
5         messageThread t1=new messageThread( theName: "Thread_1");
6         messageThread t2=new messageThread( theName: "Thread_2");
7         messageThread t3=new messageThread( theName: "Thread_3");
8         messageThread t4=new messageThread( theName: "Thread_4");
9         t1.go(); t2.go(); t3.go(); t4.go();
10    }
11
12 }
13 class messageThread implements Runnable{
14     private Thread theThread;
15     public messageThread(String theName){
16         theThread = new Thread( r::this);
17     }
18     public void go(){
19         theThread.start();
20     }
21     public void run(){
22         for(int i=0;i<50;++i)
23         {
24             System.out.println(theThread.getName()+"count= "+i);
25             try{
26                 Thread.sleep( millis: 5);
27             }
28             catch(InterruptedException e){}
29         }
30     }
31 }
```

بعد النصف

ITWT320



*اكتب برنامج بلغة الجافا لتعريف كلاس يسمى Book يتكون من العناصر التالية:اسم المؤلف، isbn، title 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.

```

...va  Box1.java X  Whois.java X  ThreadTest2.java X  PassOb.java X  SynchExample.java X
Source  History  [Icons]
1  package booktest;
2  import java.util.Hashtable;
3  public class BookTest {
4      public static void main(String[] args) {
5          Catalogue theCatalogue = new Catalogue();
6          Book aBook = new Book( anAuthor:"Adam", anISBN:"08-08-88", anTitle:"The Sky");
7          theCatalogue.addBook(aBook);
8          System.out.println( x:theCatalogue);
9          if(theCatalogue.includes(aBook))
10             System.out.println("Book found\n"+ aBook);
11         else
12             System.out.println("Book not found\n"+ aBook);
13     }
14 }
15 class Book{
16     private String author;
17     private String isbn;
18     private String title;
19     public Book(String anAuthor, String anISBN, String anTitle){
20         this.setAuthor(anAuthor);
21         this.setIsbn(anISBN);
22         this.setTitle(anTitle);
23     }
24     public void setAuthor(String anAuthor){
25         author=anAuthor;
26     }
27     public void setIsbn(String anISBN) {
28         isbn=anISBN;
29     }
30     public void setTitle(String anTitle){
31         title=anTitle;
32     }
33     public String getAuthor(){
34         return author;
35     }

```

```
35     }
36     public String getISBN(){
37         return isbn;
38     }
39     public String getTitle(){
40         return title;
41     }
42 }
43 class Catalogue{
44     private Hashtable theBookList;
45     public Catalogue(){
46         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 أخيرًا باستخدام ال sleep()

```

1  package multithreddemo;
2  public class MultiThreadDemo {
3      public static void main(String[] args) {
4          new NewThread( threadName: "one"); //start the thread
5          new NewThread( threadName: "two");
6          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 أخيرًا باستخدام ال join()

****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);
33         t.start(); // Start the thread
34     }
35     // This is the entry point for thread
36     public void run(){
37         try{
38             for(int i=5; i>0; i--){
39                 System.out.println(name+ ":" + i);
40                 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

```

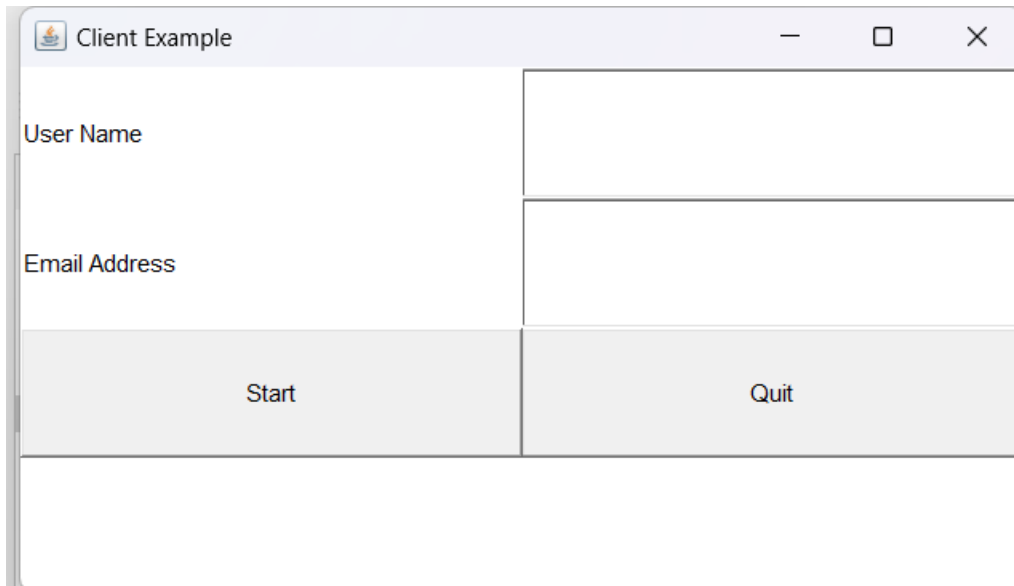
```

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);
29         startProcessing.addActionListener( l: this);
30         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 }
42 public void actionPerformed(ActionEvent ae){
43     String buttonTest = ae.getActionCommand();
44     String typedName;
45     String receivedAddress;

```

Client-Server programming (ITWT320)

```
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             //received
62             receivedAddress=br.readLine();
63             emailAddress.setText( t: receivedAddress);}}
64         catch(IOException e){
65             System.out.println("Problem Connecting the server to send/receivied"+e);
66         }}
67     public static void main(String[] args){
68         Client c = new Client( title: "Client Example");
69     }}
```



The screenshot shows a Java Swing window titled "Client Example". The window has a standard title bar with minimize, maximize, and close buttons. The main content area is divided into two columns. The left column contains two text input fields: the top one is labeled "User Name" and the bottom one is labeled "Email Address". The right column is empty. Below the input fields, there are two buttons: "Start" on the left and "Quit" on the right. The buttons are light gray with black text.

```
1 package server;
2 import java.io.*;
3 import java.net.*;
4 import java.util.*;
5 public class Server {
6     public static void main(String[] args){
7         InputStream is = null;
8         OutputStream os = null;
9         PrintWriter pw = null;
10        BufferedReader br = null;
11
12        int ConnectionCount = 0;
13        String lineReader = "";
14        Object o;
15        String reply = "";
16        System.out.println(x: "Server Starting");
17
18        Hashtable names = new Hashtable();
19        names.put(key: "Fred Smith", value: "f.smith@br.uk");
20        names.put(key: "Joe Bbgges", value: "Joe.Bbgges@lei.uk");
21        System.out.println(x: "Database done");
22
23        try{
24            ServerSocket ss = new ServerSocket(port: 2500);
25            while(true){
26                Socket s = ss.accept();
27                ConnectionCount++;
28                System.out.println("Connection "+ConnectionCount+" Once");
29
30                is = s.getInputStream();
31                os = s.getOutputStream();
32                pw = new PrintWriter(out: os, autoFlush: true);
33                br = new BufferedReader(new InputStreamReader(in: is));
34                System.out.println(x: "System Setup");
35                lineReader = "";
36
37                while(true){
38                    lineReader=br.readLine();
39                    if (lineReader.equals(anObject: "Exit"))
40                        break;
41                    o=names.get(key: lineReader);
42                    if (o==null)
43                        reply = "User not Knows";
44                    else{
45                        reply = (String)o;
46                        pw.println(x: reply);
47                    }
48                }
49                pw.close();
50                br.close();
51                is.close();
52                os.close();
53                System.out.println(x: "Cloused Connection");
54            }}
55        catch(IOException e){
56            System.out.println("Trouble "+e);
57        }
58    }
59 }
```

