

Internet Engineering

Tomasz Babczyński, Zofia Kruczkiewicz
Tomasz Kubik

Information systems modelling– UML and service description languages

Laboratory 3

Choose yourself and new technologies



Wrocław University of Technology

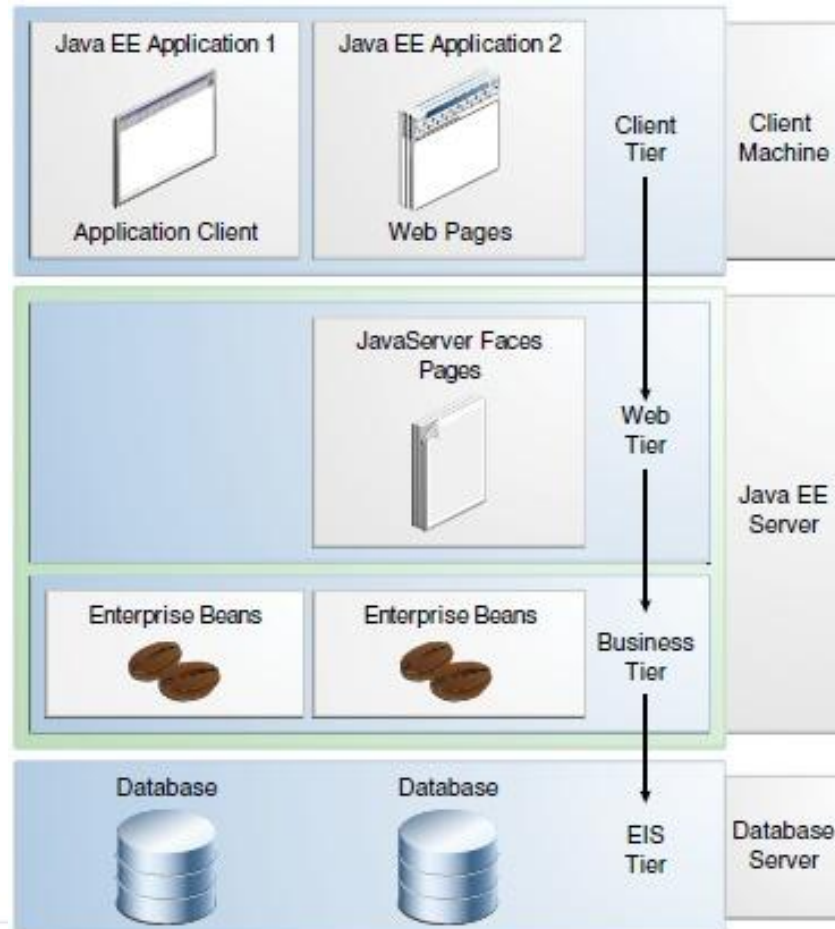


Project co-financed from the EU European Social Fund



Exemplary multitiered Information System

(Java EE 7 – Tutorial Java EE 7)





Steps for building the client tier of a multitier application

1. Create the Java EE project - right click File/New Project
2. Creation the library project for interface of EJB
3. Adding the EJB as the facade
4. Creation the library project for code from lab1-2
5. Adding the library project (p.4) to the EJB module of EE project and its interface
6. Creation of EE client tier
7. Build, deploy and run the created software
8. Some information about updating Java EE projects

Add your own results from the lab2 to this application in the 2, 6 steps and develop some changes in the Java Class Library project made in the 4 step (the Library1 project).

Example (Java EE & Java Web Learning Trail)

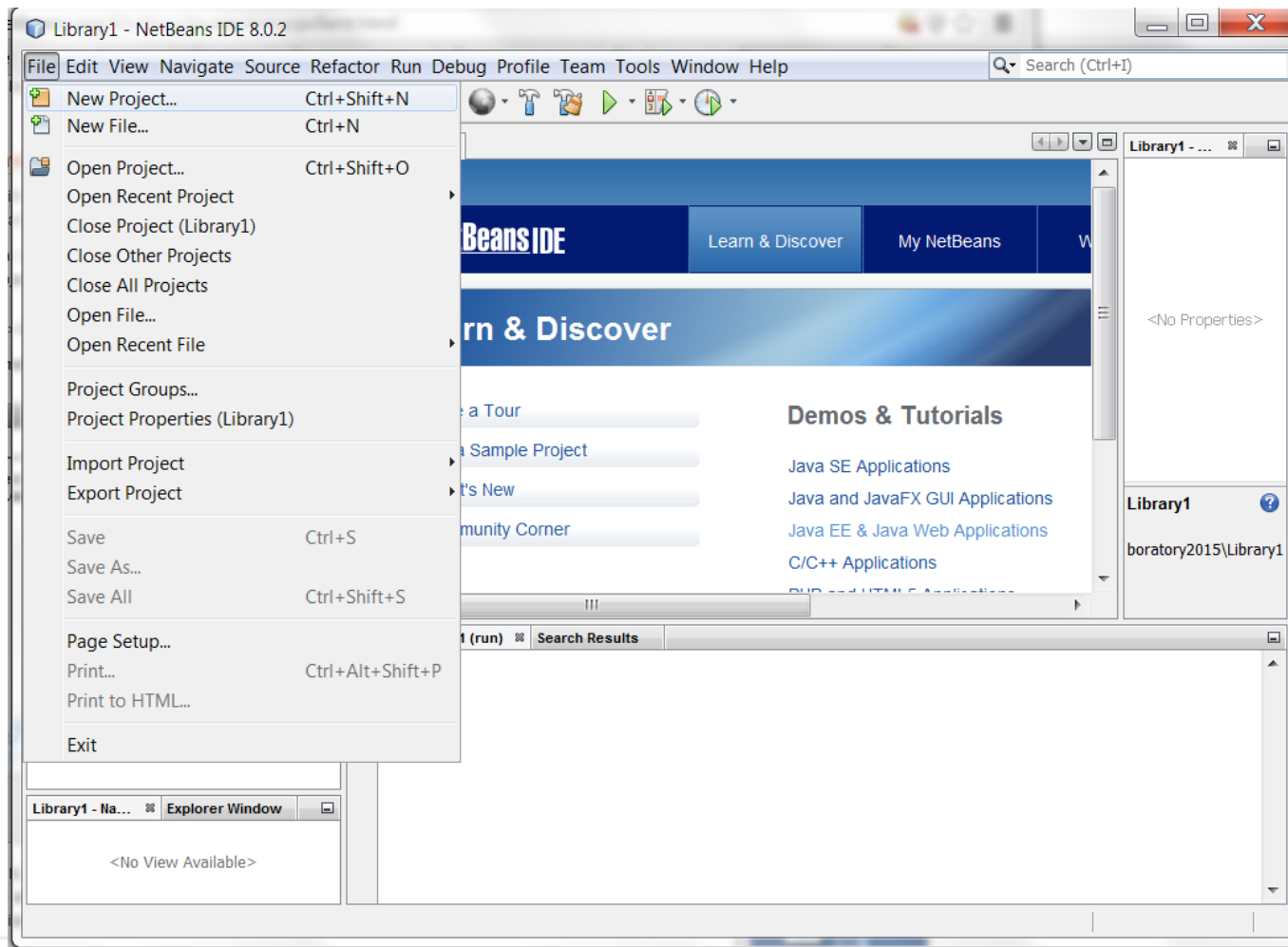
<https://netbeans.org/kb/docs/javaee/entappclient.html>

from

https://netbeans.org/kb/trails/java-ee.html?utm_source=netbeans&utm_campaign=welcomepage

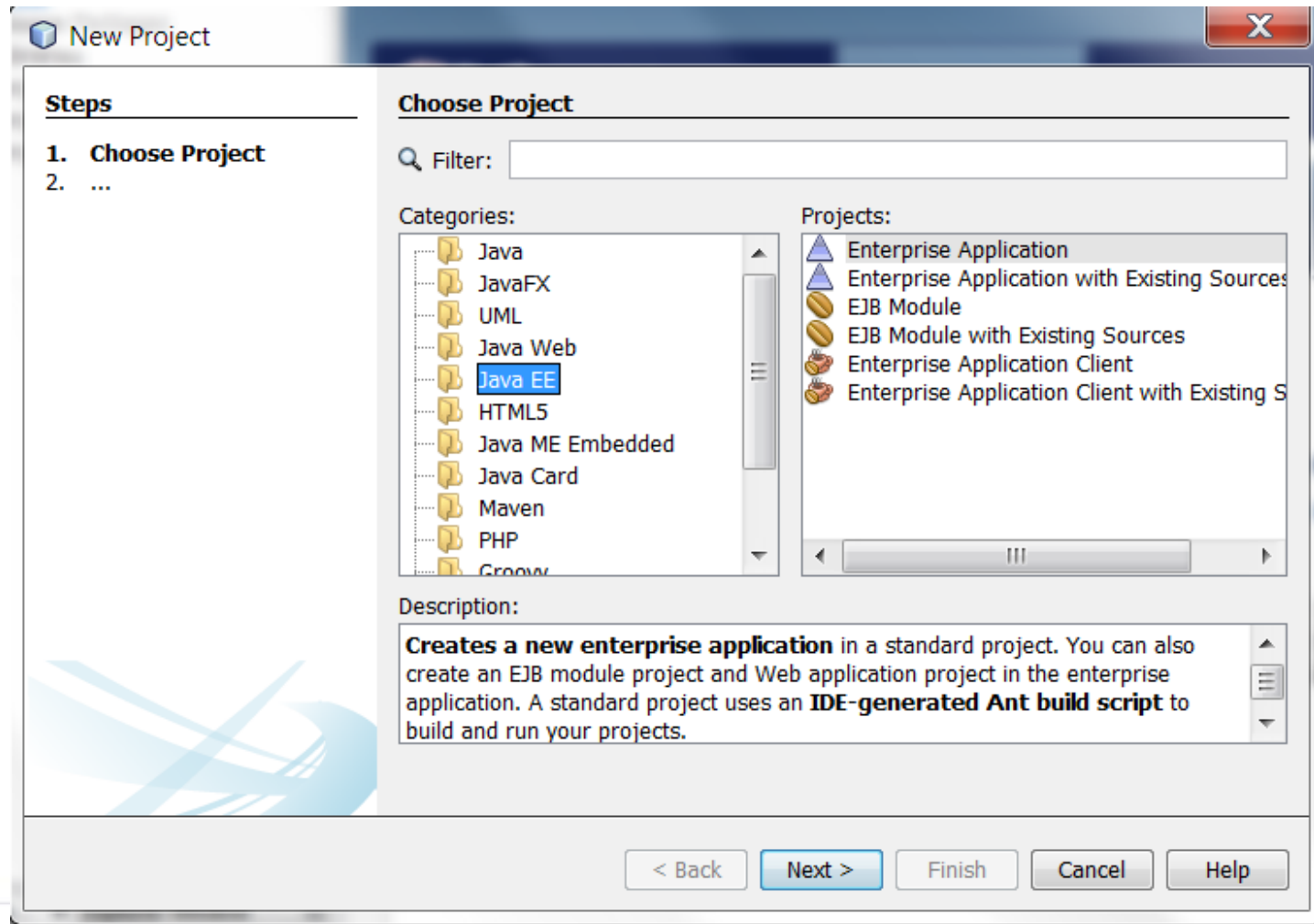


1. / 1.1. Create the Java EE project - right click File/New Project





1.2. Select Java EE\ Enterprise Application and click Next





1.3. Name and Location of EE project and next

The screenshot shows a Windows-style dialog box titled "New Enterprise Application". On the left, a "Steps" pane lists three steps: "1. Choose Project", "2. Name and Location" (which is highlighted), and "3. Server and Settings". The main area is titled "Name and Location" and contains the following fields and controls:

- Project Name:** A text box containing "Library1_EJB1".
- Project Location:** A text box containing "C:\EnglishLecture\Kruczkiewicz\Laboratory2015" with a "Browse..." button to its right.
- Project Folder:** A text box containing "nglishLecture\Kruczkiewicz\Laboratory2015\Library1_EJB1".
- Use Dedicated Folder for Storing Libraries:** An unchecked checkbox.
- Libraries Folder:** An empty text box with a "Browse..." button to its right.

Below the "Libraries Folder" field, there is a note: "Different users and projects can share the same compilation libraries (see Help for details)."

At the bottom of the dialog, there are five buttons: "< Back", "Next >" (highlighted in blue), "Finish", "Cancel", and "Help".



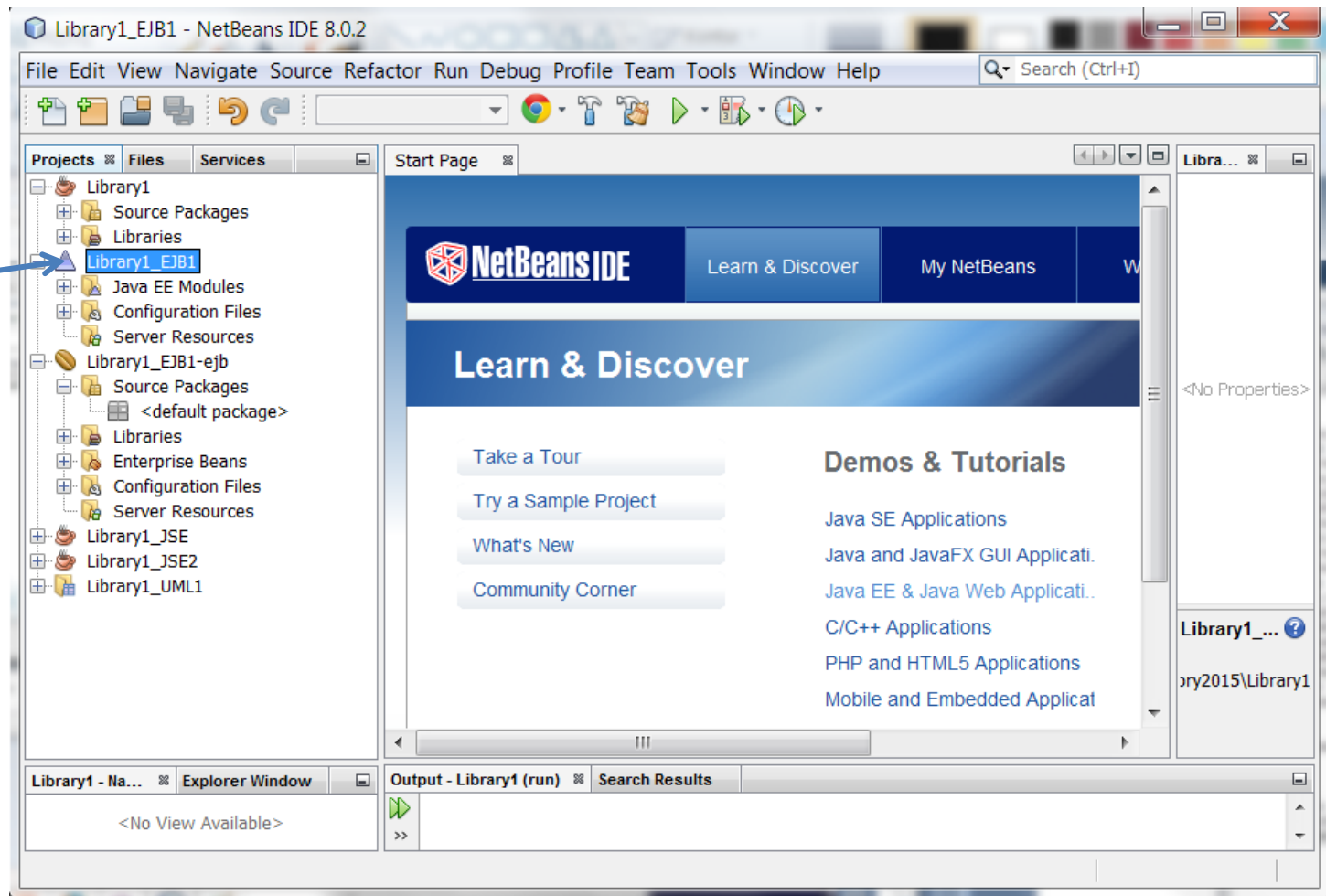


1.4. Server and Settings of EE project and next

The screenshot shows the 'New Enterprise Application' wizard window. The title bar says 'New Enterprise Application' with a close button. On the left, under the 'Steps' section, there is a list: 1. Choose Project, 2. Name and Location, and 3. **Server and Settings**. The main area is titled 'Server and Settings'. It contains the following fields and controls:

- Server:** A dropdown menu showing 'GlassFish Server 4.1' and an 'Add...' button.
- Java EE Version:** A dropdown menu showing 'Java EE 7'.
- Create EJB Module:** A checked checkbox followed by a text field containing 'Library1_EJB1-ejb'.
- Create Web Application Module:** An unchecked checkbox followed by a text field containing 'Library1_EJB1-war'.

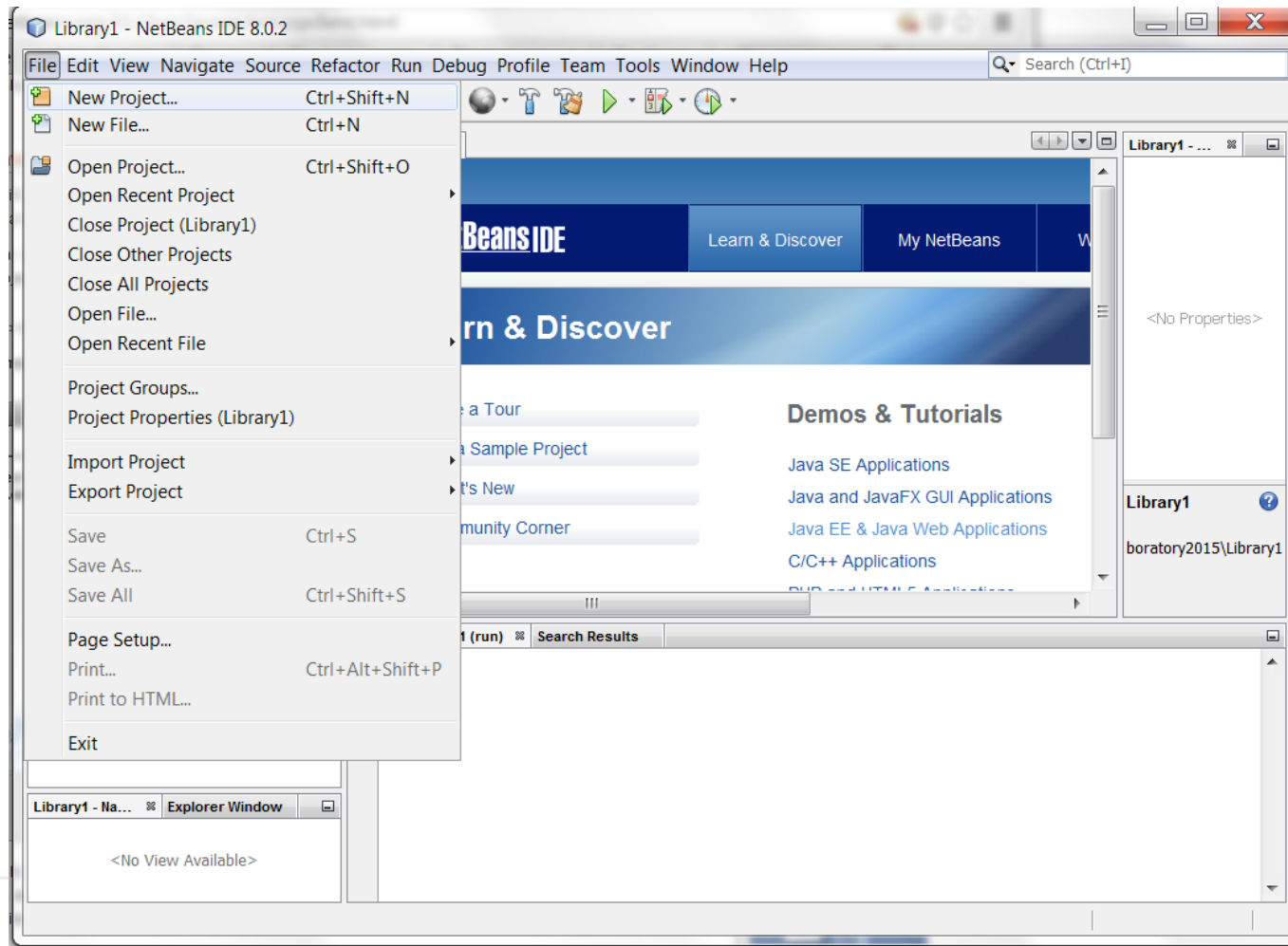
At the bottom of the window, there are five buttons: '< Back', 'Next >', 'Finish' (highlighted in blue), 'Cancel', and 'Help'.



1.5. The new
EE project

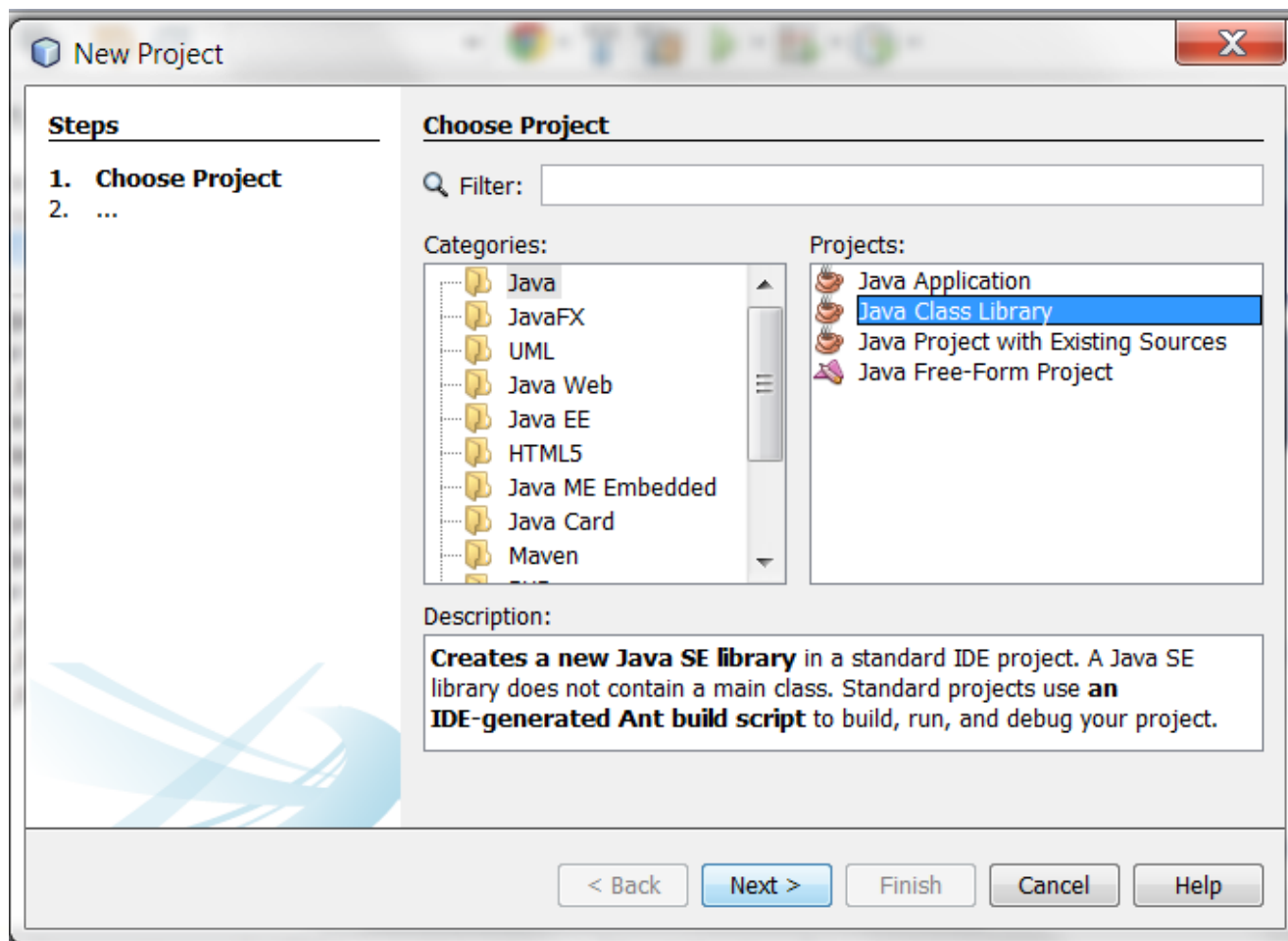


2. / 2.1. Creation the library project for interface of EJB





2.2. Creation the library project for interface of EJB – select Java/Java Class Library items





2.3. Creation the library project for interface of EJB – fill the Project Name and select the Project Location

New Java Class Library

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

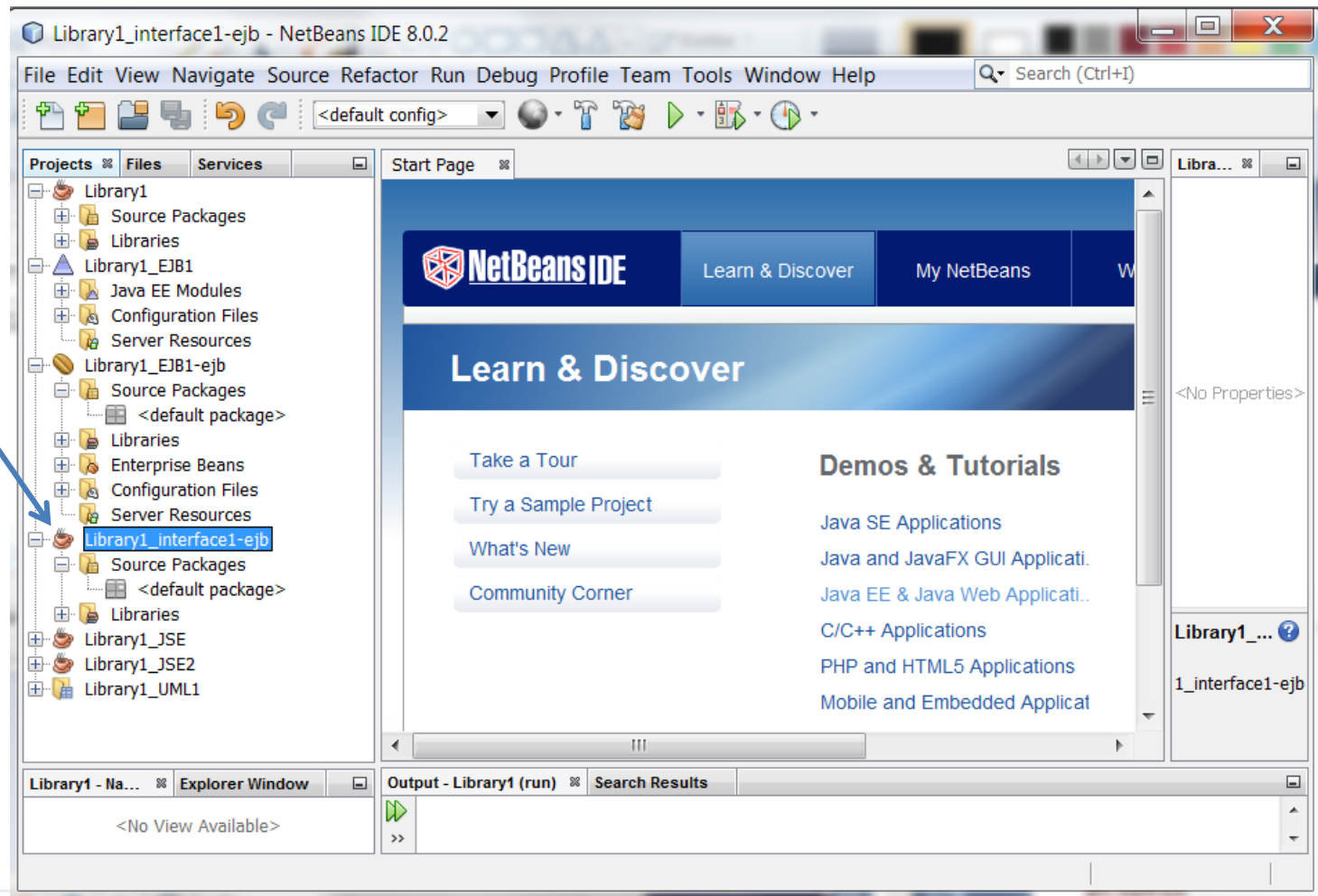
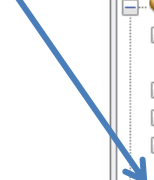
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

< Back Next > **Finish** Cancel Help

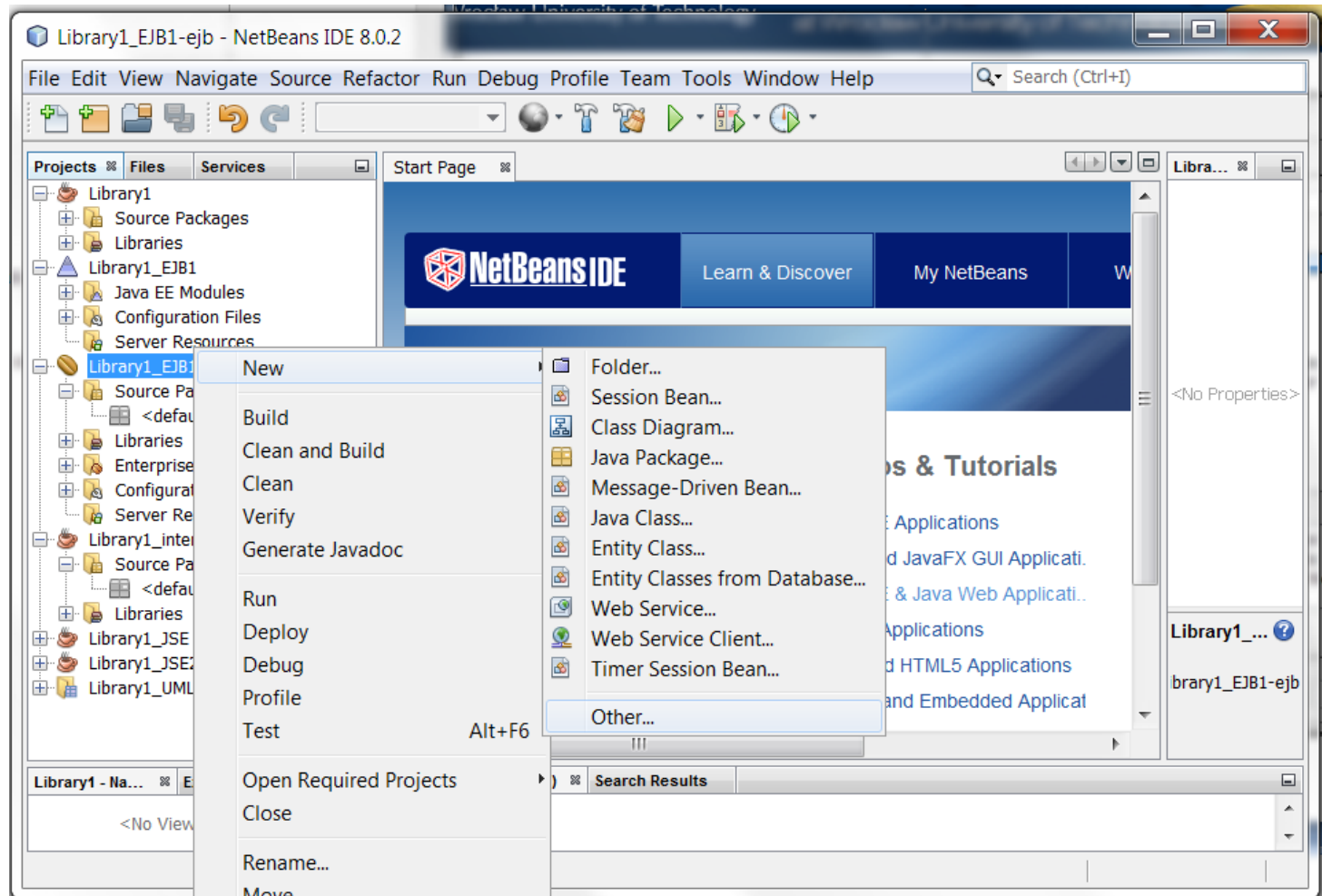


2.4. The new empty library project



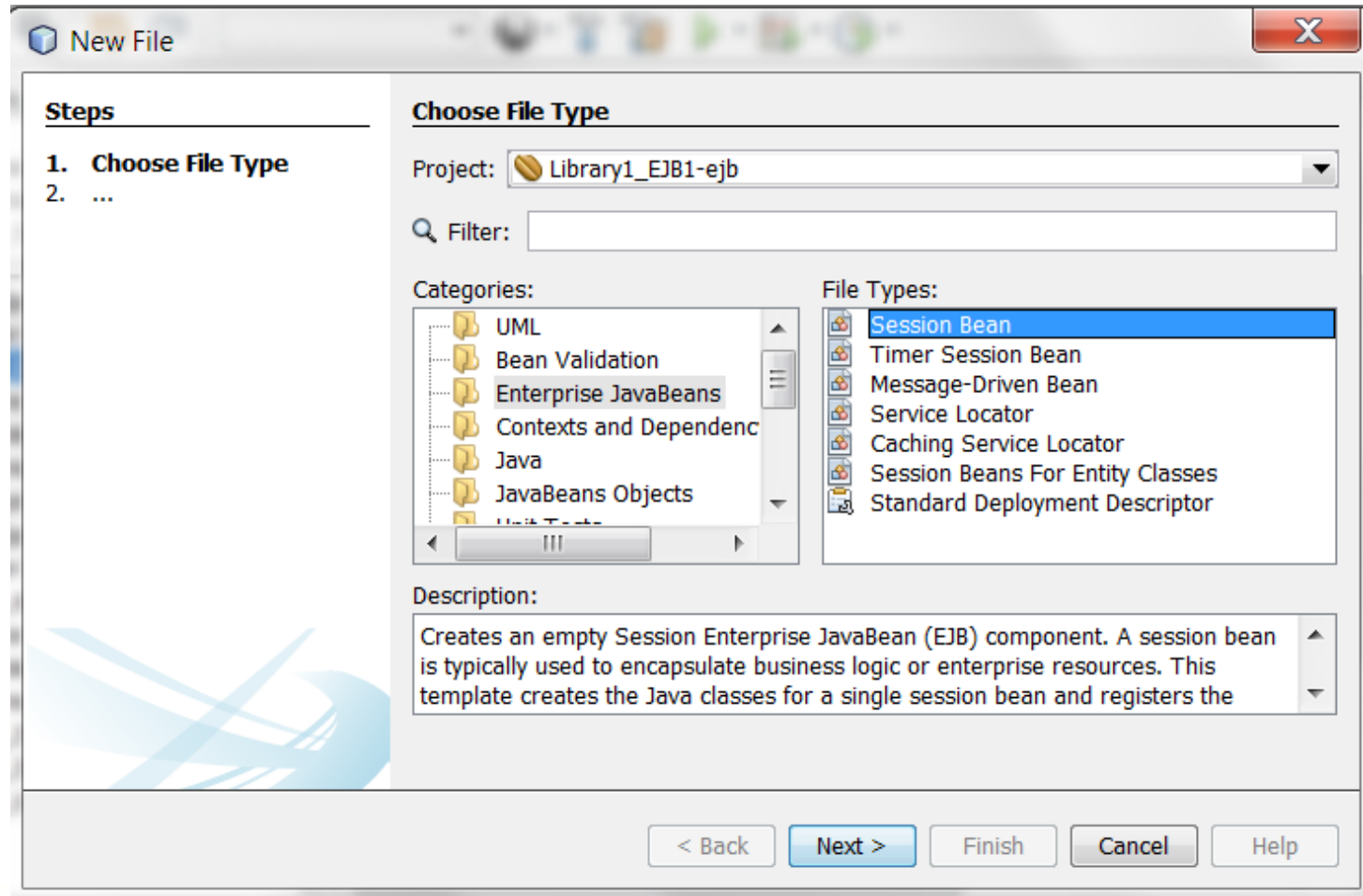


3. / 3.1. Adding the EJB as the Session Facade – Right click/New/Other



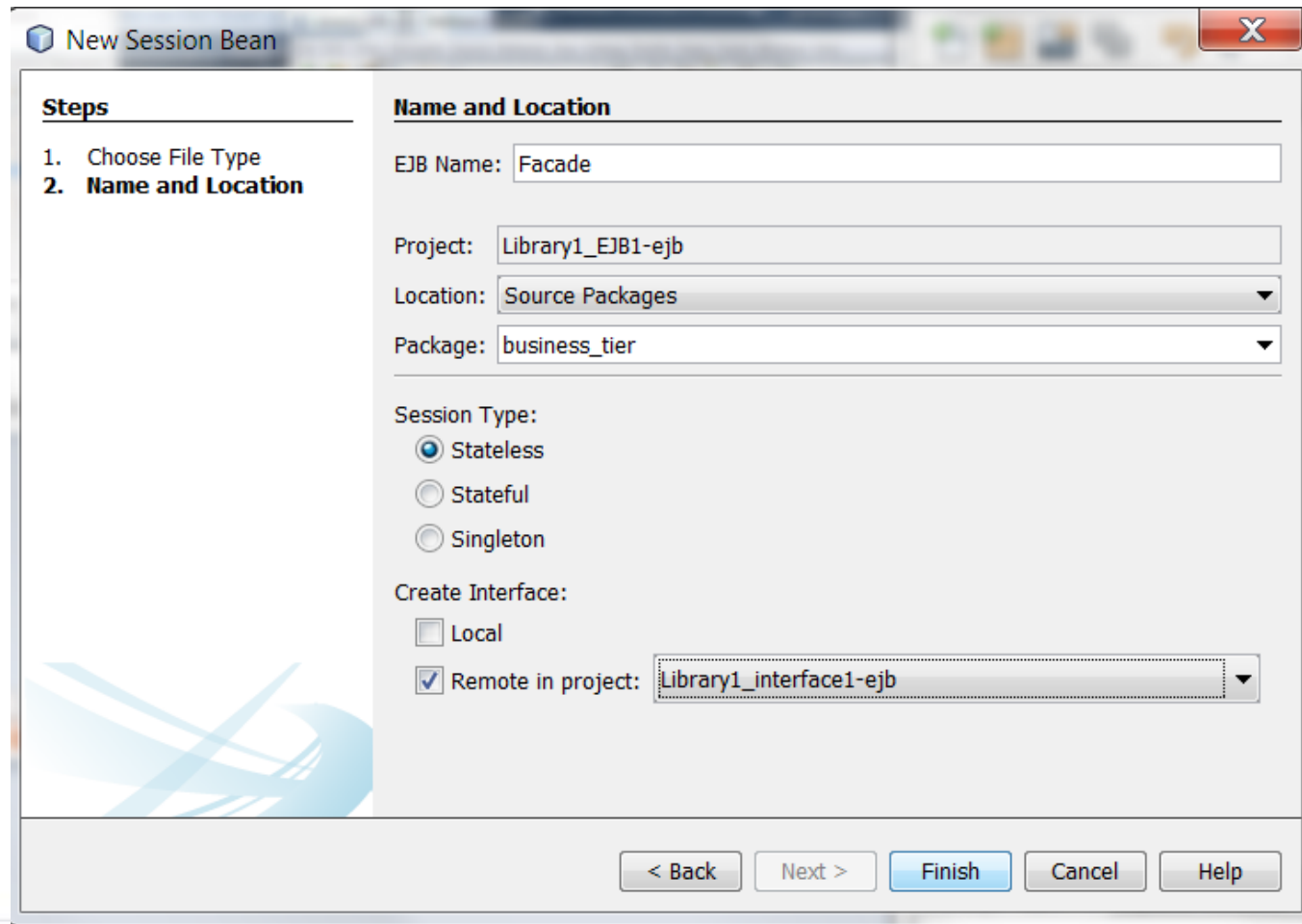


3.2. Adding the EJB as the Session Facade – select Enterprise JavaBeans/ Session Bean





3.3. Adding the EJB as the Session Facade – fill the EJB Name, name of package, select the Session Type and create interface for Remote EJB

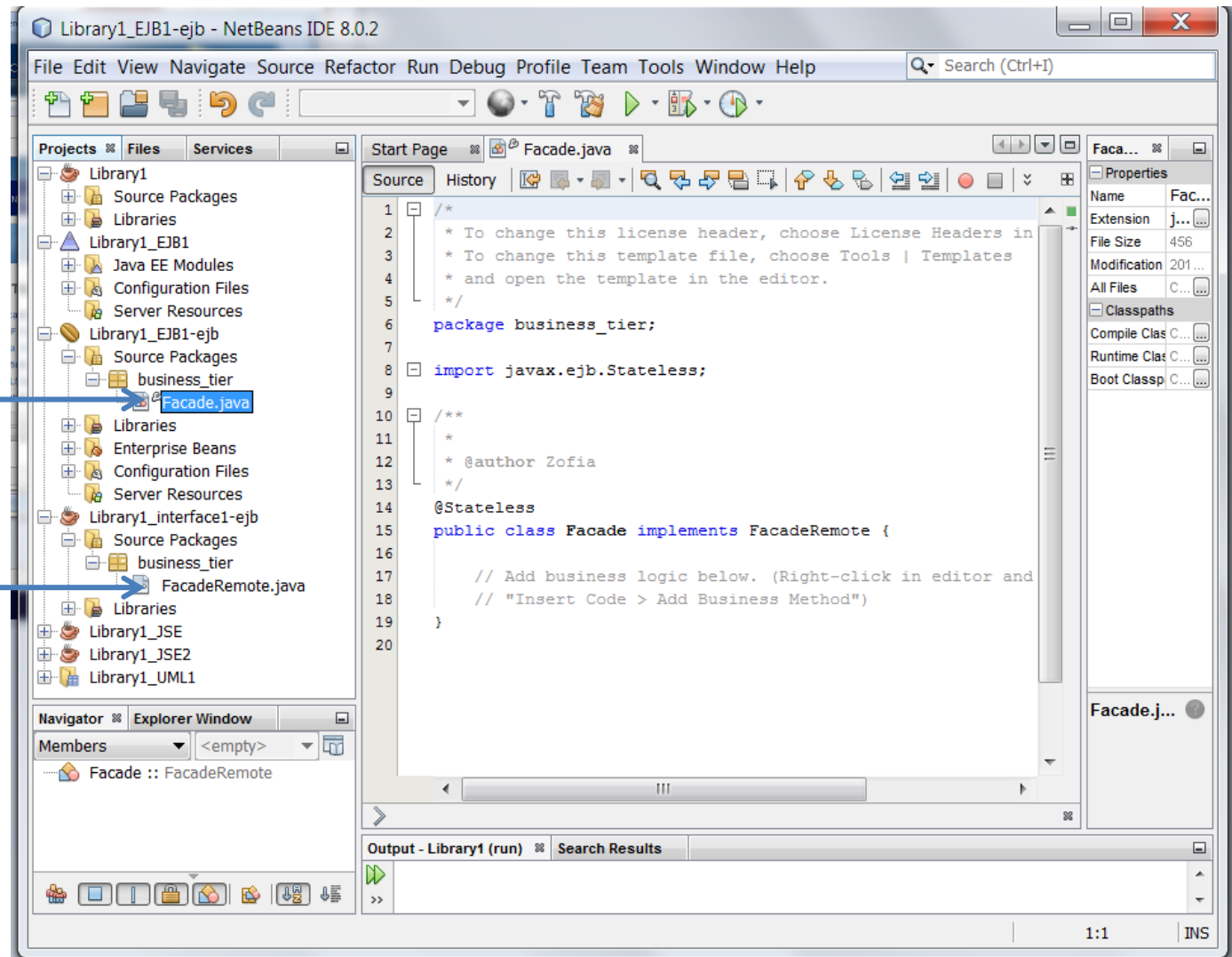


The image shows a 'New Session Bean' dialog box with the following fields and options:

- Steps:**
 1. Choose File Type
 2. Name and Location
- Name and Location:**
 - EJB Name: Facade
 - Project: Library1_EJB1-ejb
 - Location: Source Packages
 - Package: business_tier
- Session Type:**
 - ☒ Stateless
 - ☐ Stateful
 - ☐ Singleton
- Create Interface:**
 - ☐ Local
 - ☒ Remote in project: Library1_interface1-ejb

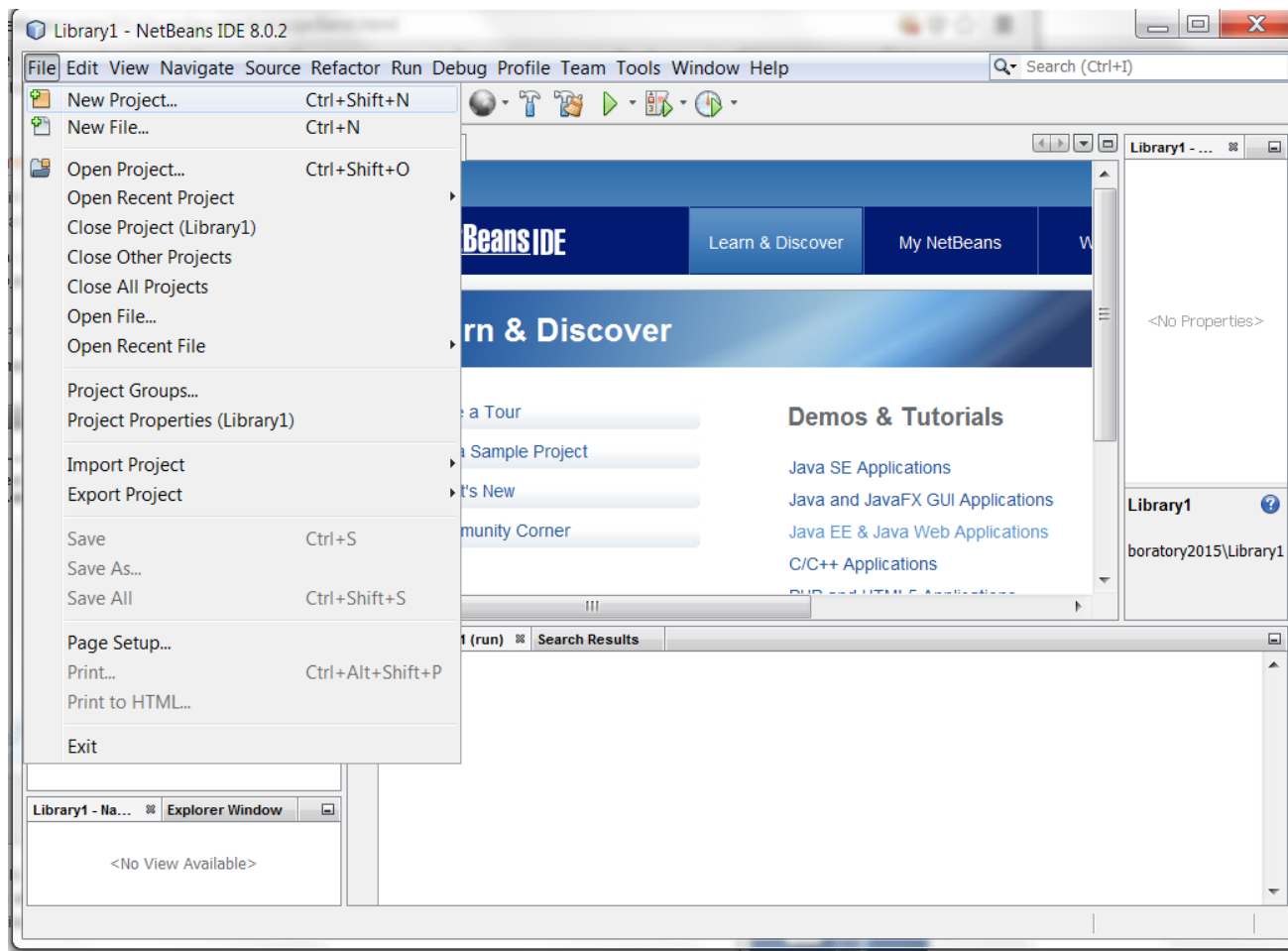
Buttons at the bottom: < Back, Next >, Finish, Cancel, Help.

3.4. The new empty EJB and its empty interface



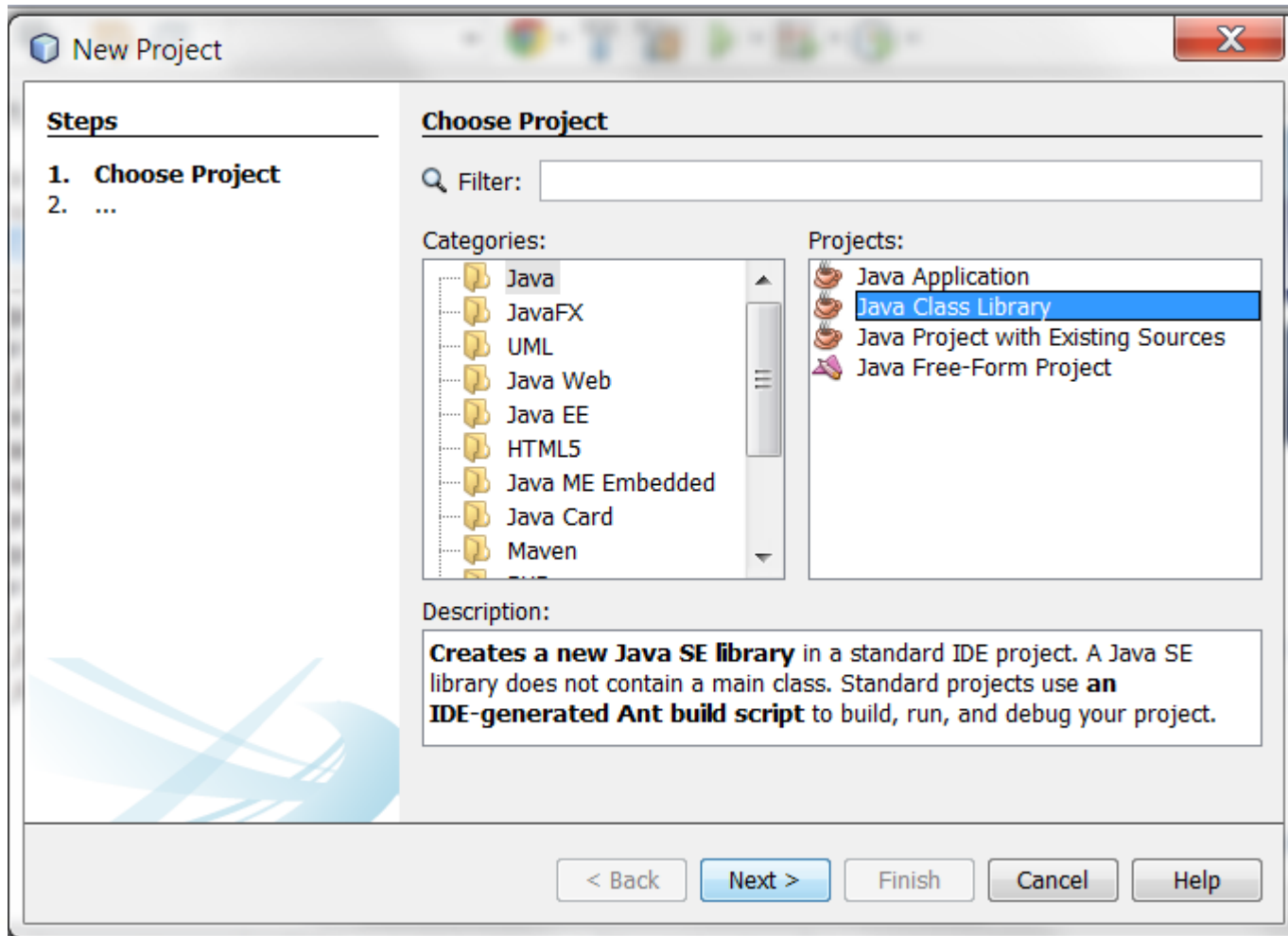


4. / 4.1./4.1.1. Creation the library project for code from lab1-2





4.1.2. Creation the library project for code from lab1-2 - select Java/Java Class Library items





4.1.3. Creation the library project for code from lab1-2 - fill the Project Name and select the Project Location

New Java Class Library

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

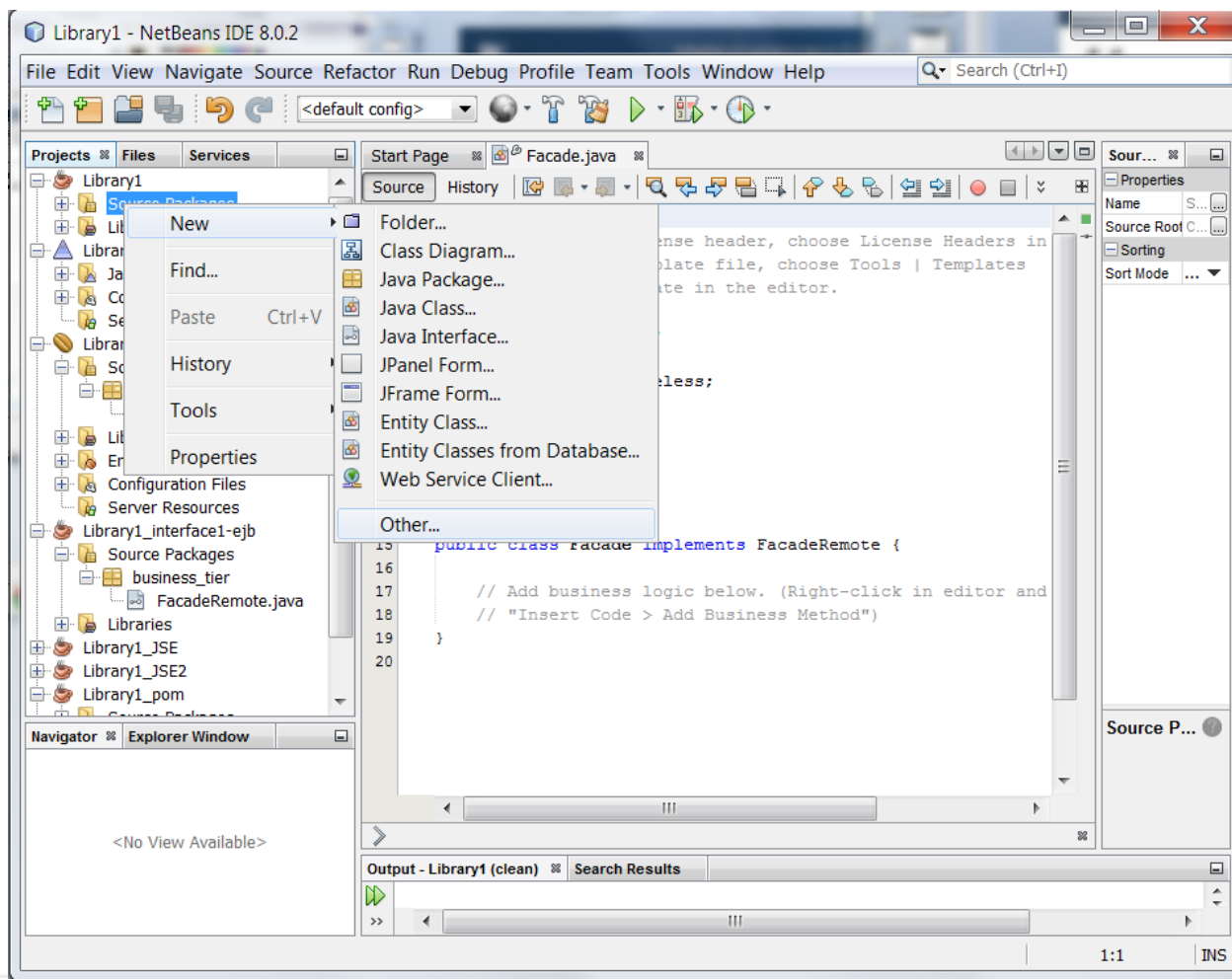
Different users and projects can share the same compilation libraries (see Help for details).

< Back Next > **Finish** Cancel Help



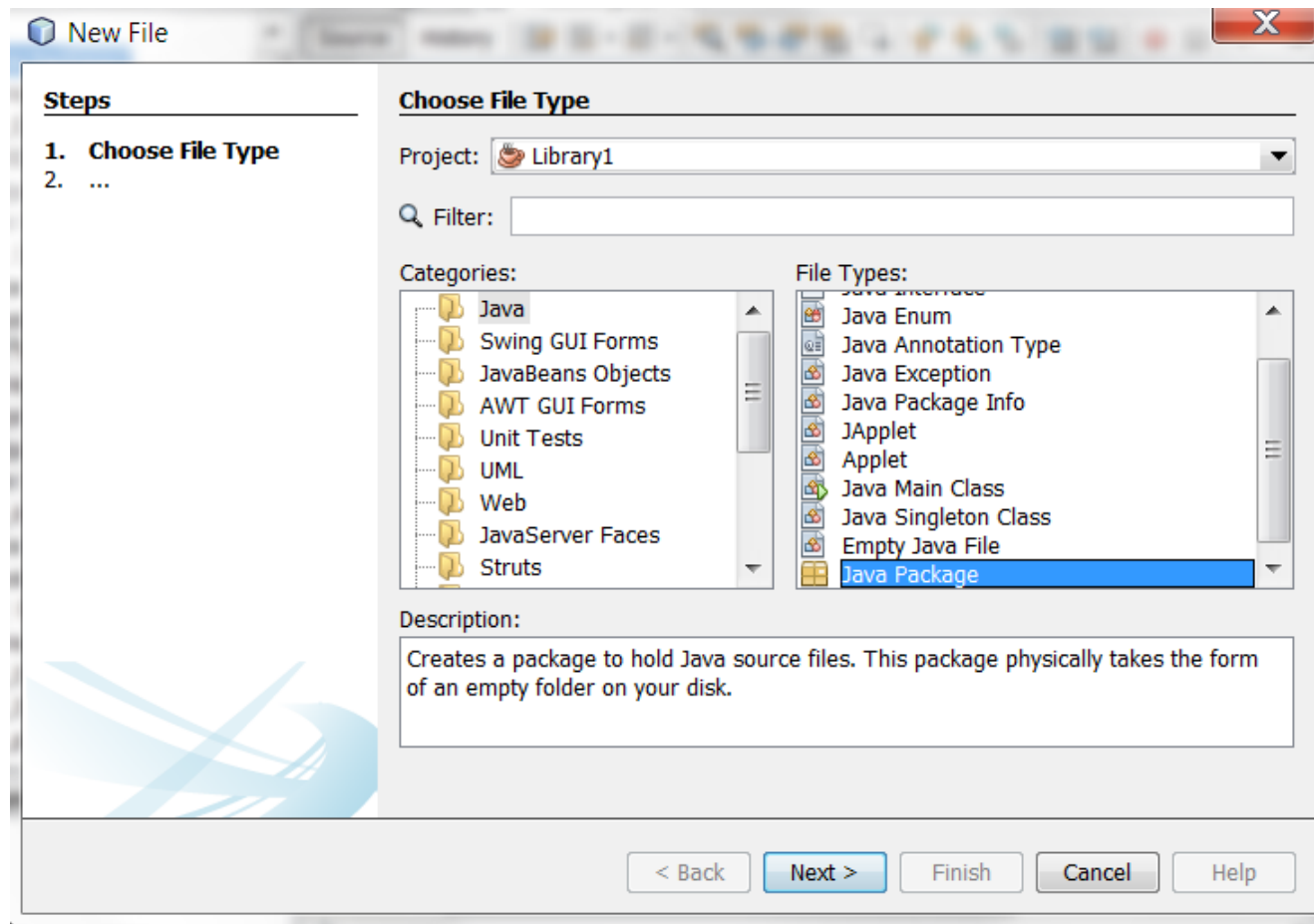


4.2. /4.2.1 Creation the package for TFacade and TFactory classes



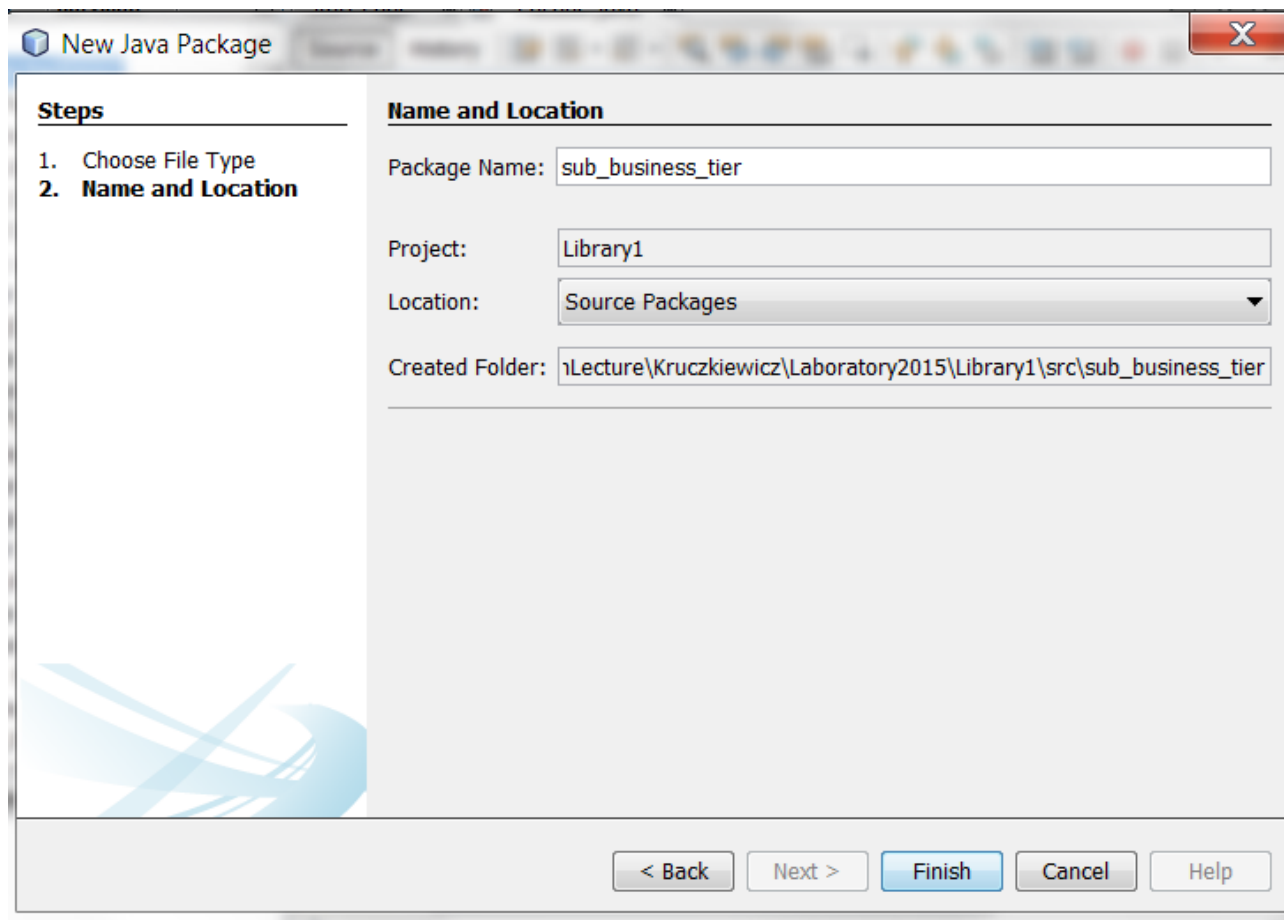


4.2.2. Creation the package for TFacade and TFactory classes – select Java/ Java Package





4.2.3. Creation the package for TFacade and TFactory classes – fill the name of package and select location . The result on the right.



New Java Package

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

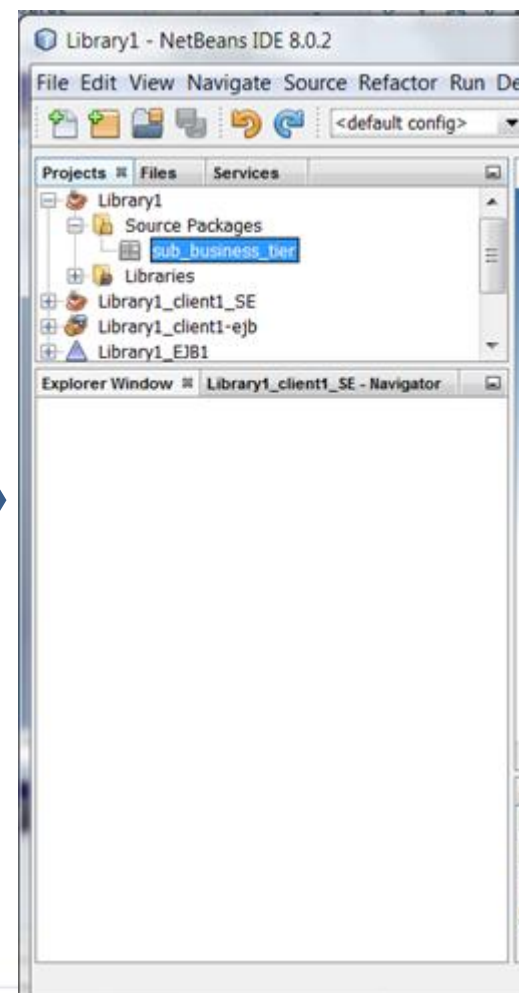
Package Name:

Project:

Location:

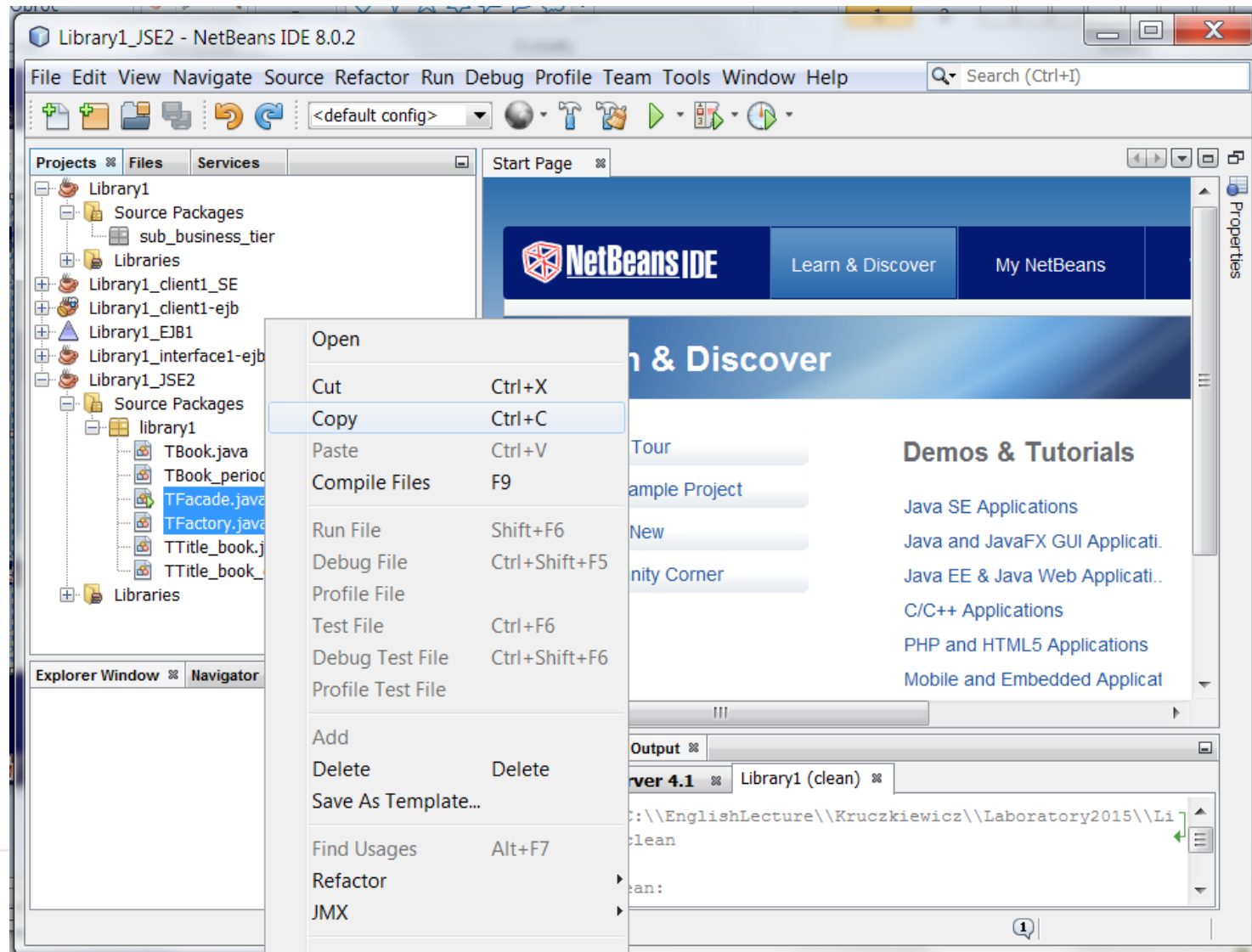
Created Folder:

< Back Next > Finish Cancel Help



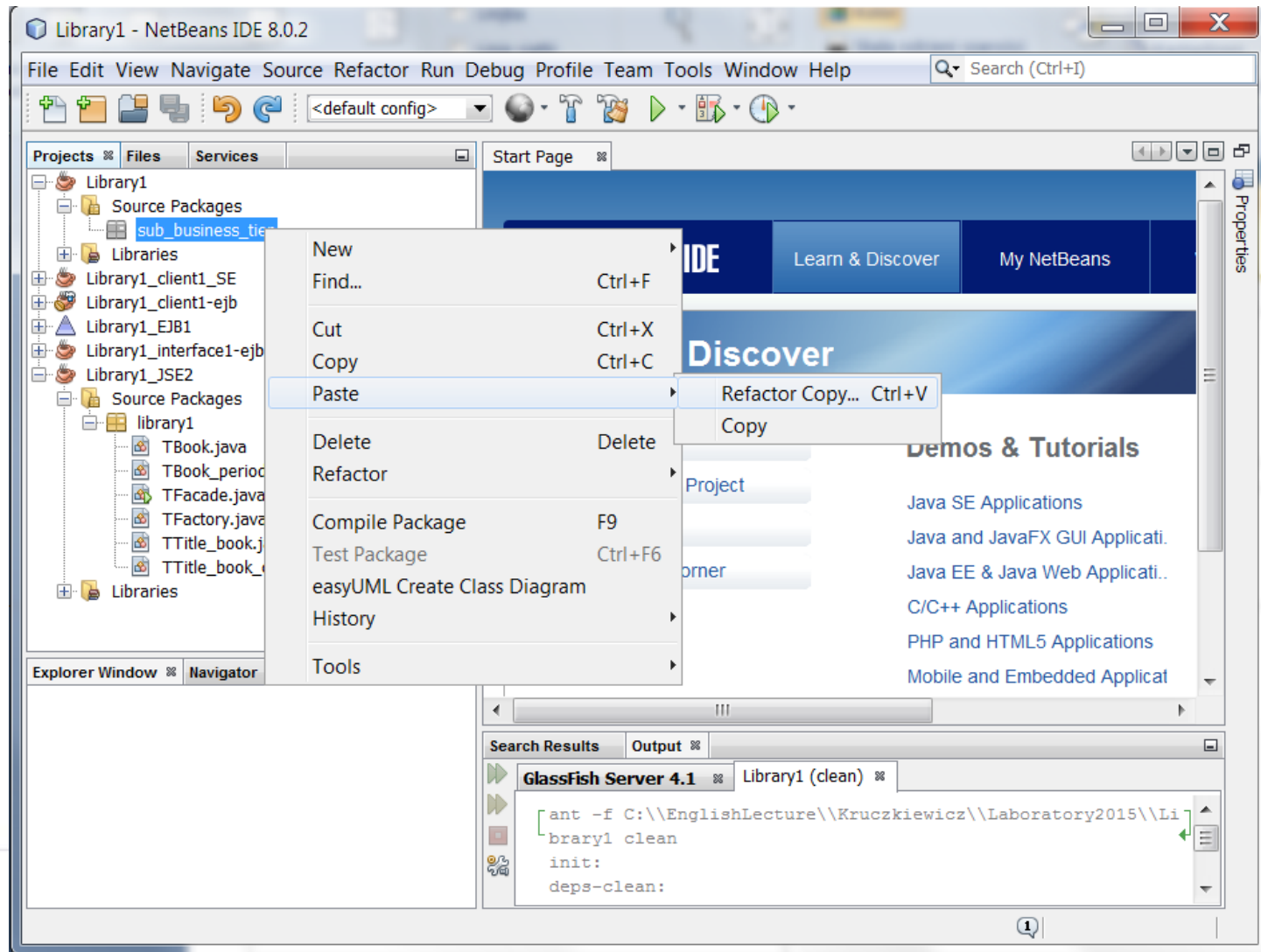


4.3. /4.3.1. Copying of TFacade and TFactory classes



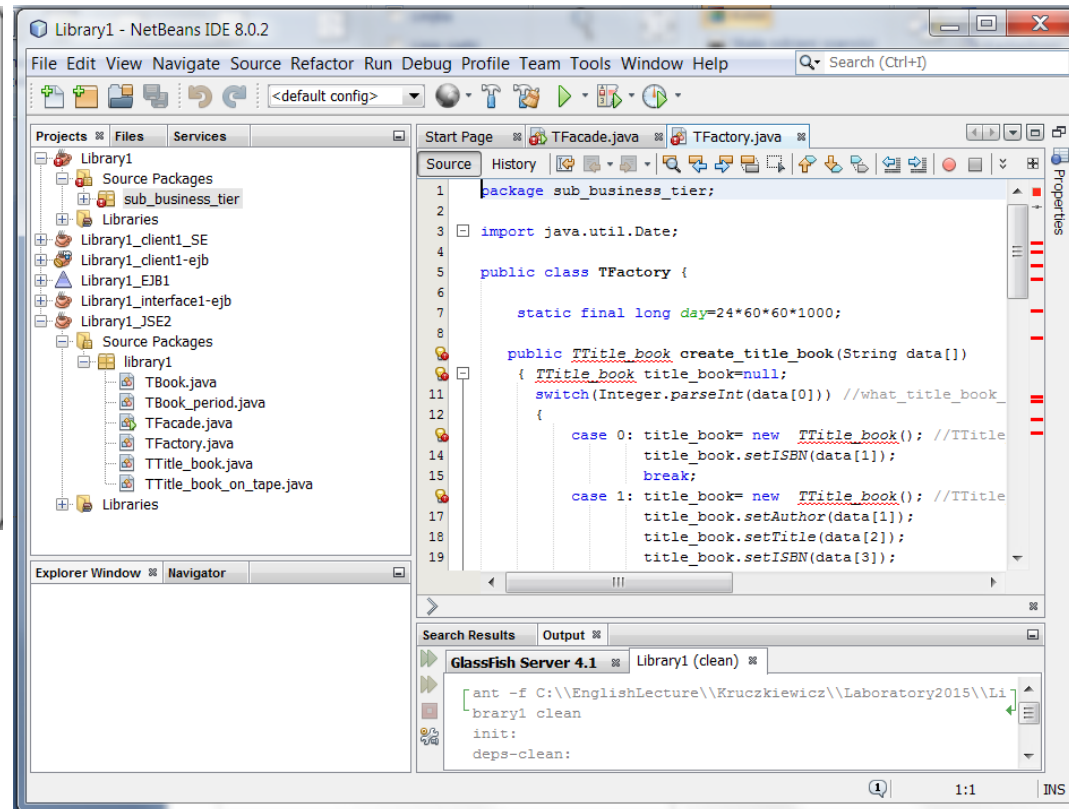
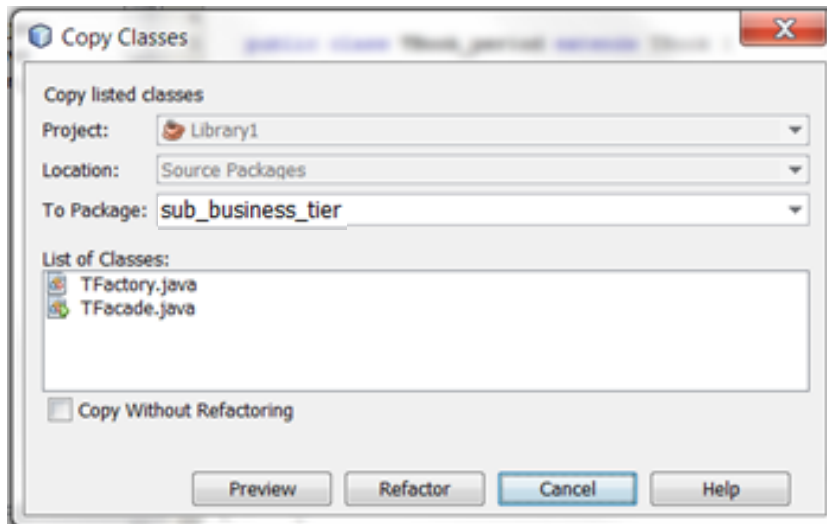


4.3.2. Paste the copies of the TFacade and TFactory classes from the lab1-2 project to the Library1 project (select the **sub_business_tier** package).



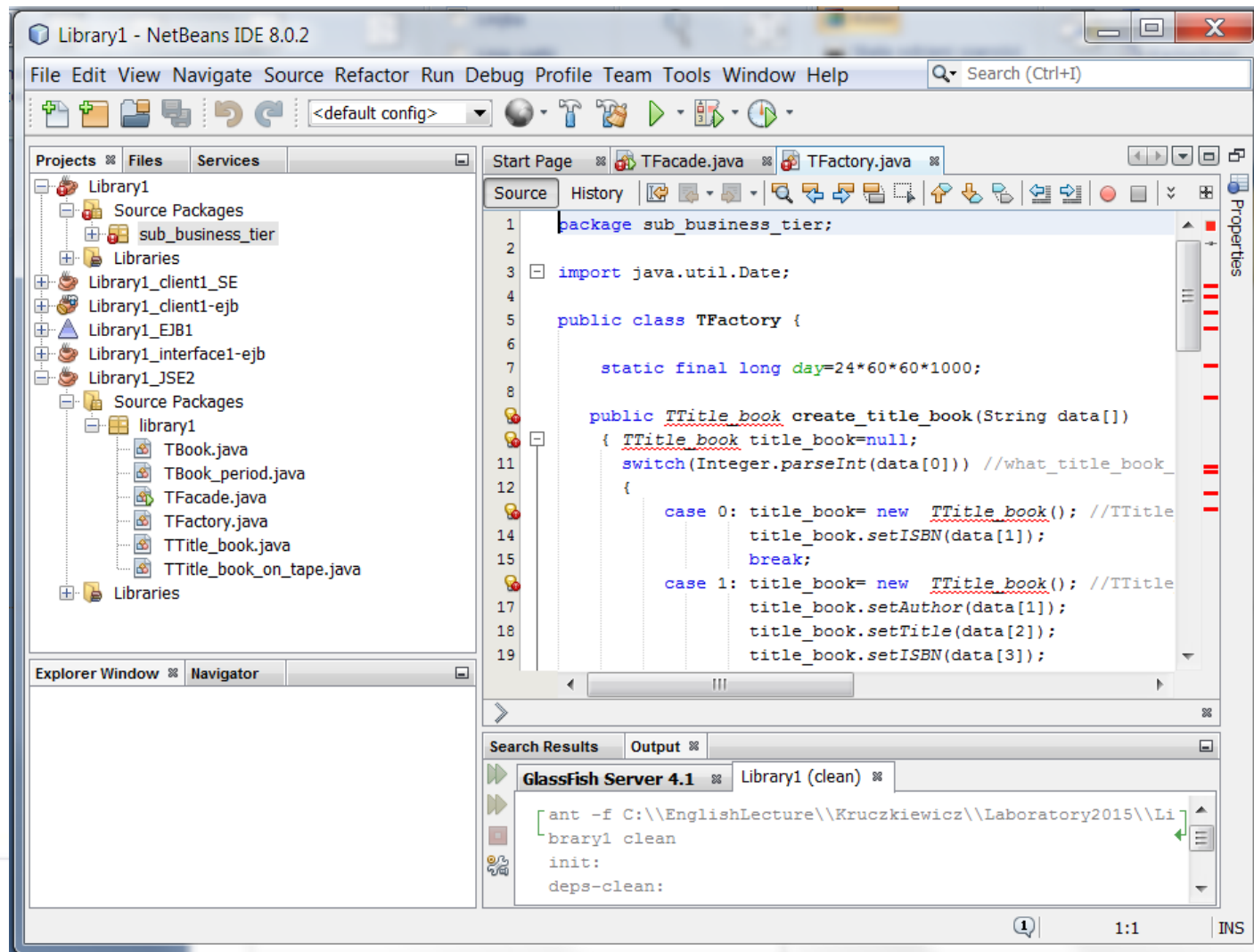


4.3.3. Paste the copies of the TFacade and TFactory classes from the lab1-2 project to the Library1 project – the refactoring process (click Refactor – on the left; the result on the right).



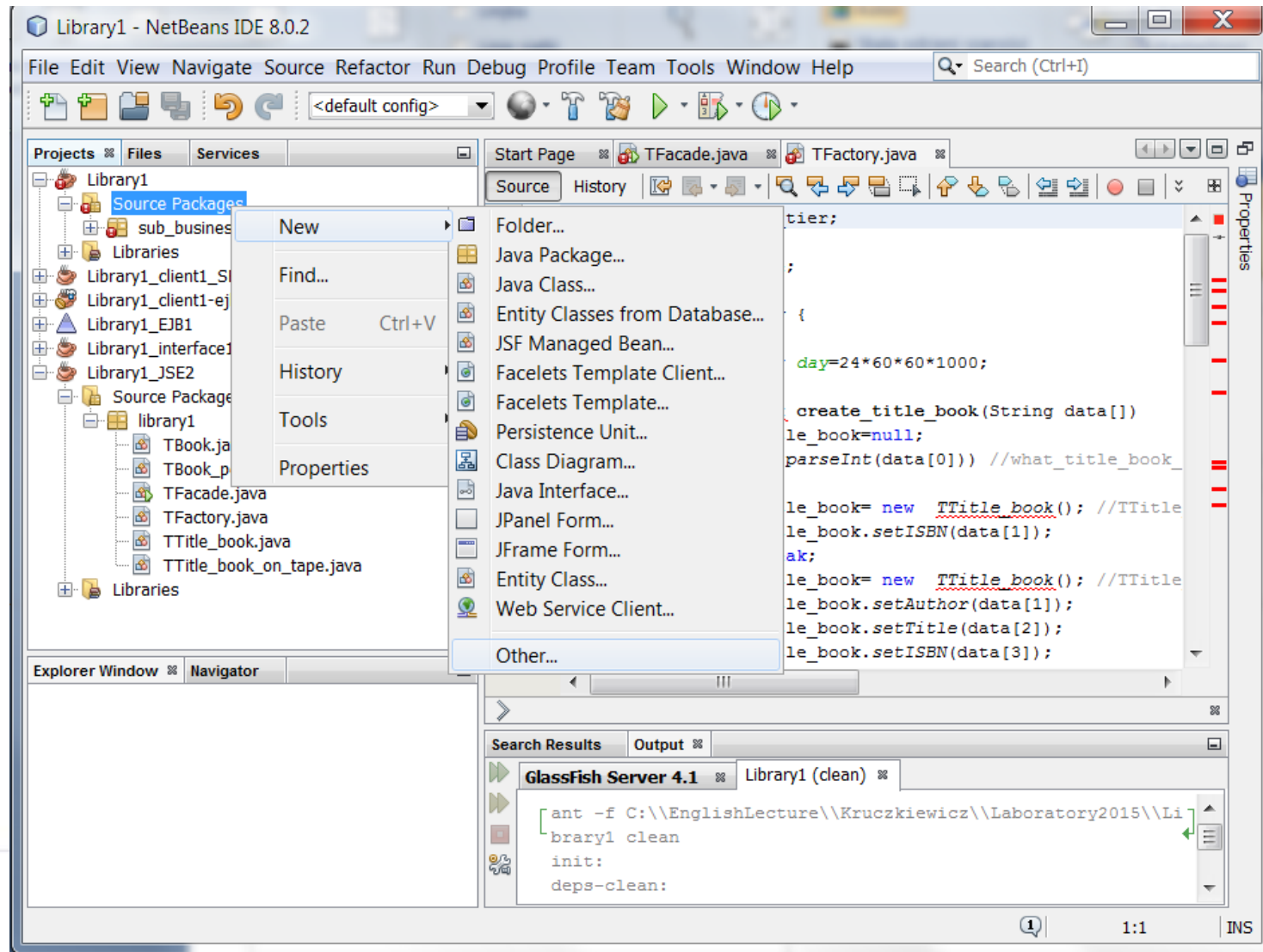


4.3.4. Paste the copies of the TFacade and TFactory classes from the lab1-2 project to the Library1 project – result



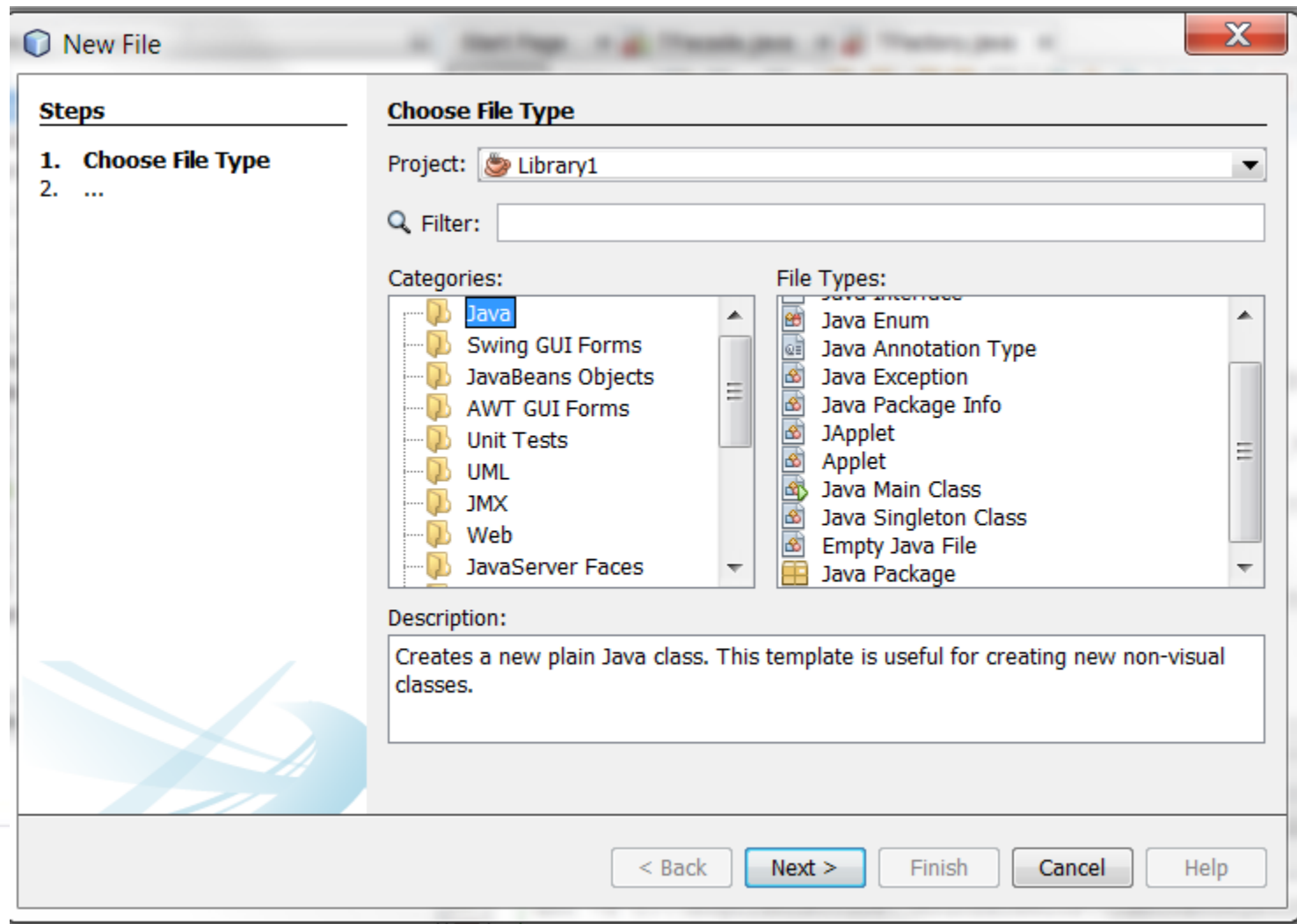


4.4. / 4.4.1. Creation of the new package for data classes from lab1-2





4.4.2. Creation of new package for data classes from lab1-2 – select Java/ Java Package





4.4.3. Creation of new package for data classes from lab1-2 – fill the name of package and select location

New Java Package

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Package Name:

Project:

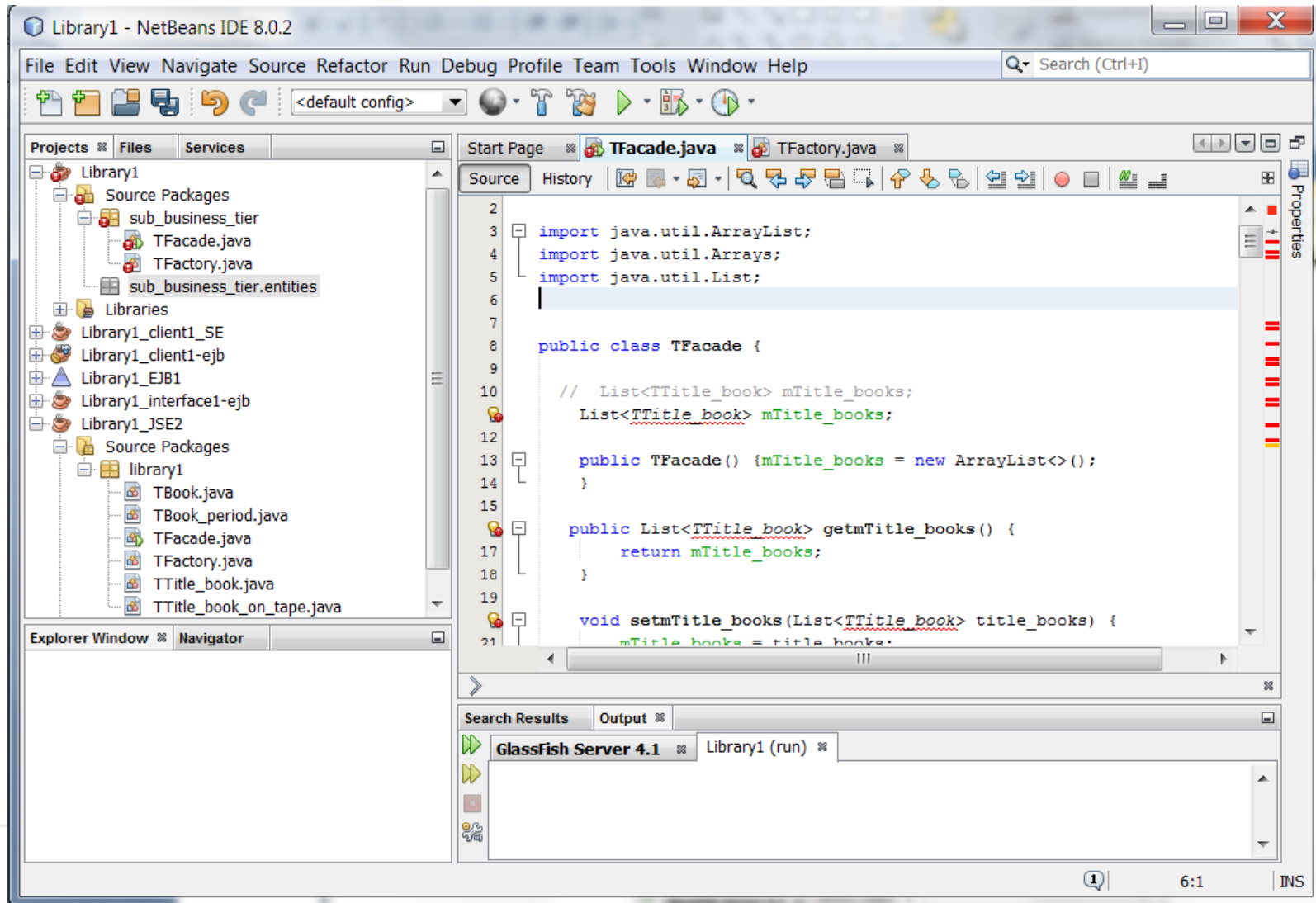
Location:

Created Folder:

< Back Next > **Finish** Cancel Help

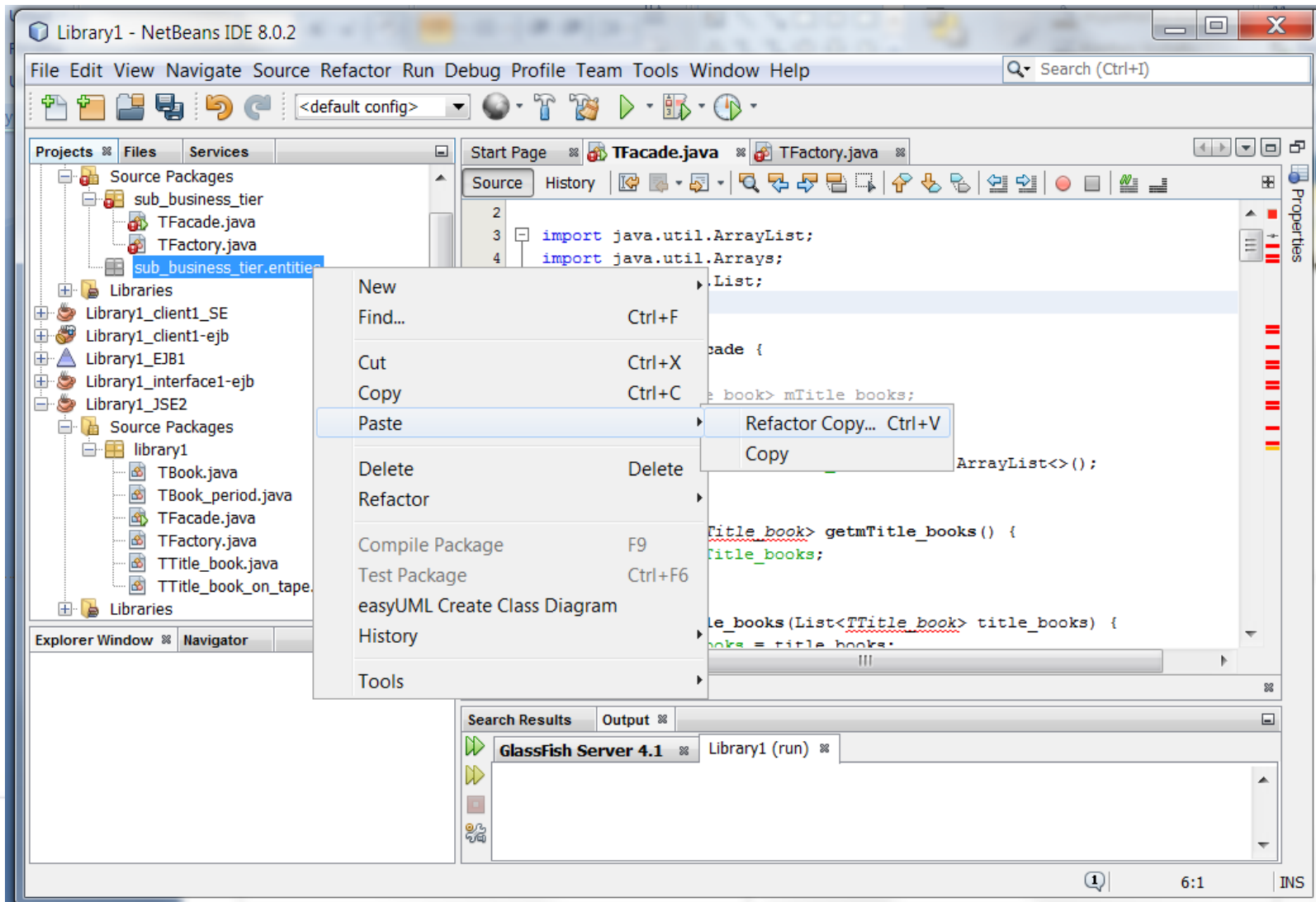


4.4. Result of creating of the sub_business_tier.entities packages



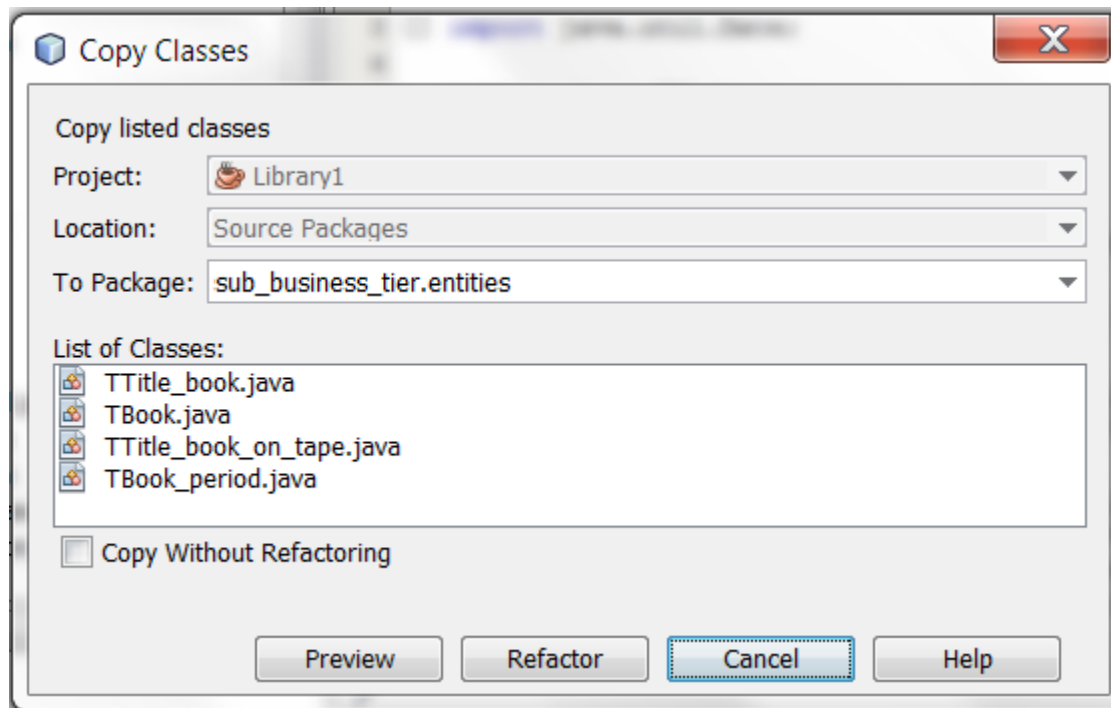


4.5.2 Paste the copies of the TTitle_book and other classes as data classes from the lab1-2 project to the Library1 project (select the **sub_business_tier.entities** package).



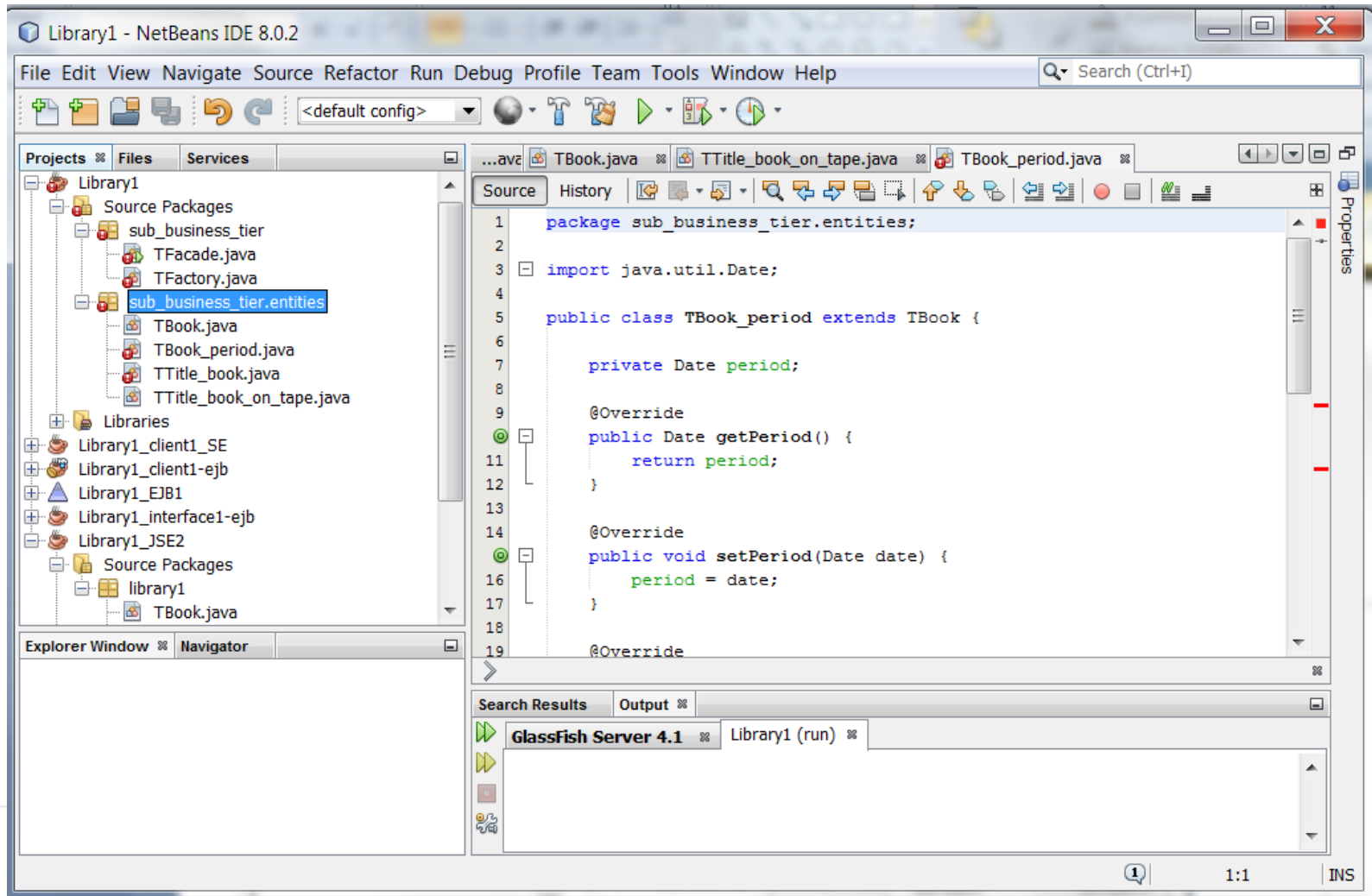


4.5.3. Paste the copies of the TTitle_book and other classes as data classes from the lab1-2 project to the Library1 project – the refactoring process (click the Refactor button).



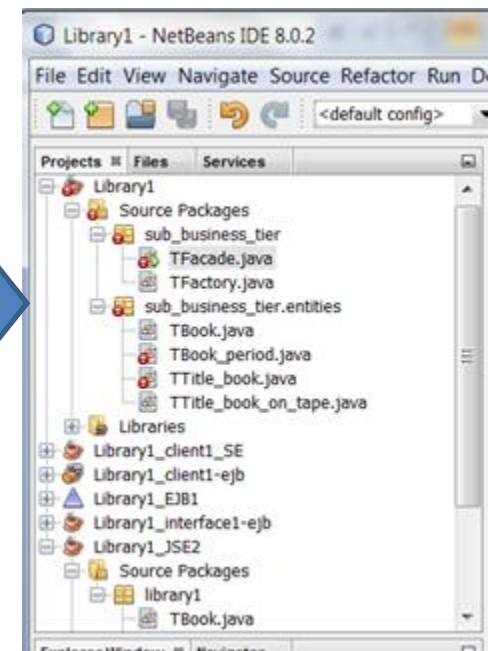
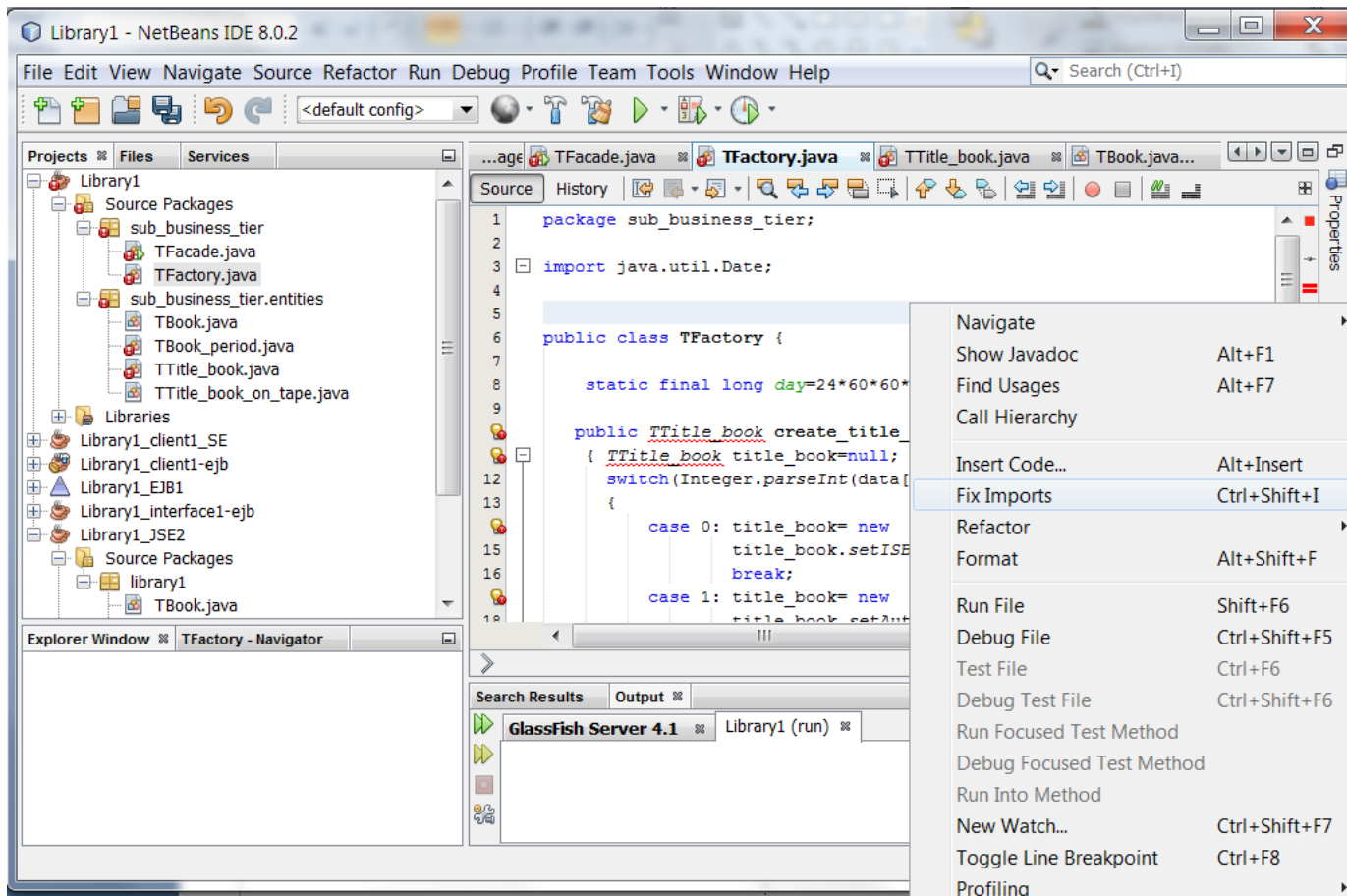


4.5.4. Paste the copies of the TTitle_book and other classes as data classes from the lab1-2 project to the Library1 project - result



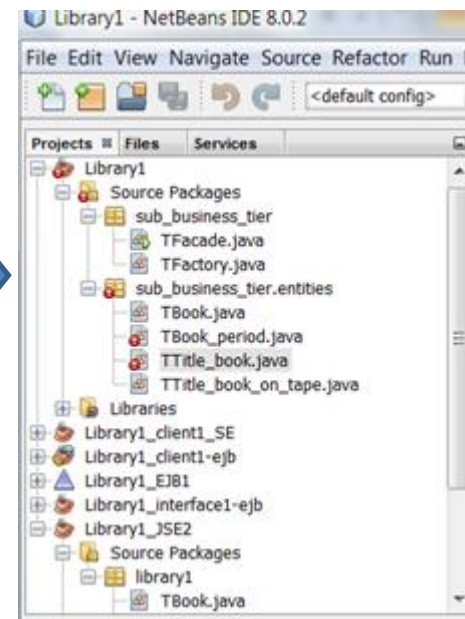
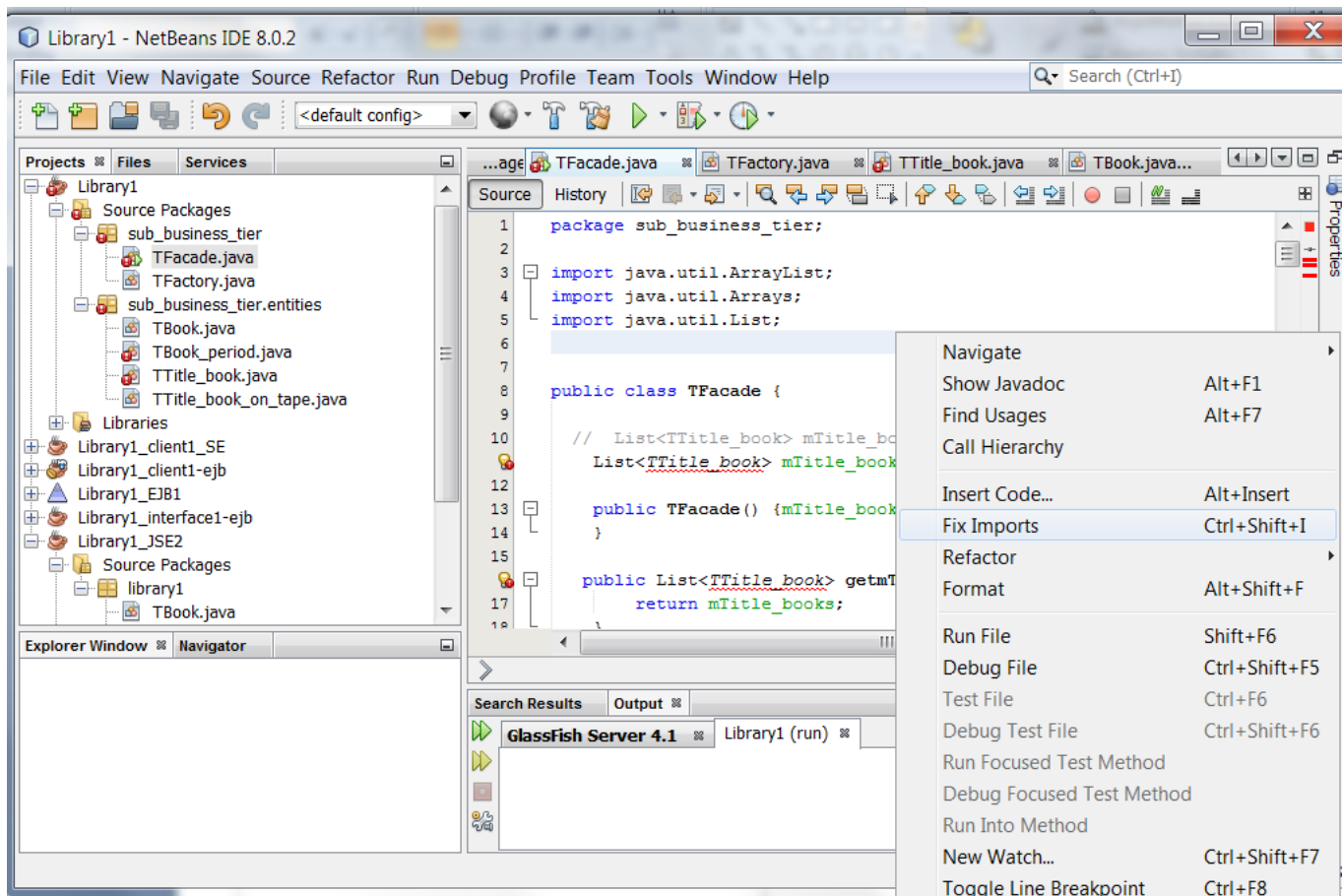


4.6. /4.6.1. „Fix Imports” process for adding the lacking imports of classes – the TFactory class



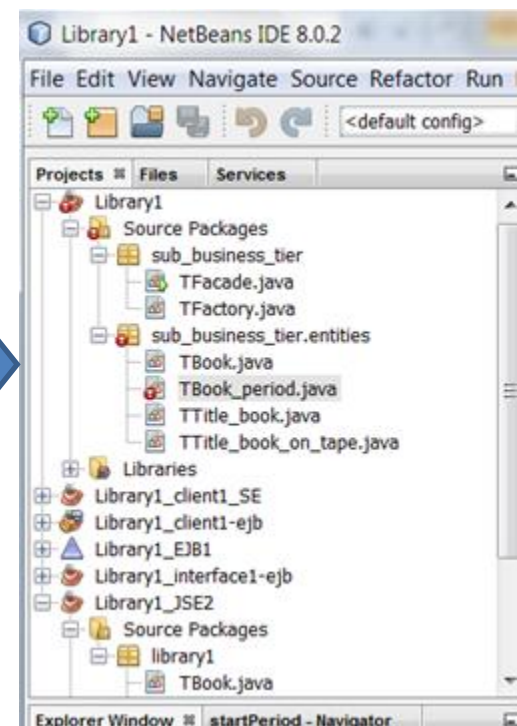
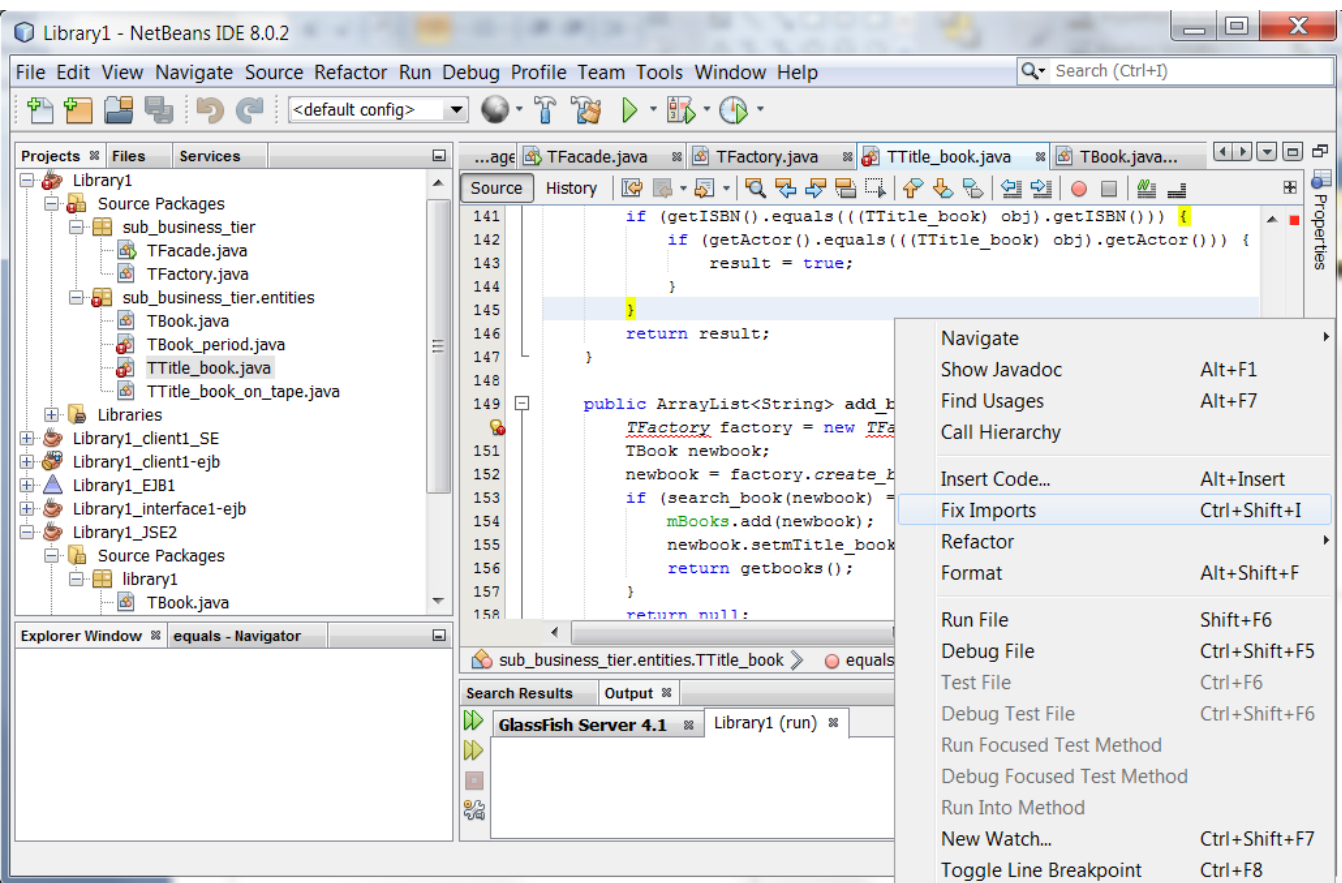


4.6.2. „Fix Imports” process for adding the lacking imports of classes – the TFacade class





4.6.3. „Fix Imports” process for adding the lacking imports of classes – the TTitle_book class





4.6.4. „Fix Imports” process for adding the lacking imports of classes – the TBook_period class

Library1 - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects Files Services

Library1

Source Packages

sub_business_tier

TFacade.java

TFactory.java

sub_business_tier.entities

TBook.java

TBook_period.java

TTitle_book.java

TTitle_book_on_tape.java

Libraries

Library1_client1_SE

Library1_client1-ejb

Library1_EJB1

Library1_interface1-ejb

Library1_JSE2

Source Packages

library1

TBook.java

Explorer Window startPeriod - Navigator

Search Results Output

GlassFish Server 4.1 Library1 (run)

Source

History

Properties

public void setPeriod(Date date) {

period = date;

}

@Override

public boolean period_pass(Object

Date date = TFactory.mdays(

return period.compareTo(date

}

@Override

public void startPeriod(Object o

{

Date help = TFactory.mdays((

setPeriod(help);

}

@Override

public String toString() // you

sub_business_tier.entities.TBook_period

Navigate

Show Javadoc Alt+F1

Find Usages Alt+F7

Call Hierarchy

Insert Code... Alt+Insert

Fix Imports Ctrl+Shift+I

Refactor

Format Alt+Shift+F

Run File Shift+F6

Debug File Ctrl+Shift+F5

Test File Ctrl+F6

Debug Test File Ctrl+Shift+F6

Run Focused Test Method

Debug Focused Test Method

Run Into Method

New Watch... Ctrl+Shift+F7

Toggle Line Breakpoint Ctrl+F8



Library1 - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run

Search (Ctrl+I)

Projects Files Services

Library1

Source Packages

sub_business_tier

TFacade.java

TFactory.java

sub_business_tier.entities

TBook.java

TBook_period.java

TTitle_book.java

TTitle_book_on_tape.java

Libraries

Library1_client1_SE

Library1_client1-ejb

Library1_EJB1

Library1_interface1-ejb

Library1_JSE2

Source Packages

library1

TBook.java





4.6.5. The result

The screenshot shows the NetBeans IDE 8.0.2 interface. The main editor displays the `TBook_period.java` file, which contains the following code:

```
public void setPeriod(Date date) {  
    period = date;  
}  
  
@Override  
public boolean period_pass(Object data) {  
    Date date = TFactory.mdays((String) data);  
    return period.compareTo(date) < 0;  
}  
  
@Override  
public void startPeriod(Object data)  
{  
    Date help = TFactory.mdays((String) data);  
    setPeriod(help);  
}  
  
@Override  
public String toString() // your code here
```

The left sidebar shows the project structure for `Library1`, including source packages and libraries. The bottom status bar indicates that `TBook_period.java` has been saved.



4.7. Build and Run the Library1 Project

Library1 - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects Files Services

Library1

- Source Packages
 - sub_business_tier
 - TFacade.java
 - TFactory.java
 - sub_business_tier.entities
 - TBook.java
 - TBook_period.java
 - TTitle_book.java
 - TTitle_book_on_tape.java
- Libraries
 - Library1_client1_SE
 - Library1_client1-ejb
 - Library1_EJB1
 - Library1_interface1-ejb
 - Library1_JSE2
- Source Packages
 - library1
 - TBook.java
 - TBook_period.java
 - TFacade.java

Navigator

Members

TBook_period :: TBook

- getPeriod(): Date
- period_pass(Object data): boolean
- setPeriod(Date date)
- startPeriod(Object data)
- toString(): String
- period: Date

Source

```
10  @Override
11  public Date getPeriod() {
12      return period;
13  }
14
15  @Override
16  public void setPeriod(Date date) {
17      period = date;
18  }
19
20  @Override
21  public boolean period_pass(Object data) {
22      sub_business_tier.entities.TBook_period > period_pass >
```

Output

GlassFish Server 4.1 Library1 (run)

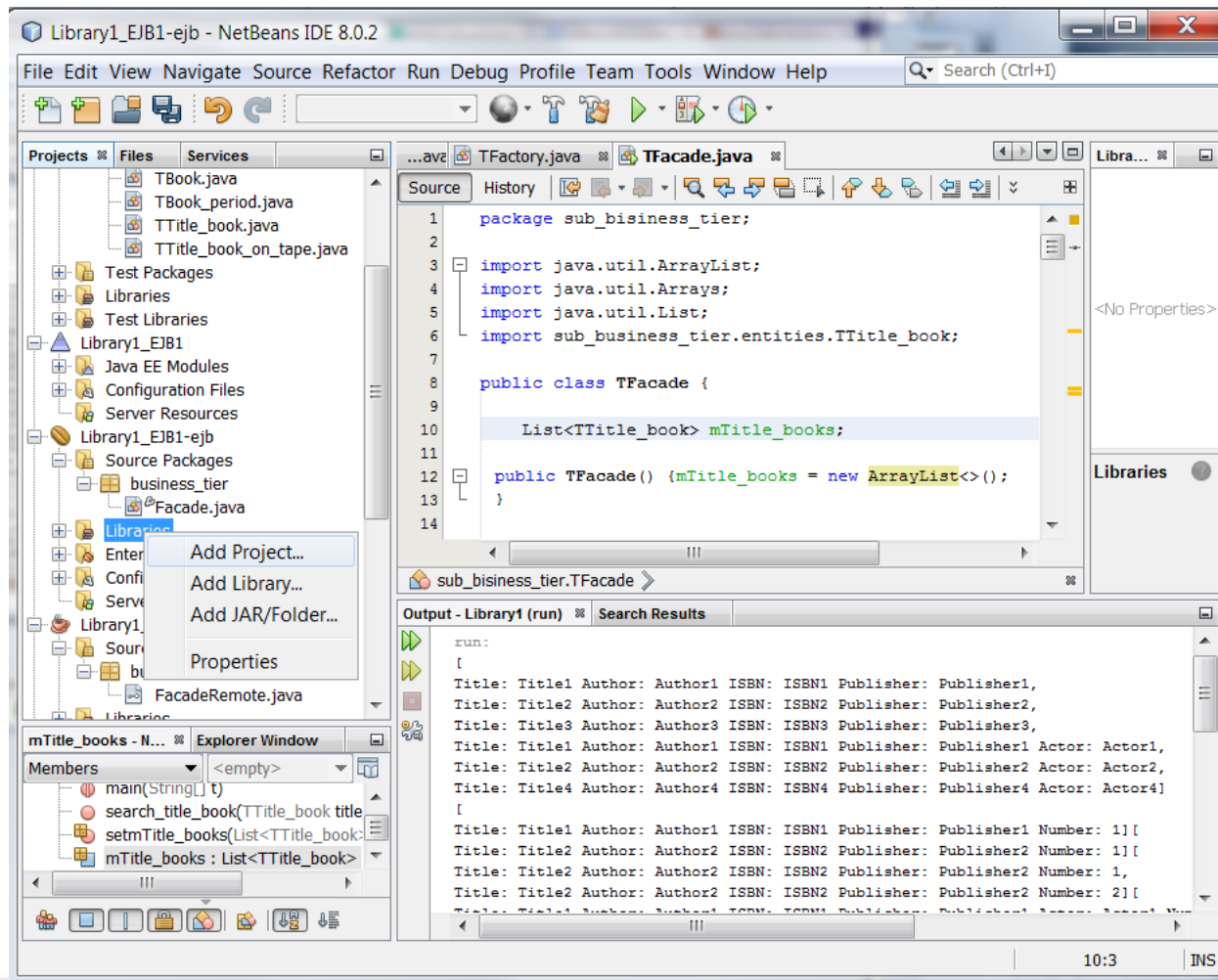
```
run:
[
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2,
Title: Title3 Author: Author3 ISBN: ISBN3 Publisher: Publisher3,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Actor: Actor2,
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4]
[
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Number: 1][
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 1][
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 2][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Sat Mar 14 18:52:46 CET 2015][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Sat Mar 14 18:52:46 CET 2015,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Tue Mar 10 18:52:46 CET 2015][
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2]

Titles of book
[Publisher1, ISBN1, Title1, Author1, ]
[Publisher2, ISBN2, Title2, Author2, ]
[Publisher3, ISBN3, Title3, Author3, ]
[Publisher1, ISBN1, Title1, Author1, Actor1]
[Publisher2, ISBN2, Title2, Author2, Actor2]
[Publisher4, ISBN4, Title4, Author4, Actor4]
```

20:14 INS

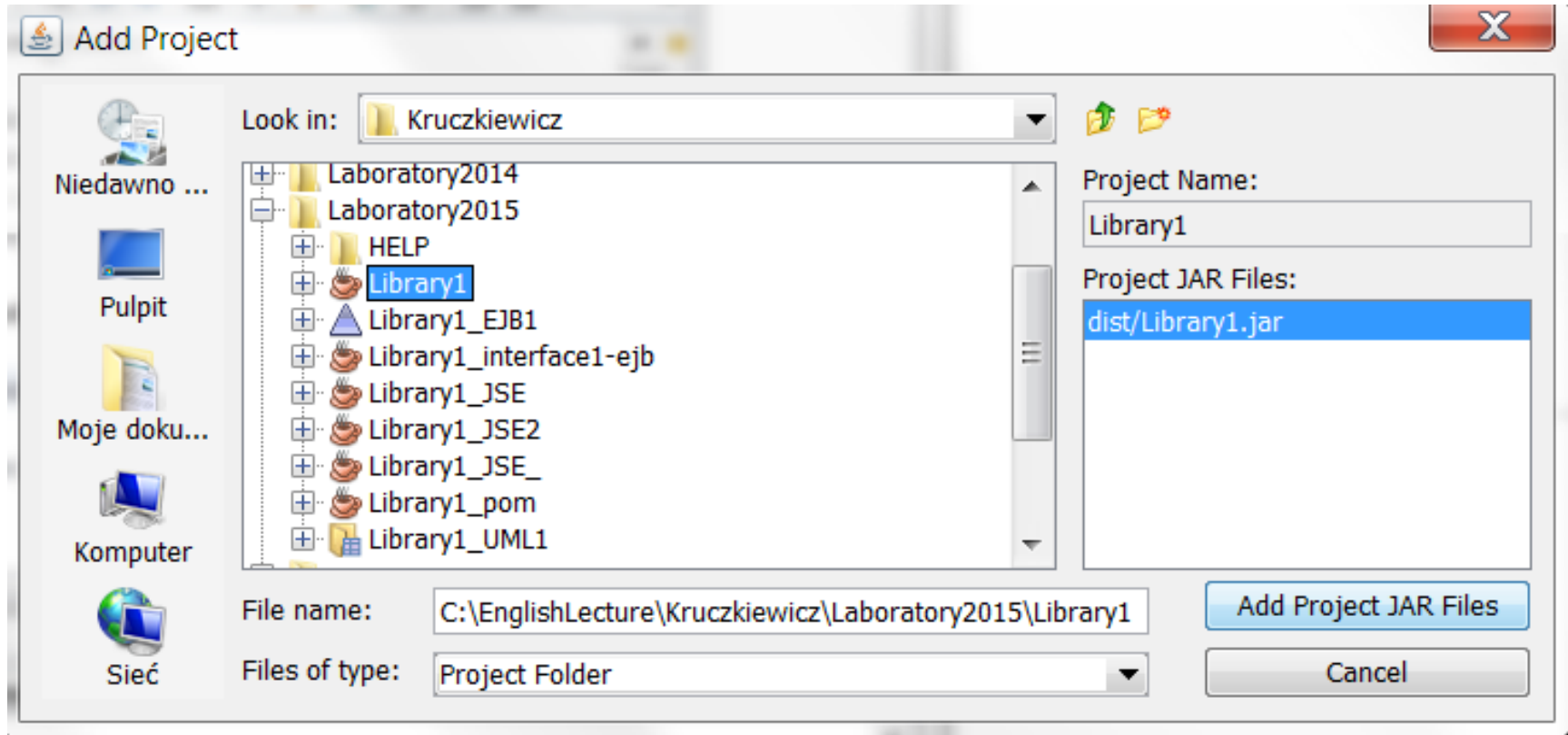


5./ 5.1. /5.1.1. Adding the library project (p.4) to the EJB module of EE project



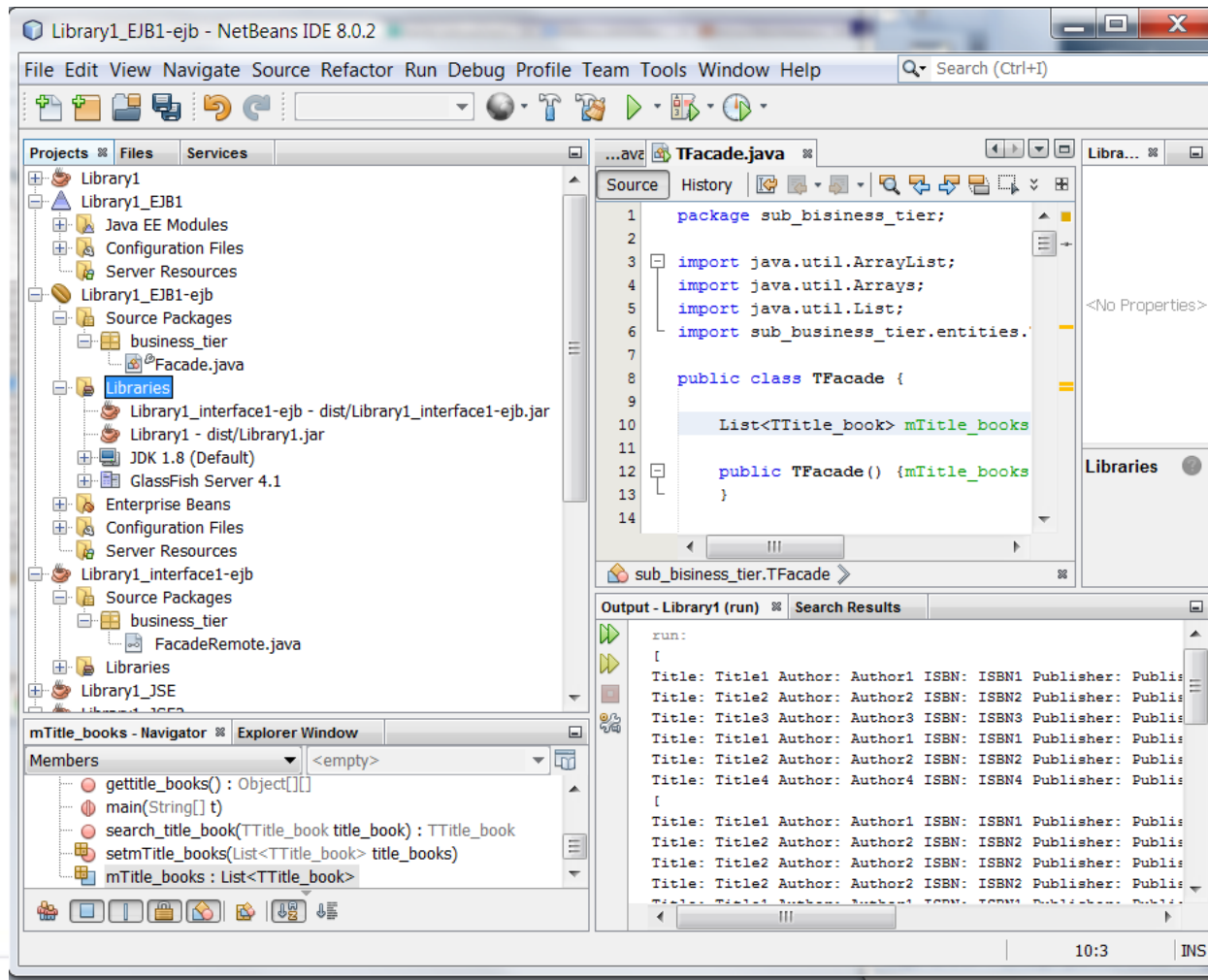


5.1.2. Adding the library project (p.4) to the EJB module of EE project - select the Library1 project and click the Add Project JAR Files





5.1.3. Adding the library project (p.4) to the EJB module of EE project p select the Library1 project - result





5.2. /5.2.1. Definition of interface of EJB Facade (as the adapter of methods of POJO TFacade)

The screenshot shows the NetBeans IDE 8.0.2 interface. The main editor window displays the `FacadeRemote.java` file, which defines the `FacadeRemote` interface. The interface includes the following methods:

```
package business_tier;

import javax.ejb.Remote;

/**
 * @author Zofia
 */
@Remote
public interface FacadeRemote {

    public Object[][] gettitle_books();

    public String add_title_book(String data[]);

    public ArrayList<String> add_book(String data1[], String data2[]);

    public ArrayList<String> Search_title_book(String data[]);

    public String Search_accessible_book(String data1[], Object data2);
}
```

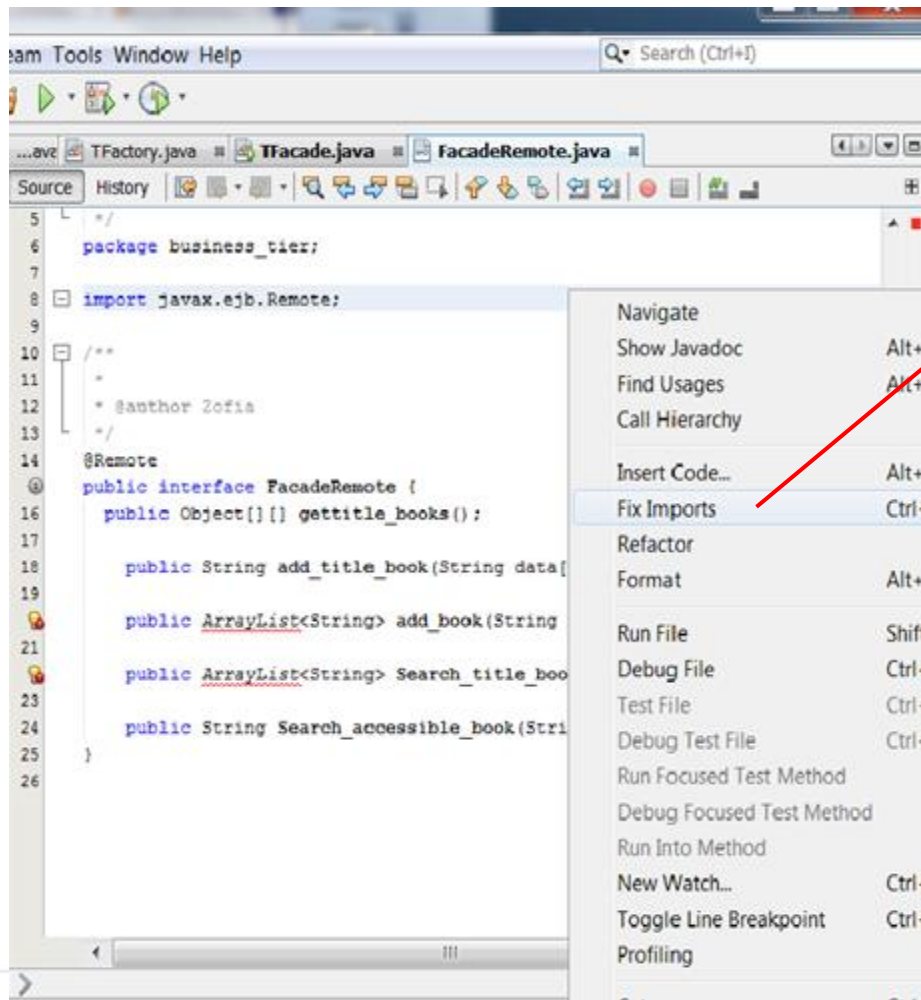
The left sidebar shows the project structure for `Library1`, including `Library1_EJB1`, `Library1_EJB1-ejb`, `Source Packages`, `business_tier`, `Libraries`, and `FacadeRemote.java`. The bottom left pane shows the `Search_accessible_book - Navigator` window, which lists the members of the `FacadeRemote` interface:

- `Search_accessible_book(String[] data1, Object data2) : String`
- `Search_title_book(String[] data) : <any>`
- `add_book(String[] data1, String[] data2) : <any>`
- `add_title_book(String[] data) : String`
- `gettitle_books() : Object[][]`

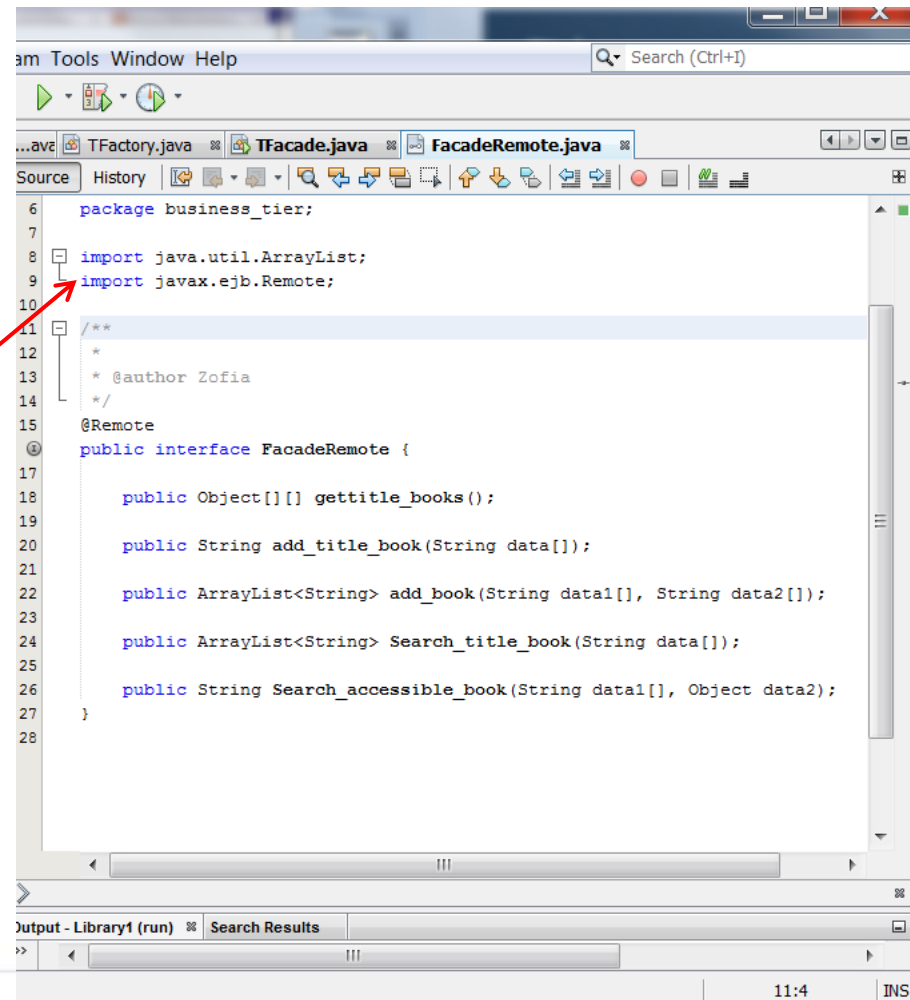
The bottom right pane shows the `Output - Library1 (run)` window, which is currently empty.



5.2.2. „Fix Imports” process



5.2.3. „Fix Imports” process - result





5.2.4 – Definition of the FacadeRemote interface - **you must add the declaration of these methods, which you have added during the development of the lab1 during the lab2**

```
package business_tier;

import java.util.ArrayList;
import javax.ejb.Remote;

@Remote
public interface FacadeRemote {

    public Object[][] gettitle_books();

    public String add_title_book(String data[]);

    public ArrayList<String> add_book(String data1[], String data2[]);

    public ArrayList<String> Search_title_book(String data[]);

    public String Search_accessible_book(String data1[], Object data2);
    // you must add the declaration of these methods, which you have added
    //during the development of the lab1 during the lab2
}
```



5.3. /5.3.1. Definition of EJB Facade as the Session Bean type attribute in the Library_EJB1 – ejb module (as the adapter of methods of the POJO TFacade class) – before Fix Imports

The screenshot displays the NetBeans IDE 8.0.2 interface. The left sidebar shows the project structure for 'Library1_EJB1-ejb'. The 'Library1_EJB1-ejb' module is expanded, showing the 'business_tier' package containing the 'Facade.java' file. The main editor window shows the source code of 'Facade.java'. The code defines a stateless EJB facade that implements the 'FacadeRemote' interface. It uses a 'TFacade' POJO class as an adapter. The code includes imports for 'javax.ejb.Stateless' and 'TFacade'. The facade methods delegate calls to the 'TFacade' instance.

```
1 ...5 lines
6 package business_tier;
7
8 import javax.ejb.Stateless;
9
10 @Stateless
11 public class Facade implements FacadeRemote {
12     TFacade facade = new TFacade();
13     @Override
14     public Object[][] gettitle_books() {
15         return facade.gettitle_books();
16     }
17     @Override
18     public String add_title_book(String data[]) {
19         return facade.add_title_book(data);
20     }
21     @Override
22     public ArrayList<String> add_book(String data1[], String data2[]) {
23         return facade.add_book(data1, data2);
24     }
25     @Override
26     public ArrayList<String> Search_title_book(String data[]) {
27         return facade.Search_title_book(data);
28     }
29     @Override
30     public String Search_accessible_book(String data1[], Object data2) {
31         return facade.Search_accessible_book(data1, data2);
32     }
33 }
```

The bottom status bar shows the time as 8:28 and the location as INS.



5.3.2. Definition of EJB Facade as the Session Bean type attribute in the Librrary_EJB1 – ejb module (as the adapter of methods of the POJO TFacade class) - after Fix Imports

The screenshot displays the NetBeans IDE 8.0.2 interface. The left sidebar shows the project structure for 'Library1_EJB1-ejb'. The 'Sources' tab is selected, showing the 'business_tier' package with the 'Facade.java' file. The 'Facade.java' file is open in the editor, showing the following code:

```
1 ...5 lines
2
3 package business_tier;
4
5
6 import java.util.ArrayList;
7 import javax.ejb.Stateless;
8 import sub_business_tier.TFacade;
9
10 @Stateless
11 public class Facade implements FacadeRemote {
12     TFacade facade = new TFacade();
13     @Override
14     public Object[][] gettitle_books() {
15         return facade.gettitle_books();
16     }
17     @Override
18     public String add_title_book(String data[]) {
19         return facade.add_title_book(data);
20     }
21     @Override
22     public ArrayList<String> add_book(String data1[], String data2[]) {
23         return facade.add_book(data1, data2);
24     }
25     @Override
26     public ArrayList<String> Search_title_book(String data[]) {
27         return facade.Search_title_book(data);
28     }
29     @Override
30     public String Search_accessible_book(String data1[], Object data2) {
31         return facade.Search_accessible_book(data1, data2);
32     }
33 }
```

The 'FacadeRemote' interface is visible in the 'Libraries' tab. The 'Facade' class is a stateless session bean that implements the 'FacadeRemote' interface. It uses the 'TFacade' class as a facade for the underlying business logic.

The bottom status bar shows 'Facade.java saved.' and the time '13:46'.



```
package business_tier;
```

```
import java.util.ArrayList;  
import javax.ejb.Stateless;  
import sub_business_tier.TFacade;
```

```
@Stateless  
public class Facade implements FacadeRemote {
```

```
    TFacade facade = new TFacade();
```

```
    @Override  
    public Object[][] gettitle_books() {  
        return facade.gettitle_books();    }
```

```
    @Override  
    public String add_title_book(String data[]) {  
        return facade.add_title_book(data);    }
```

```
    @Override  
    public ArrayList<String> add_book(String data1[], String data2[]) {  
        return facade.add_book(data1, data2);    }
```

```
    @Override  
    public ArrayList<String> Search_title_book(String data[]) {  
        return facade.Search_title_book(data);    }
```

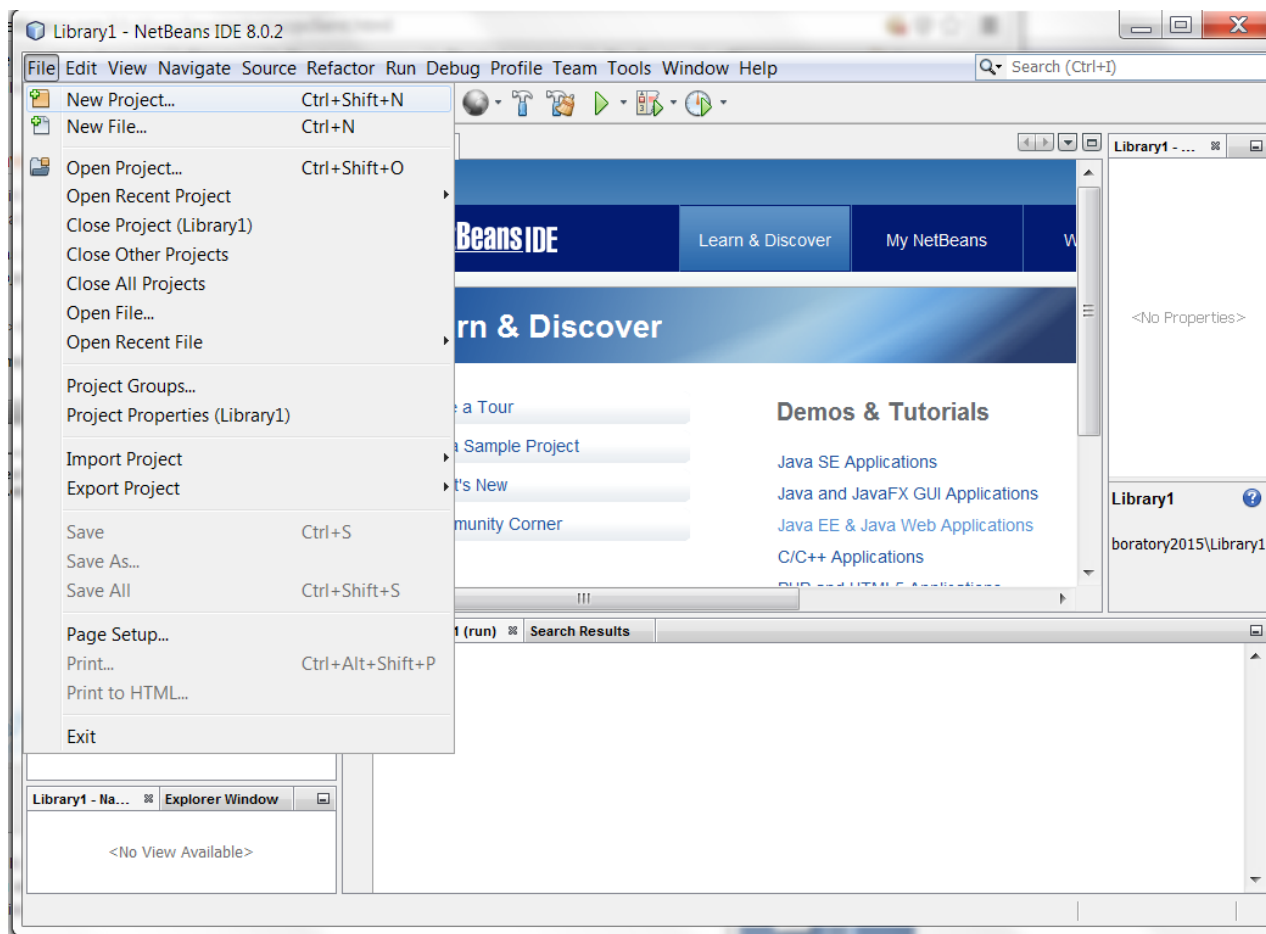
```
    @Override  
    public String Search_accessible_book(String data1[], Object data2) {  
        return facade.Search_accessible_book(data1, data2);    }
```

```
}
```

5.3.3. The implementation of FacadeRemote interface as the Facade SessionBean - // you must add the implementation of these methods, which you have added during the development of the lab1 during the lab2 and declared in the FacadeRemote interface.

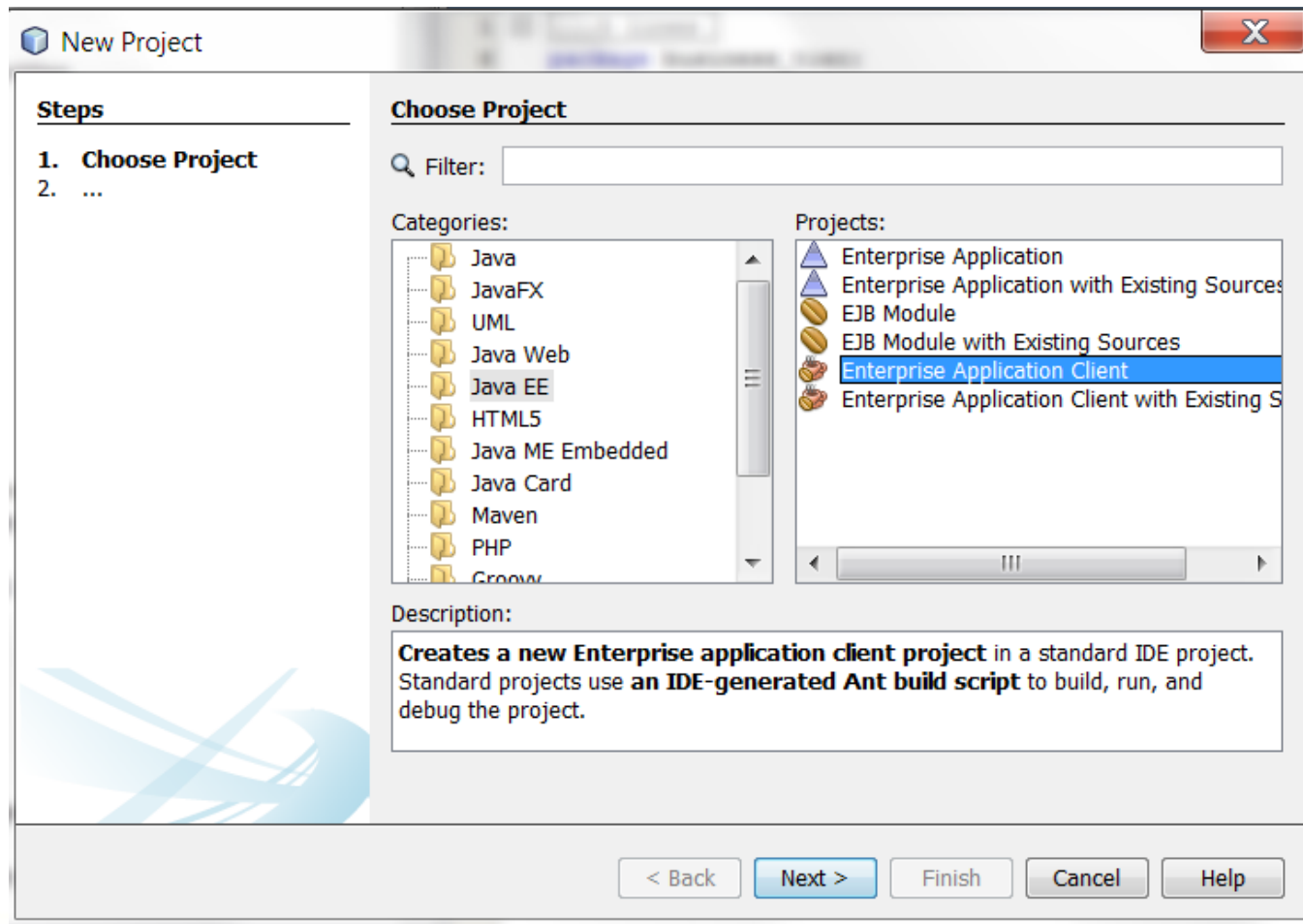


6. /6.1. /6.1.1. Creation of the EE client tier – click File/New Project...





6.1.2. Creation of the EE client tier – select the Java EE/ Enterprise Application Client items. Click Next.





6.1.3. Creation of the EE client tier – fill the project name and select the project location. Click Next.

New Enterprise Application Client

Steps

1. Choose Project
- 2. Name and Location**
3. Server and Settings

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).



6.1.4. Creation of the EE client tier – select the server, Java EE version and fill the name of main class. Click Finish.

New Enterprise Application Client

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**

Server and Settings

Add to Enterprise Application: <None>

Server: GlassFish Server 4.1 Add...

Java EE Version: Java EE 7

Main Class: client_tier.Client

< Back Next > Finish Cancel Help



6.2. / 6.2.1. Definition of code of the EE client based on code of the SE client – only change of TFacade class from POJO SE TFacade class into the EJB Facade class.

Creation of connection from the Facade Session Bean of Library1_EJB1-ejb module to the Client class of the Library1_client1-ejb project – expand the Source Packages node of the Library1_client1-ejb project and open Client.java in the editor. Right-click in the source code and choose Insert Code... item and select Call Enterprise Bean to open the Call Enterprise Bean dialog.

The screenshot shows the NetBeans IDE 8.0.2 interface. The 'Library1_client1-ejb' project is open. The 'Source Packages' node is expanded, showing the 'client_tier' package. The 'Client.java' file is open in the editor. The code in the editor is as follows:

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package client_tier;
7
8  /**
9   *
10   * @author Zofia
11   */
12  public class Client {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20  }
21
22
```

A right-click context menu is open over the code, with the following options:

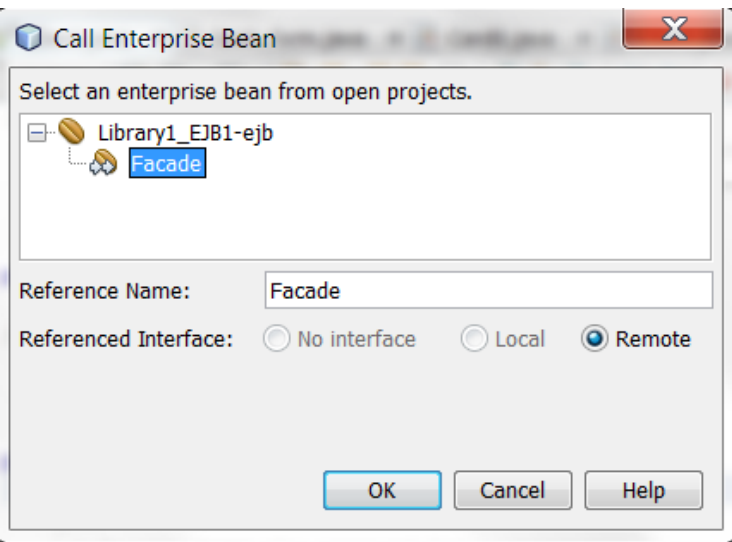
- Navigate
- Show Javadoc Alt+F1
- Find Usages Alt+F7
- Call Hierarchy
- Insert Code... Alt+Insert**
- Fix Imports Ctrl+Shift+I
- Refactor
- Format Alt+Shift+F
- Run File Shift+F6
- Debug File Ctrl+Shift+F5
- Test File Ctrl+F6
- Debug Test File Ctrl+Shift+F6
- Run Focused Test Method
- Debug Focused Test Method
- Run Into Method
- New Watch... Ctrl+Shift+F7
- Toggle Line Breakpoint Ctrl+F8
- Profiling
- Cut Ctrl+X
- Copy Ctrl+C

The 'Insert Code...' option is selected, and a secondary 'Generate' menu is open, showing the following options:

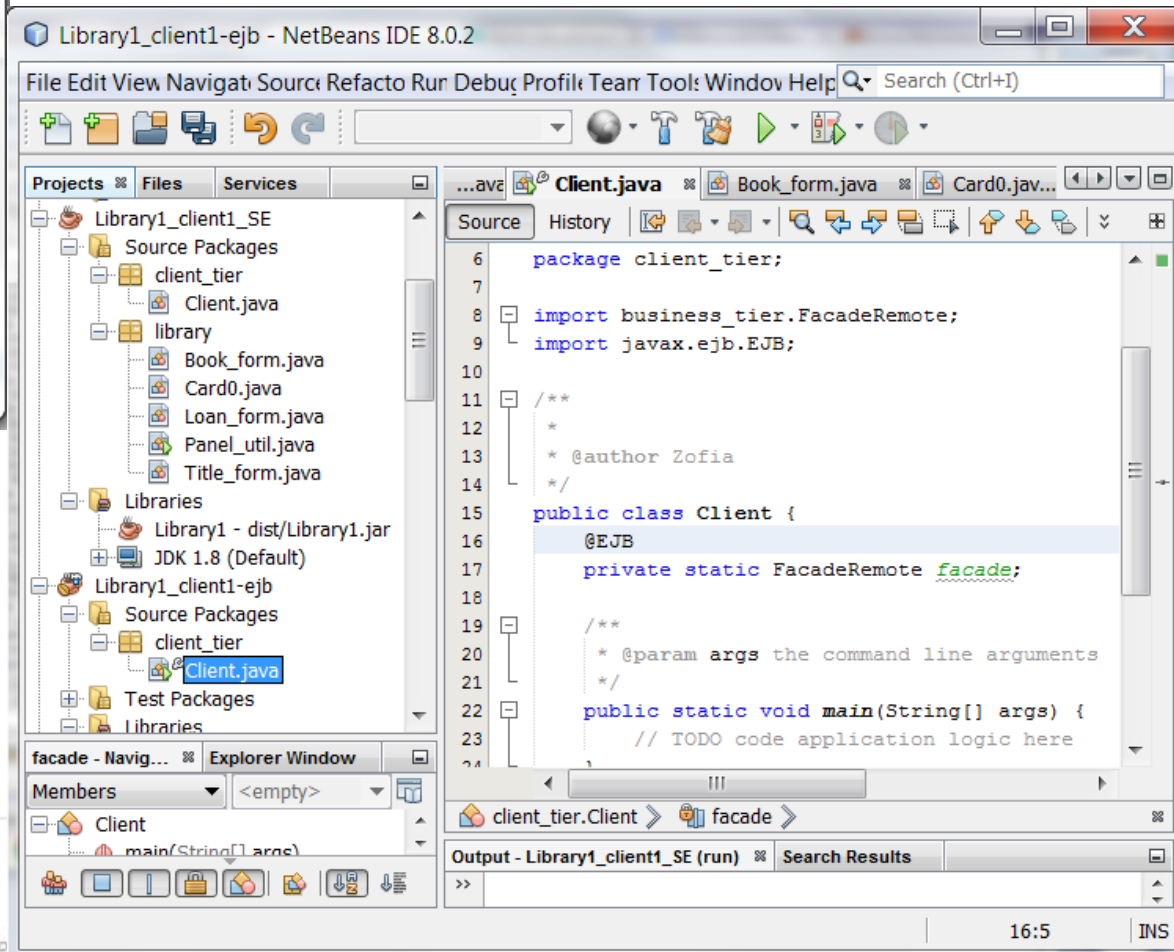
- Generate
- Constructor...
- Logger...
- toString()...
- Override Method...
- Add Property...
- Call Enterprise Bean...**
- Use Database...
- Send JMS Message...
- Send E-mail...
- Call Web Service Operation...
- Generate REST Client...



6.2.2. Creation of connection from the Facade Session Bean of Library1_EJB1-ejb module to the Client class of the Library_client1-ejb project - select the Facade EJB from the Library1_EJB1-ejb module of Library1_EJB1 project

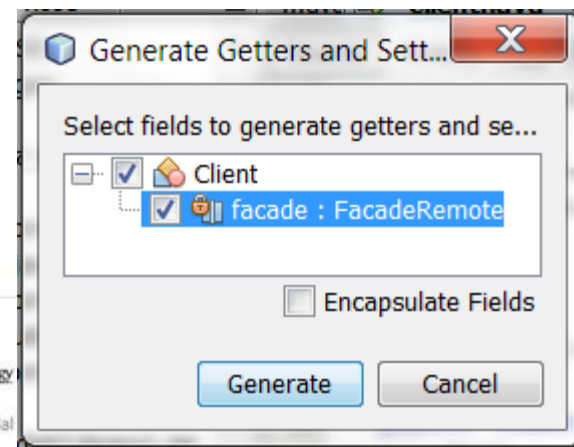
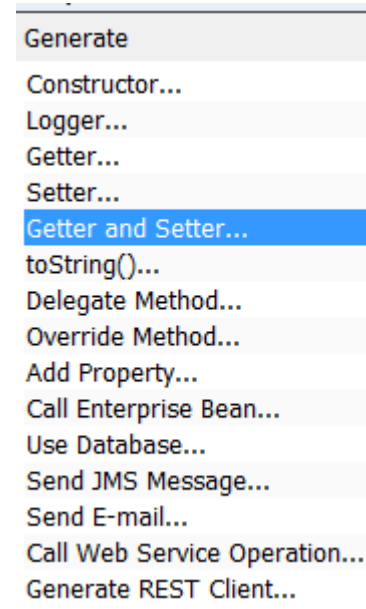
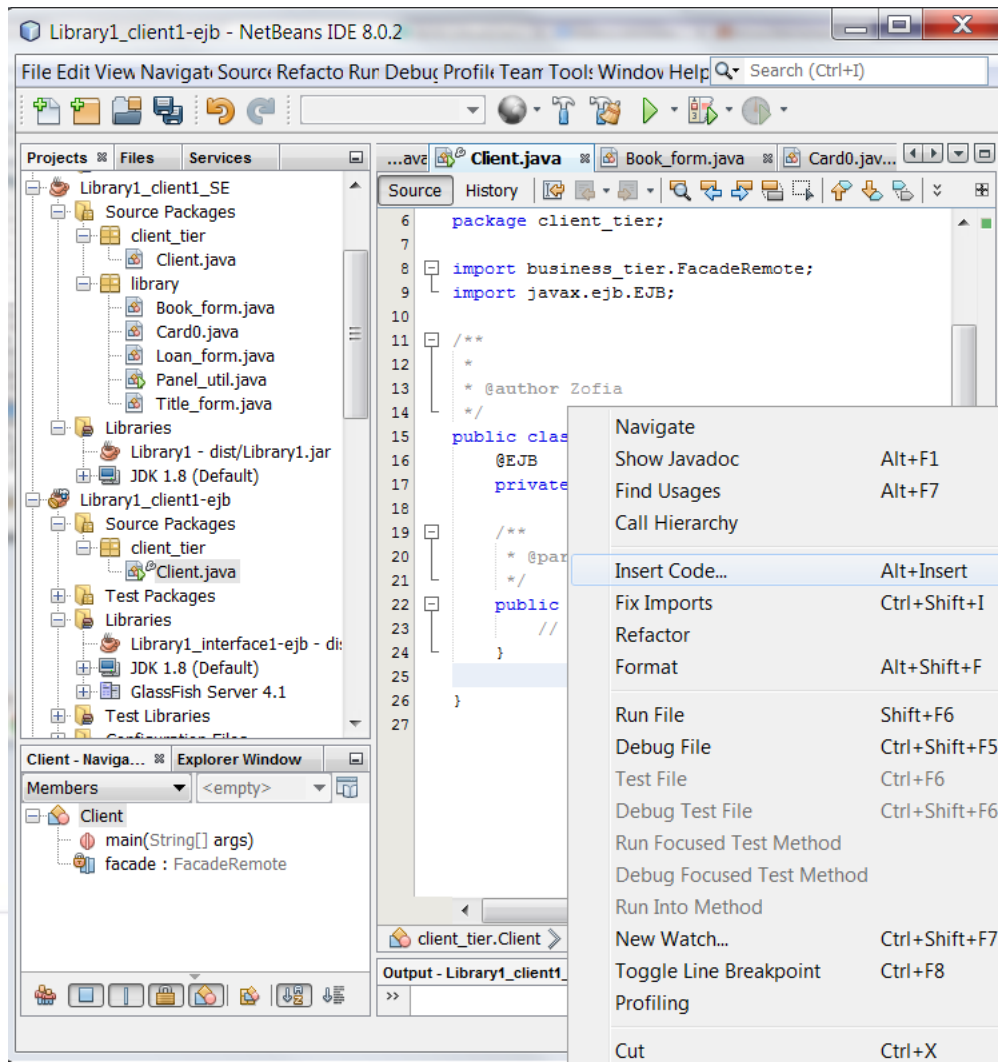


6.2.3. The result of creation the connection to the Facade SessionBean of the Library1_EJB1-ejb module.



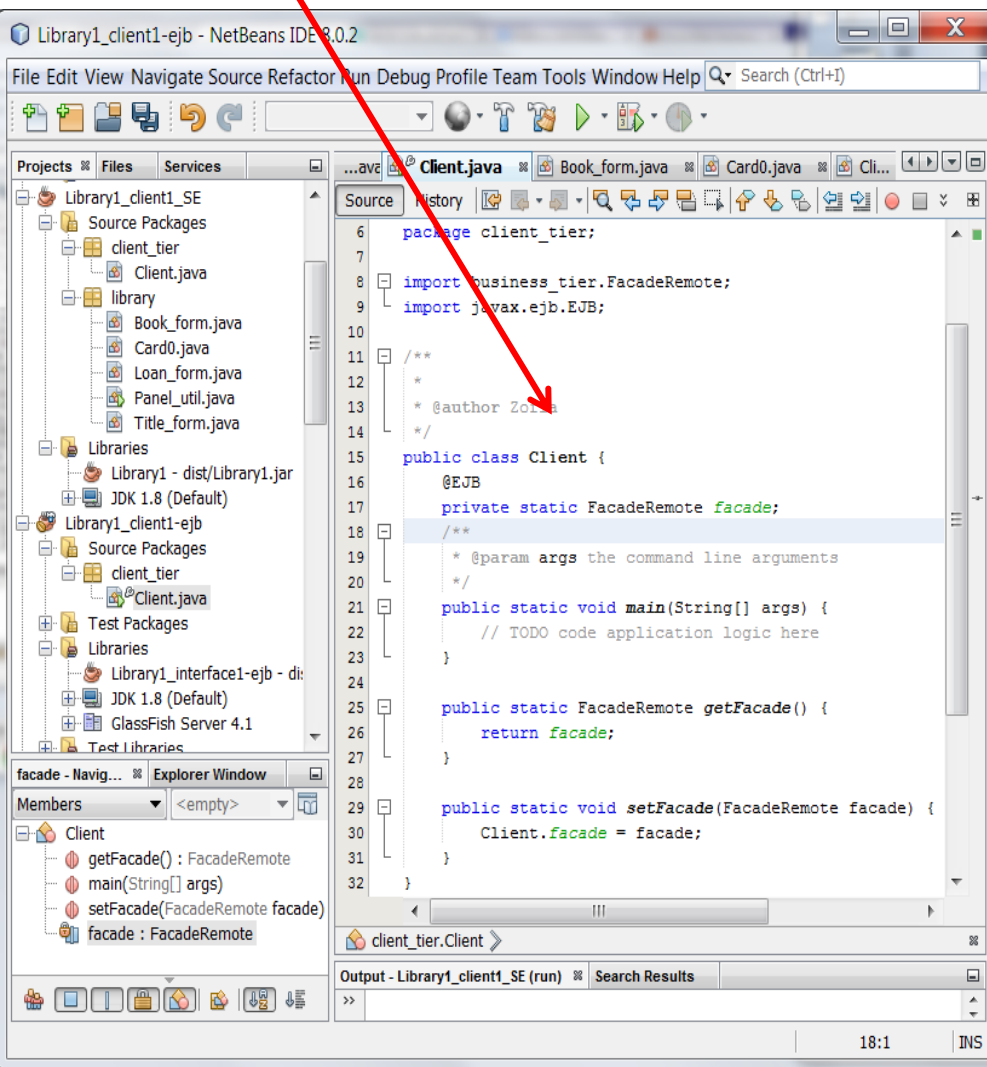


6.3. /6.3.1. Adding the getter and setter methods to the facade attribute as the reference to the Facade SessionBean by using the „Insert Code” process – select Insert Code/Getter and Setter... items to open the Generate Getters and Setters dialog and select the facade attribute





6.3.2. The result



6.3.3. Define the body of main method based on code of the Client class main method from the Standard Edition client project named Library1_client1_SE

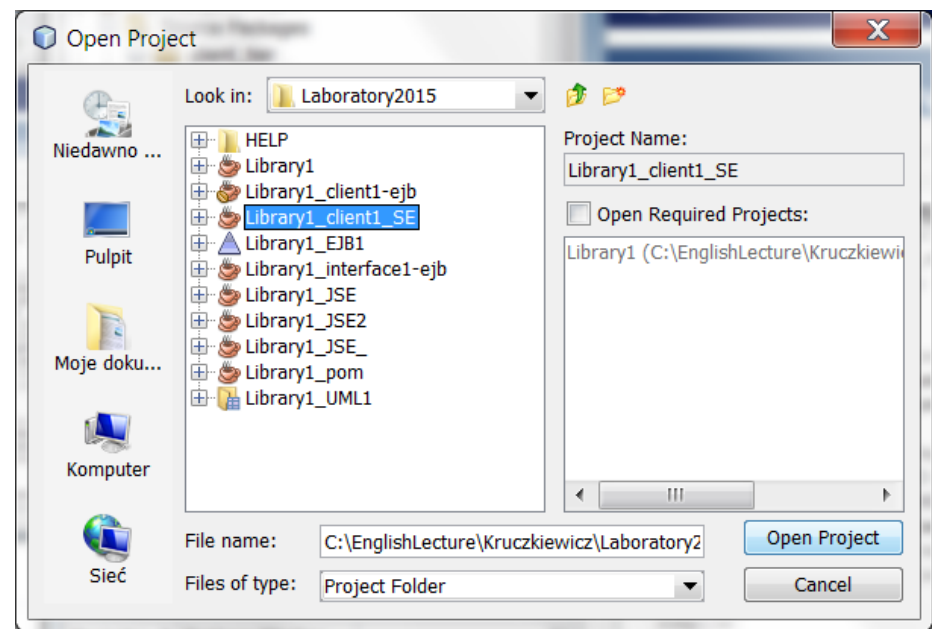
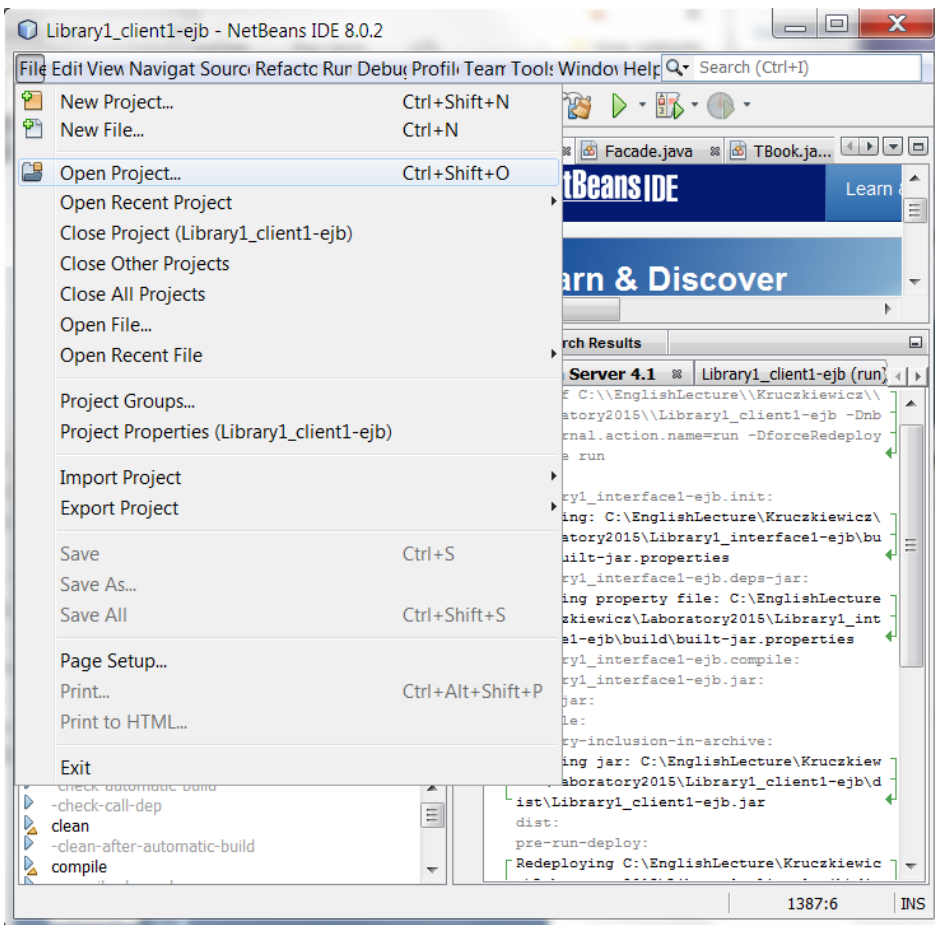
```

public static void main(String[] args) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            Panel_util.createAndShowGUI();
        }
    });
}

```

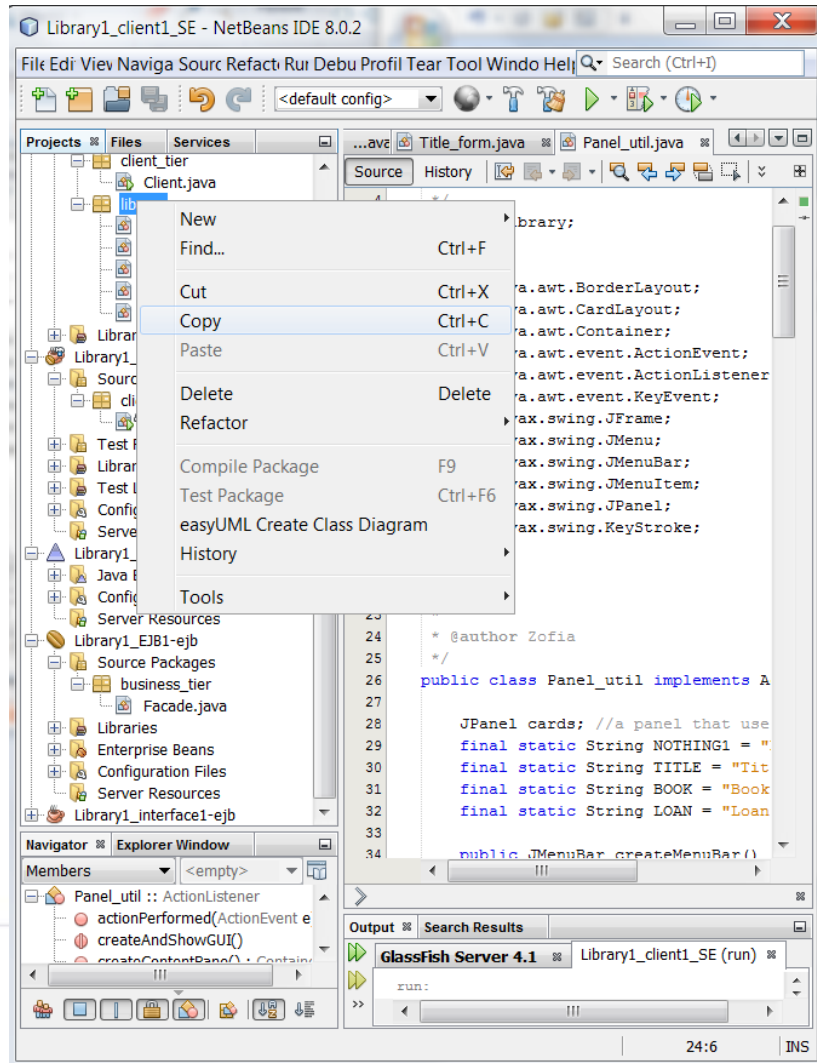



6.4. /6.4.1. Coping the library package from the Standard Edition Application Client project as the Library1_client1_SE one (attachment to the lab3)

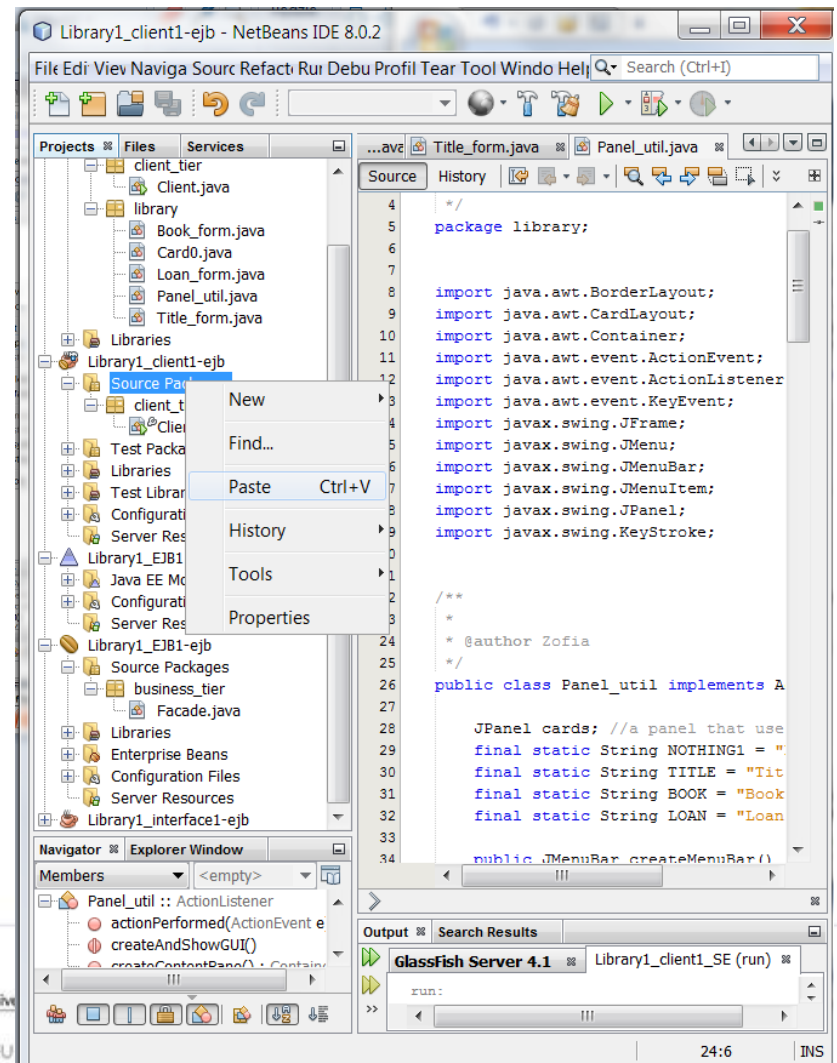




6.4.2. Coping the library package from the Standard Edition Application Client project as the Library1_client1_SE one

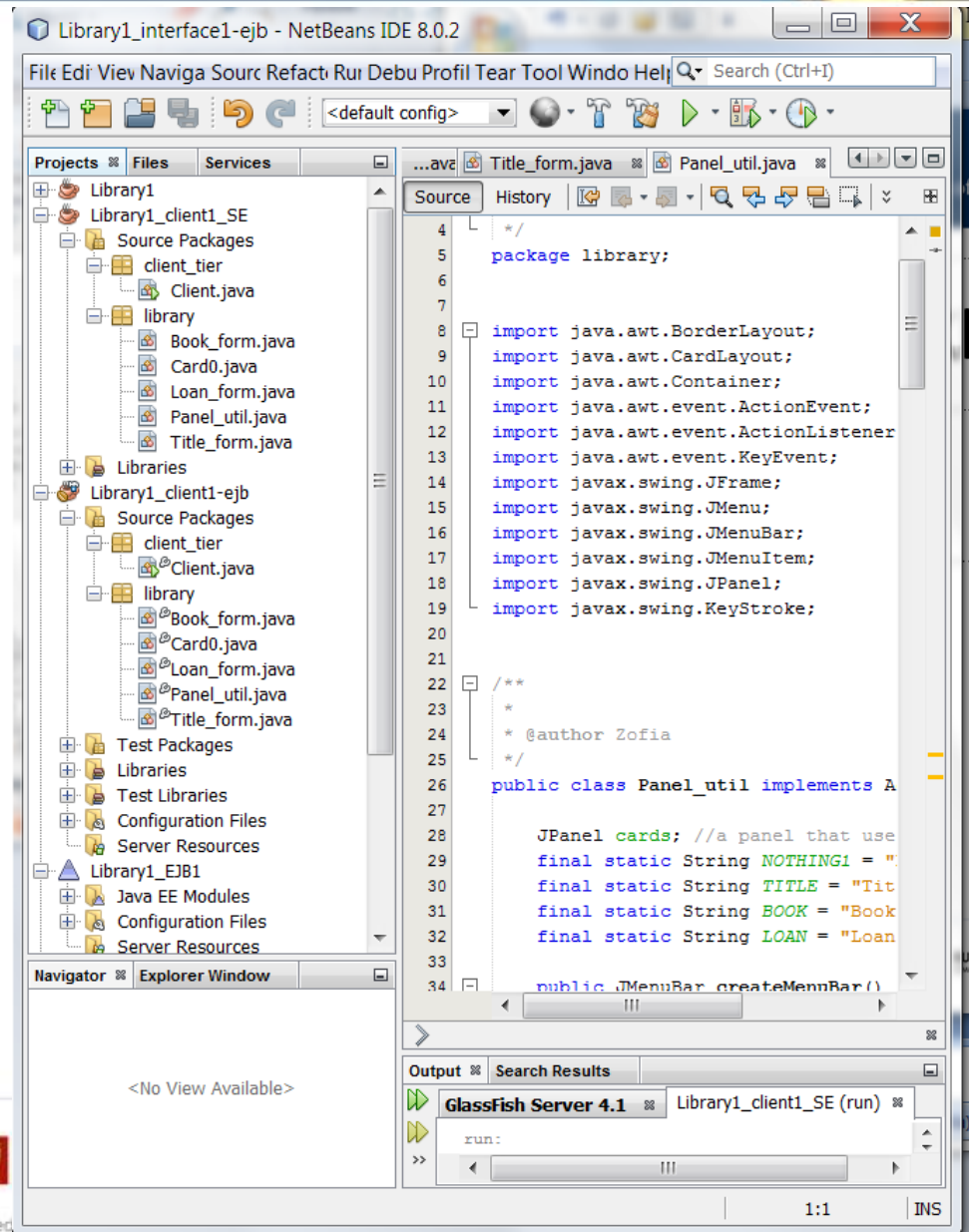


6.4.3. Paste the library package from the Library1_client1_SE to the Library1_client1-ejb type Enterprise Application Client project





**6.4.4. The result – creation the
Library1_client1-ejb type the Enterprise
Application Client project based on the
code of the Standard Edition version of the
client project**





6.5. The change of getter and setter methods of the Facade class as the Session Bean type attribute

```
package client_tier;

import library.Panel_util;
import sub_business_tier.TFacade;

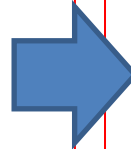
public class Client {

    static TFacade facade = new TFacade();

    static public TFacade getFacade() {
        return facade;
    }

    static public void setFacade(TFacade facade) {
        Client.facade = facade;
    }

    public static void main(String[] args) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                Panel_util.createAndShowGUI();
            }
        });
    }
}
```



```
package client_tier;

import business_tier.FacadeRemote;
import javax.ejb.EJB;
import library.Panel_util;

public class Client {
    @EJB
    private static FacadeRemote facade;

    public static FacadeRemote getFacade() {
        return facade;
    }

    public static void setFacade(FacadeRemote facade) {
        Client.facade = facade;
    }

    public static void main(String[] args) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                Panel_util.createAndShowGUI();
            }
        });
    }
}
```





6.6. The same code – different type of return value from client.getFacade()

```
public void actionPerformed(ActionEvent evt) {  
    String[] data = form_title();  
    if (data == null) {  
        return;  
    }  
    Client.getFacade().add_title_book(data);  
}
```

```
public void actionPerformed(ActionEvent evt) {  
    String[] data = form_title();  
    if (data == null) {  
        return;  
    }  
    Client.getFacade().add_title_book(data);  
}
```

Object POJO in SE
Client tier

EJB in EE Client
tier





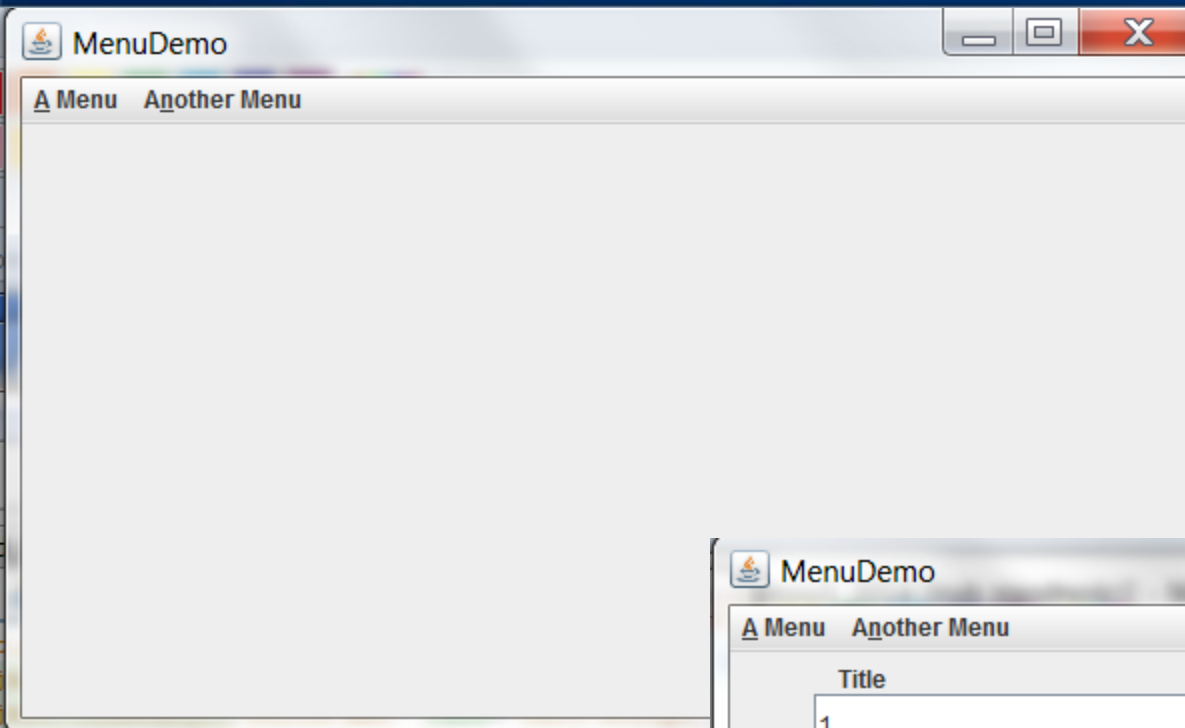
6.7. You must add the new panels with forms for supported the new functions (the reservation or the loan of books)

- 1. At first, you must add the new panels with forms for supported the new functions (the reservation or the loan of books) in the Java Application projects (as Library1_client1_SE).**
- 2. Secondly, you must transform this project into the Enterprise Application Client project, accordingly the instruction of the lab3 (50-62 slides)**



7.0. Running the program

1. After this development, your program will be executed properly, if you clean and build the following programs:
 1. Library1
 2. Library_interface-ejb
 3. Library_EJB1-ejb
 4. Library_client-ejb
2. Then you must deploy the Library_EJB1 program.
3. Finally, you may run a few instances of Library_client-ejb programs - these programs share the common data as titles and books.
4. In the Service Tab you may see, if your EE program deploy properly (Server item). The other useful information you may get from the Glassfish output window tab.



7.1. The view of the Enterprise Application Client (**Library_client-ejb**) for processing of application data - with the same responsibilities as of the version of third laboratory.

MenuDemo

A Menu Another Menu

Title
1

Author
1

ISBN
1

Publisher
1

Actor

Add title





7.2. The **Library_client-ejb** form to adding the new books of the selected title, as the application data.

The screenshot shows a Java Swing window titled "MenuDemo". It contains a menu bar with "A Menu" and "Another Menu". Below the menu bar is a table with 5 columns: Publisher, ISBN, Title, Author, and Actor. The table has two rows, both containing the value "1". Below the table are three text input fields labeled "Number of a book", "Period of a book", and "Add book". Below these fields is a section titled "Books" containing a list of book entries. The first entry is "Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Sun Feb 22 22:00:43 CET 2015" with a dropdown arrow. The second entry is "Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Sun Feb 22 22:00:43 CET 2015". The third entry is "Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 5".

Publisher	ISBN	Title	Author	Actor
1	1	1	1	
1	1	1	1	1

Number of a book

Period of a book

Add book

Books

Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Sun Feb 22 22:00:43 CET 2015 ▼

Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Sun Feb 22 22:00:43 CET 2015

Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 5



7.3. Restored data from database, after again opening EE application with the one kind of client:
Library_client-ejb as the Enterprise Application client (below)

The screenshot shows a Java Swing window titled "MenuDemo". It contains a menu bar with "Menu" and "Another Menu". Below the menu bar is a table with 5 columns: Publisher, ISBN, Title, Author, and Actor. The table has two rows of data, both with the value "1" in each column. Below the table are three input fields labeled "Number of a book", "Period of a book", and "Add book". At the bottom, there is a section titled "Books" containing a list of book entries. The first two entries are identical: "Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Sun Feb 22 22:00:43 CET 2015". The third entry is "Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 5".

Publisher	ISBN	Title	Author	Actor
1	1	1	1	
1	1	1	1	1

Number of a book

Period of a book

Add book

Books

- Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Sun Feb 22 22:00:43 CET 2015
- Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Sun Feb 22 22:00:43 CET 2015
- Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 5



8. /8.1. The help information about development of Java EE projects

The screenshot shows the NetBeans Help window. The left pane displays a search results list for 'updating Java EE projects'. The right pane shows the content of the selected topic, 'Using Java EE'.

Contents **Search**

Znajdź: updating Java EE projects

- 3 persistence.xml Visual Editor
- 1 Free-Form Project Properties Dialog Box: Web Sources
- 2 New Enterprise Application with Existing Sources: Application Modules
- 6 New Web Application Wizard: Server and Settings
- 5 New Enterprise Application Client Wizard: Server and Settings
- 5 New EJB Module Wizard: Server and Settings
- 5 New Enterprise Application Wizard: Server and Settings
- 5 New Web Project with Existing Sources Wizard: Server and Settings
- 4 New Enterprise Application Wizard with Existing Sources: Server and Settings
- 2 New Session Bean Wizard
- 2 Project Properties: Maven Project
- 2 New Free-Form Project Wizard: Web Sources
- 2 Standard Web Project Properties Dialog Box: Run
- 2 **Using Java EE**
- 2 Java Projects
- 1 New Message-Driven Bean Wizard: Name and Location
- 1 Create Java SE Embedded JRE
- 1 Add Message Destination Dialog Box
- 1 New Enterprise Application Client Wizard: Name and Location
- 1 New Web Application Wizard: Name and Location
- 1 New Entity Classes from Database Wizard: Database Tables
- 1 New Message-Driven Bean Wizard: Activation Config Properties
- 1 New Enterprise Application Wizard: Name and Location
- 1 New EJB Module Wizard: Name and Location
- 1 Add Web Service Dialog Box
- 1 New EJB Module with Existing Sources Wizard: Existing Sources and Libraries
- 1 Standard EJB Module Project Properties Dialog Box: Run
- 1 New Enterprise Application with Existing Sources: Name and Location
- 1 Standard Web Project Properties Dialog Box: Disabled JAX-RPC Web Services
- 1 Standard Web Project Properties Dialog Box: Disabled JAX-RPC Web Service Clients
- 2 Projects Window
- 2 Java Platform Manager Dialog Box
- 1 Java FX Project Properties: Deployment
- 1 Import Wireless Toolkit Project Wizard: Specify Wireless Toolkit Project Page
- 1 Java ME Embedded Project Properties Dialog Box: Libraries
- 1 Standard Java SE Project Properties Dialog Box: Libraries
- 1 Java ME Embedded Project Properties Dialog Box: Libraries
- 1 Java ME Embedded Project Properties Dialog Box: Libraries

Using Java EE

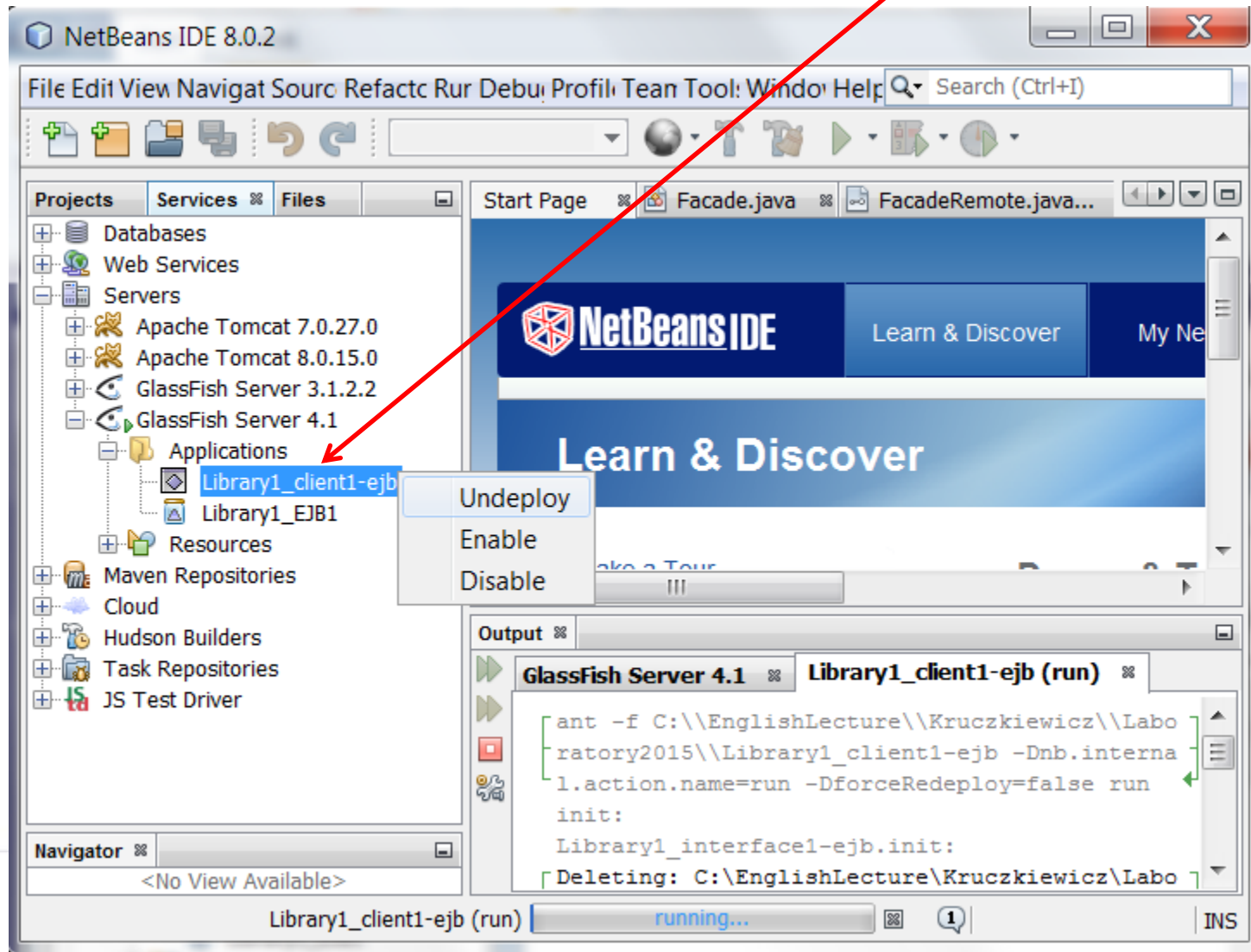
For information on working with Java Enterprise Edition (Java EE) in the IDE, see the following chapter in the User's Guide:

Developing Applications with NetBeans IDE, "Developing Enterprise Applications"

[Copyright © 2014, Oracle and/or its affiliates. All rights reserved.](#)



8.2. The closing or before the update of Enterprise application – after undeploy process of of EE project components





8.3. The result after undeploying process of EE project components

