

# Internet Engineering

**Tomasz Babczyński, Zofia Kruczkiewicz**  
Tomasz Kubik

**Information systems modelling– UML and  
service description languages**  
Laboratory 3

*Choose yourself and new technologies*



**HUMAN CAPITAL**  
HUMAN – BEST INVESTMENT!



Wrocław University of Technology

EUROPEAN  
SOCIAL FUND

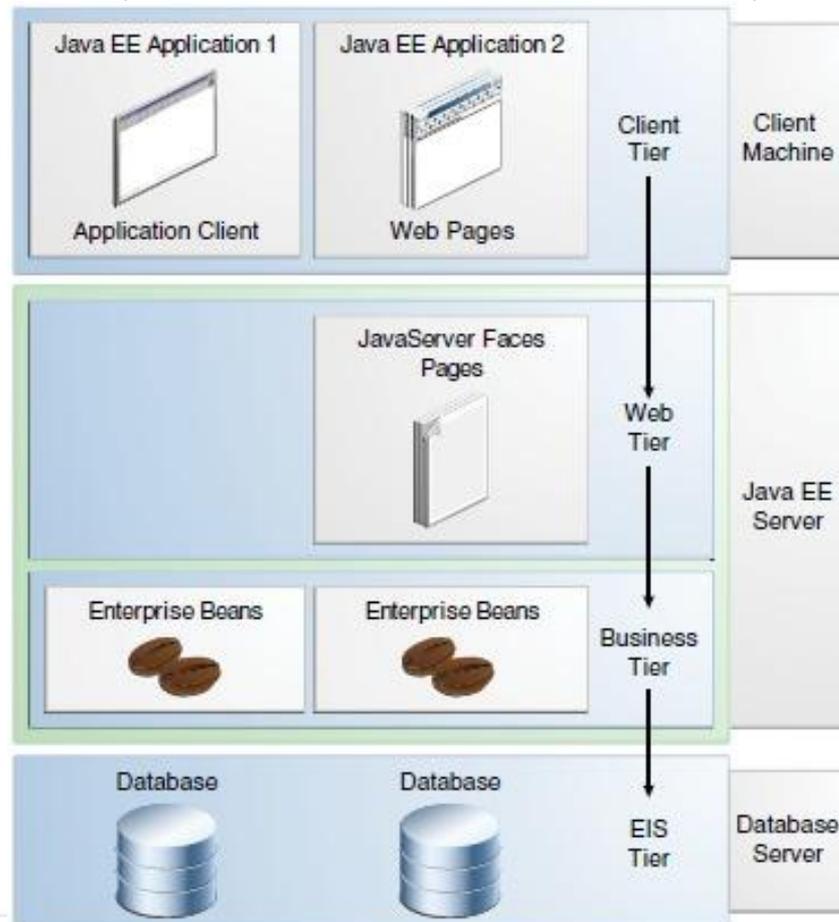


Project co-financed from the EU European Social Fund



# 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)

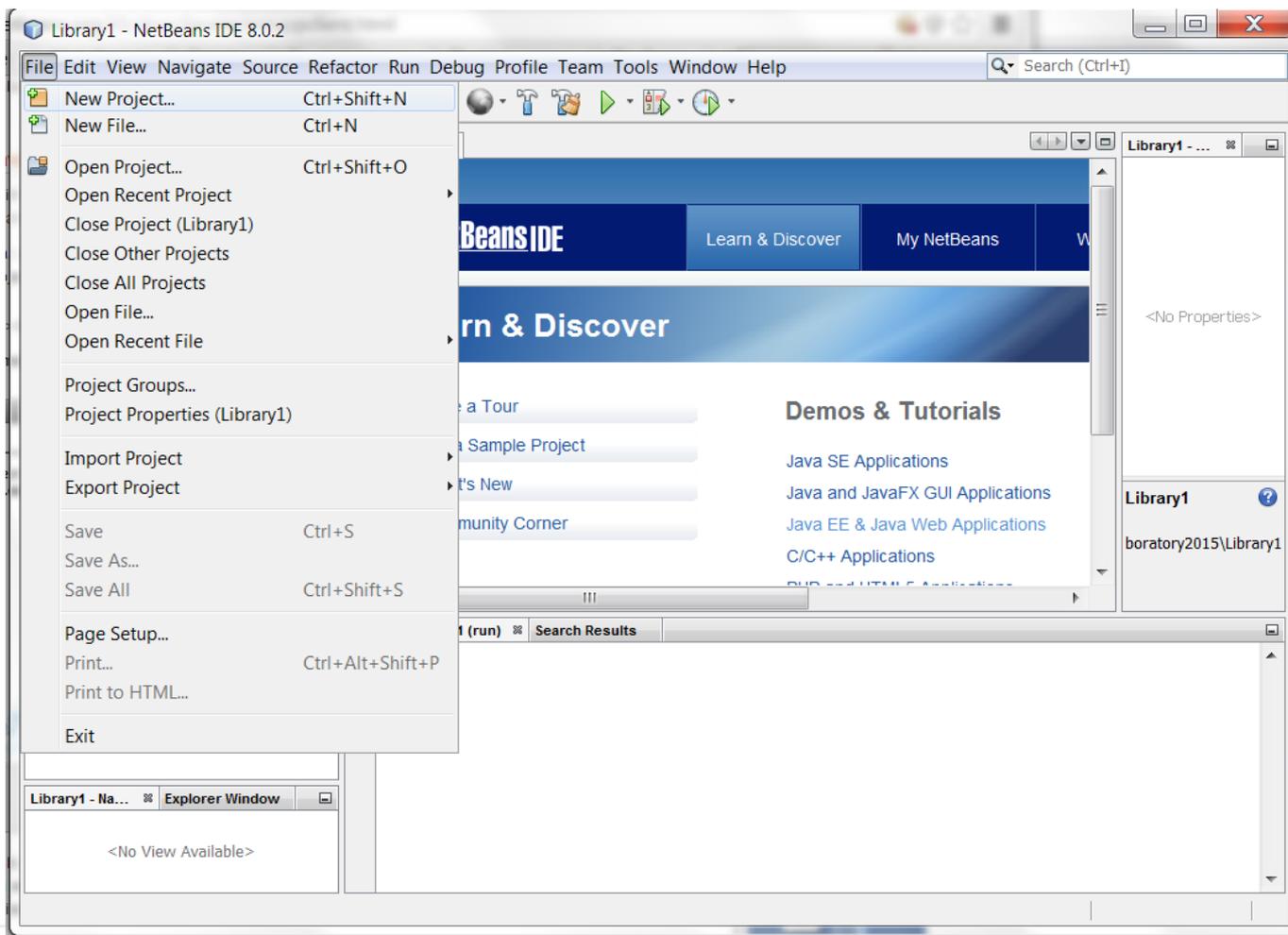
<https://netbeans.org/kb/docs/javaee/entappclient.html>

from

[https://netbeans.org/kb/trails/java-ee.html?utm\\_source=netbeans&utm\\_campaign=welcomepage](https://netbeans.org/kb/trails/java-ee.html?utm_source=netbeans&utm_campaign=welcomepage)

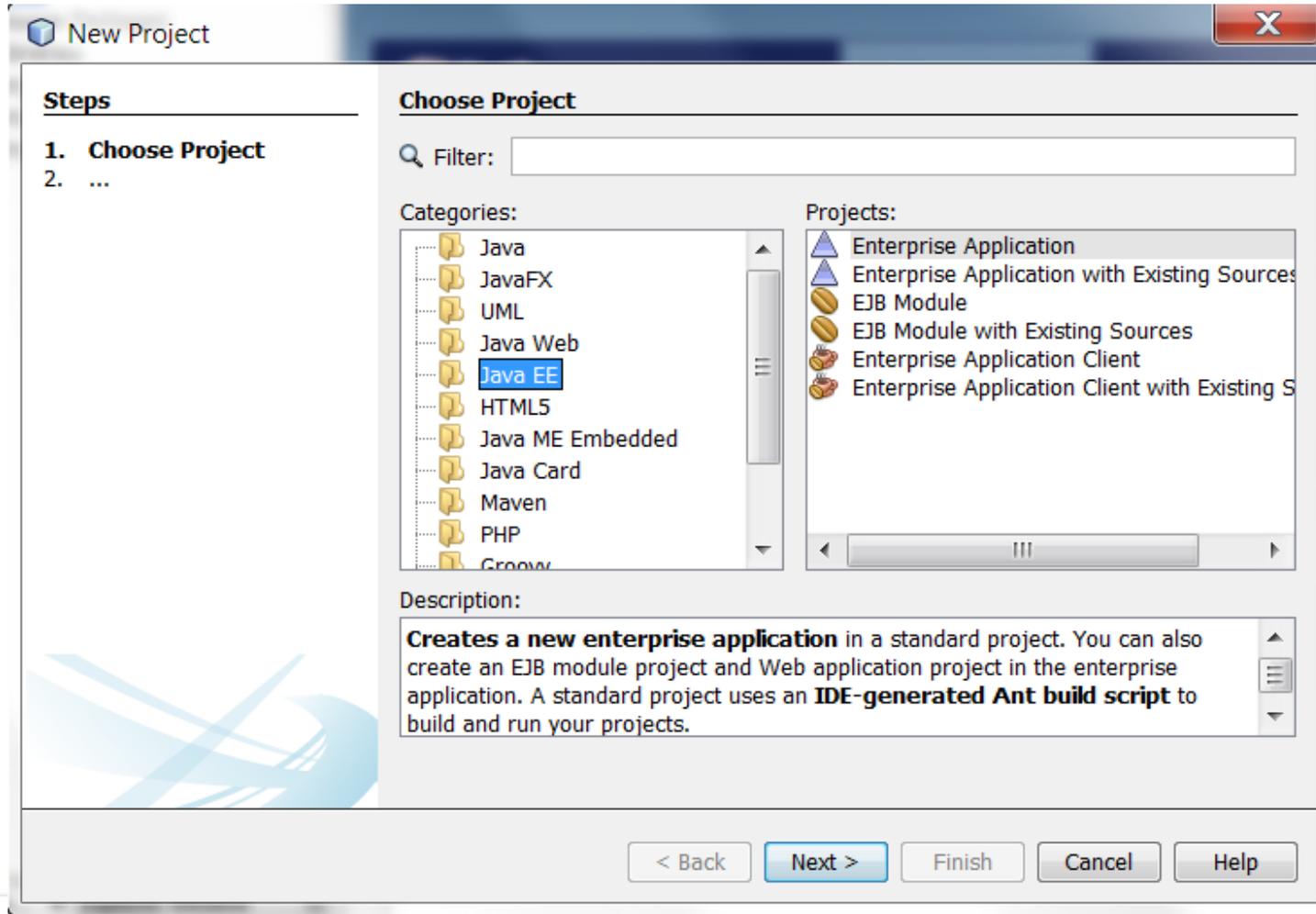


## 1. / 1.1. Create the Java EE project - right click File/New Project





## 1.2. Select Java EE\ Enterprise Application and click Next





### 1.3. Name and Location of EE project and next

The screenshot shows a 'New Enterprise Application' dialog box with a 'Steps' sidebar and a main 'Name and Location' section. The 'Steps' sidebar lists: 1. Choose Project, 2. **Name and Location**, and 3. Server and Settings. The 'Name and Location' section contains the following fields and options:

- Project Name: Library1\_EJB1
- Project Location: C:\EnglishLecture\Kruczkiewicz\Laboratory2015 (with a 'Browse...' button)
- Project Folder: nglshLecture\Kruczkiewicz\Laboratory2015\Library1\_EJB1
- Use Dedicated Folder for Storing Libraries
- Libraries Folder: (empty field with a 'Browse...' button)

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 >', 'Finish', 'Cancel', and 'Help'.



## 1.4. Server and Settings of EE project and next

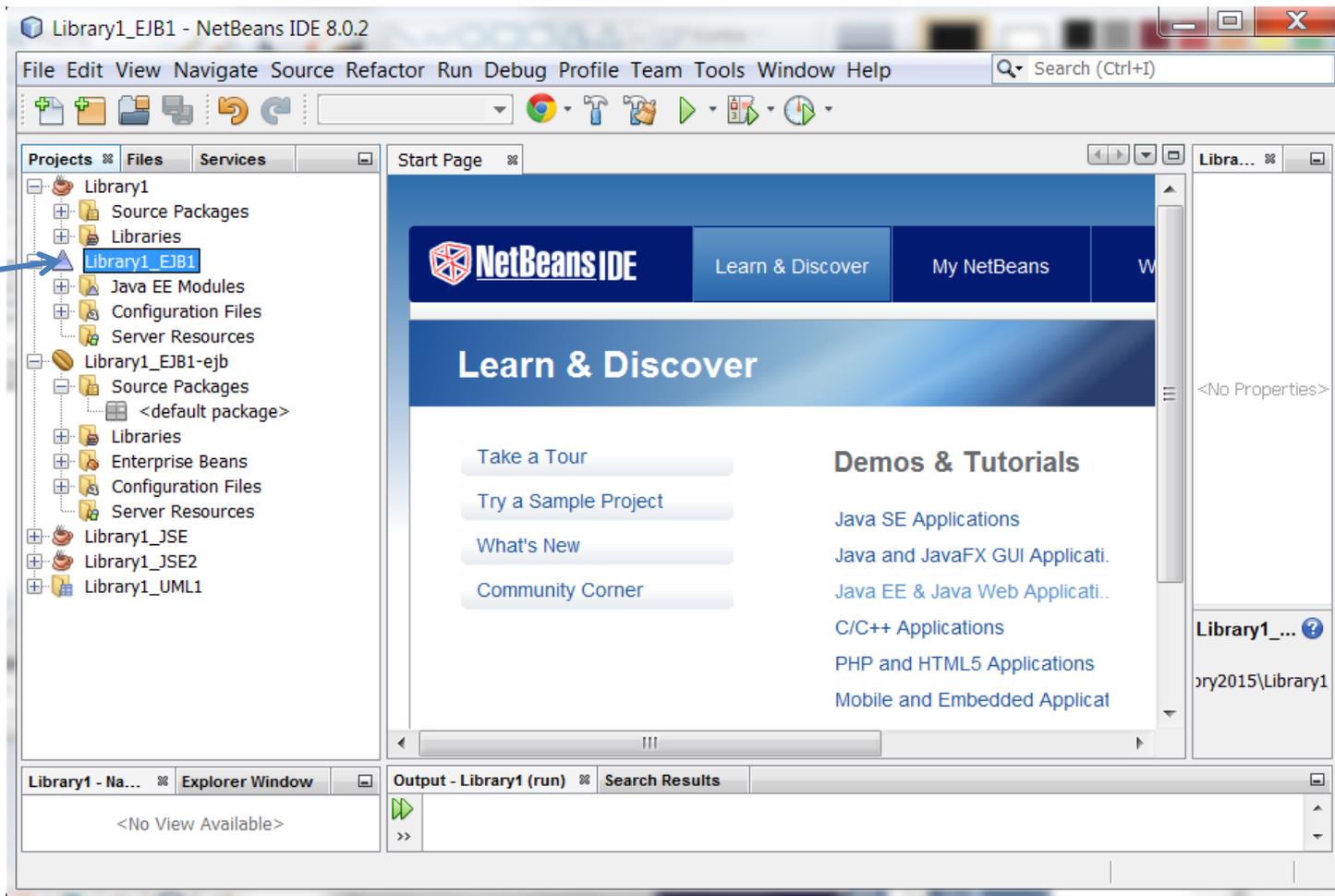
The screenshot shows the 'New Enterprise Application' wizard window. The title bar reads 'New Enterprise Application' with a close button. On the left, a 'Steps' pane lists: 1. Choose Project, 2. Name and Location, and 3. **Server and Settings**. The main area is titled 'Server and Settings' and contains the following fields and options:

- Server: GlassFish Server 4.1 (dropdown menu) with an 'Add...' button.
- Java EE Version: Java EE 7 (dropdown menu).
- Create EJB Module: Library1\_EJB1-ejb (text field).
- Create Web Application Module: Library1\_EJB1-war (text field).

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

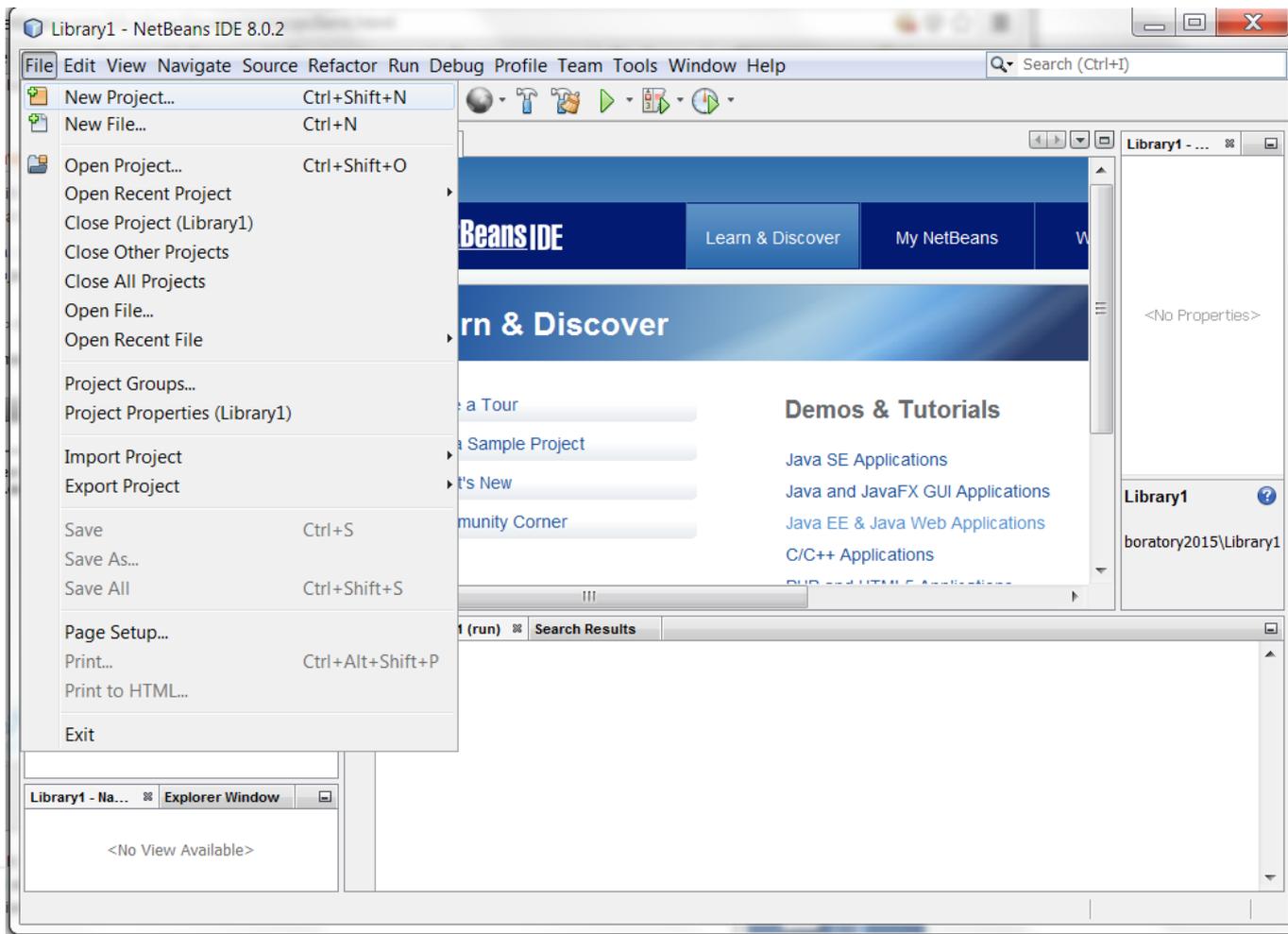


1.5. The new  
EE project



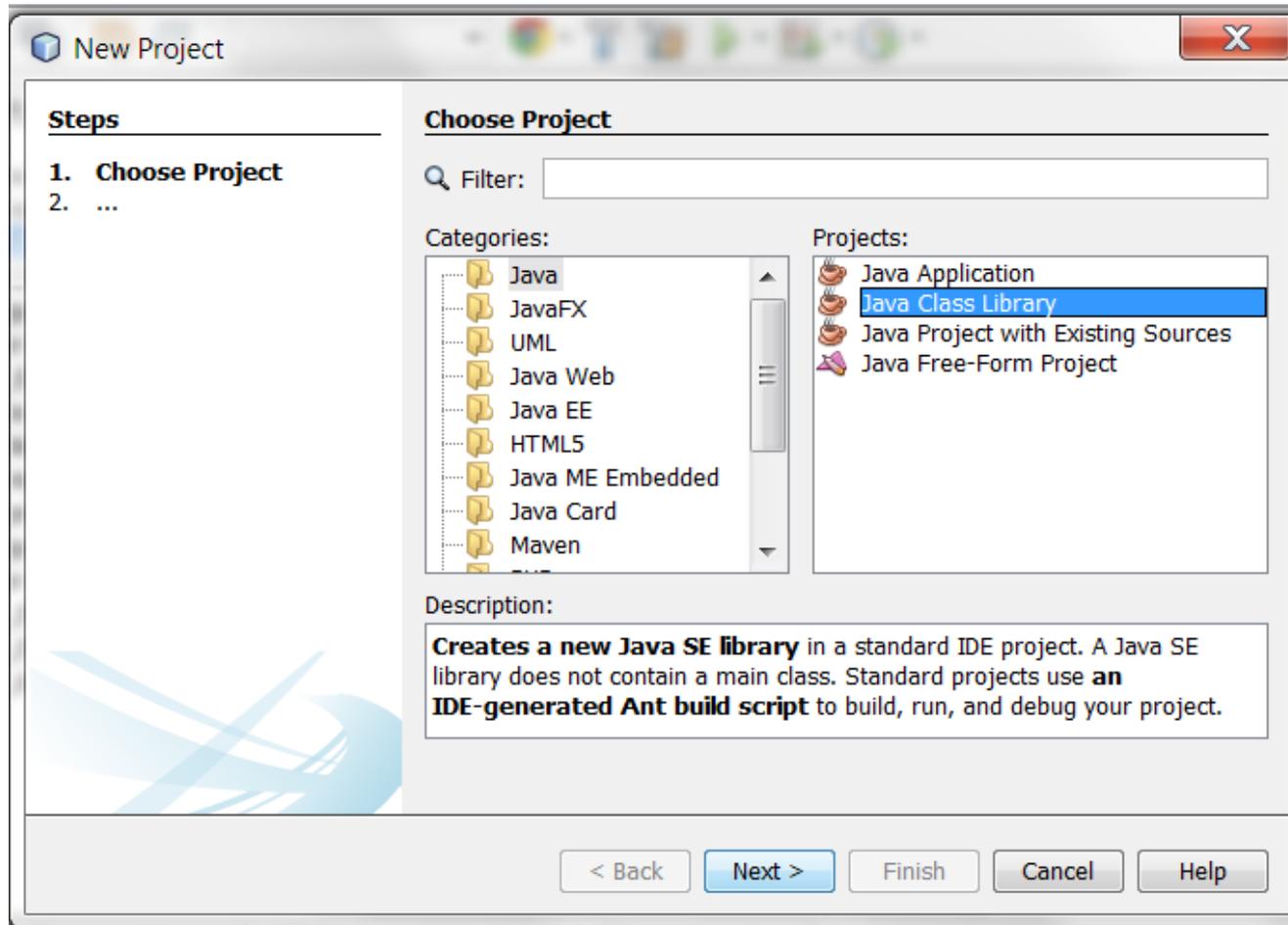


## 2. / 2.1. Creation the library project for interface of EJB





## 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

**Steps**

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

**Name and Location**

Project Name:

Project Location:  Browse...

Project Folder:

Use Dedicated Folder for Storing Libraries

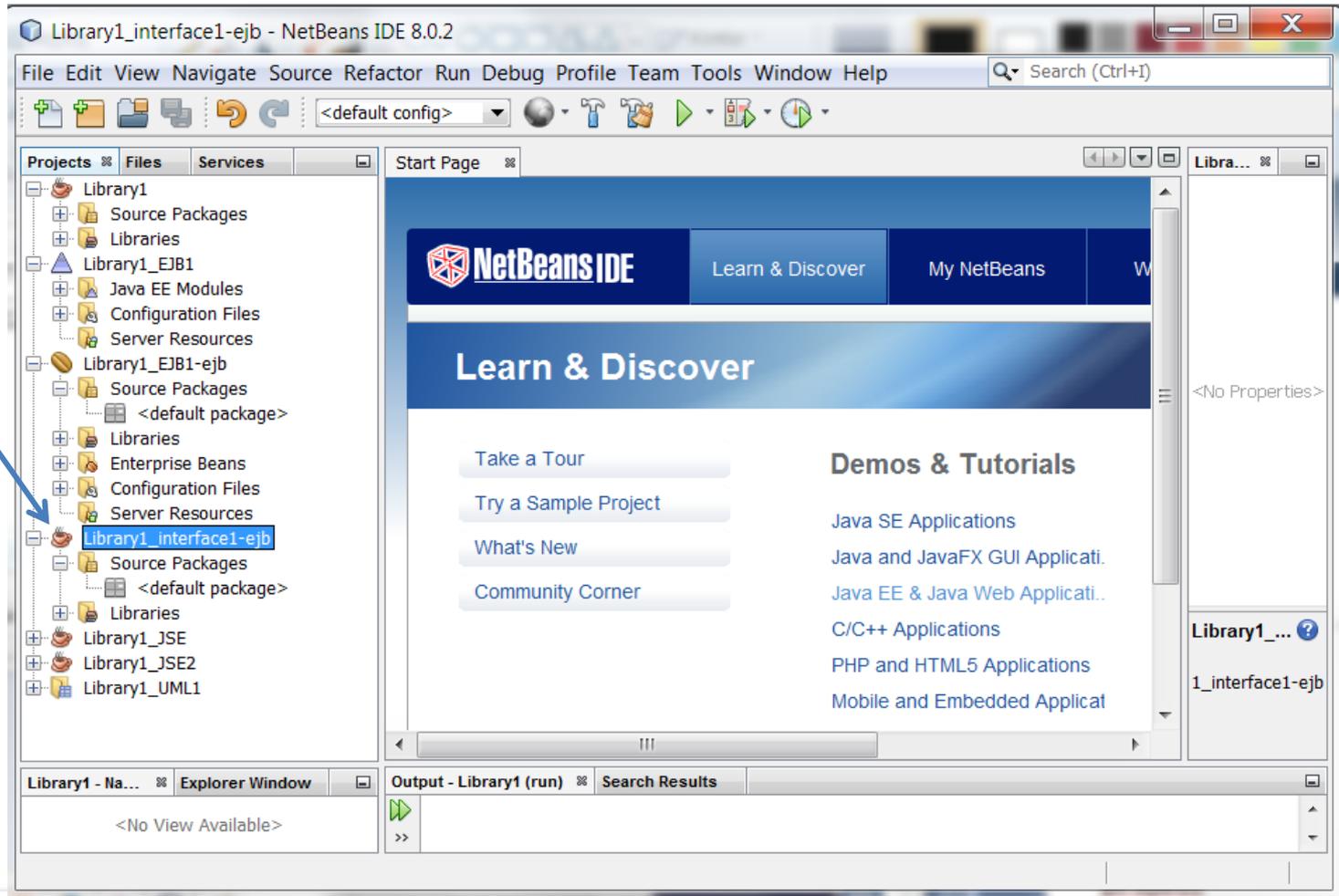
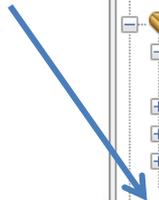
Libraries Folder:  Browse...

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

< Back   Next >   Finish   Cancel   Help

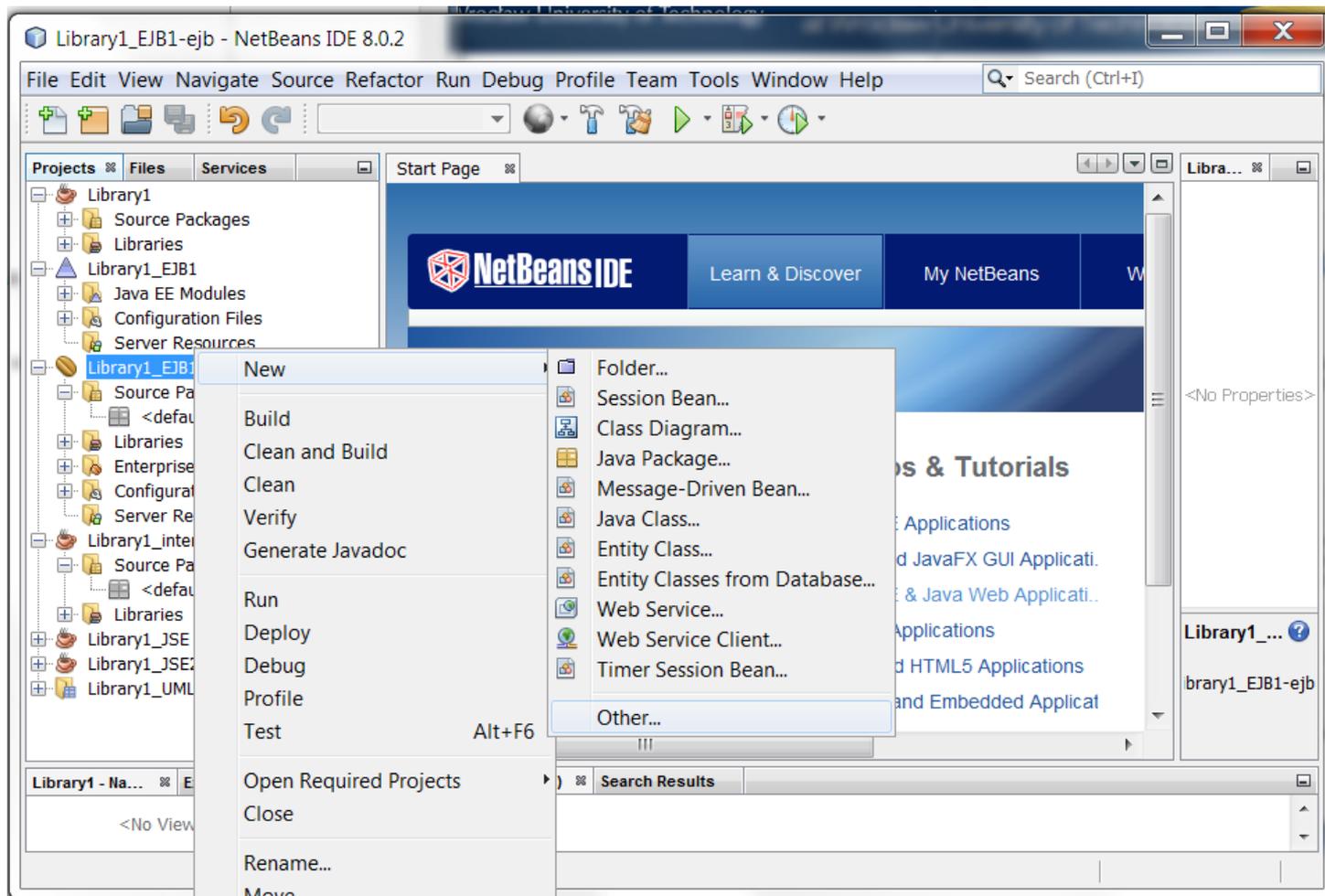


2.4. The new empty library project



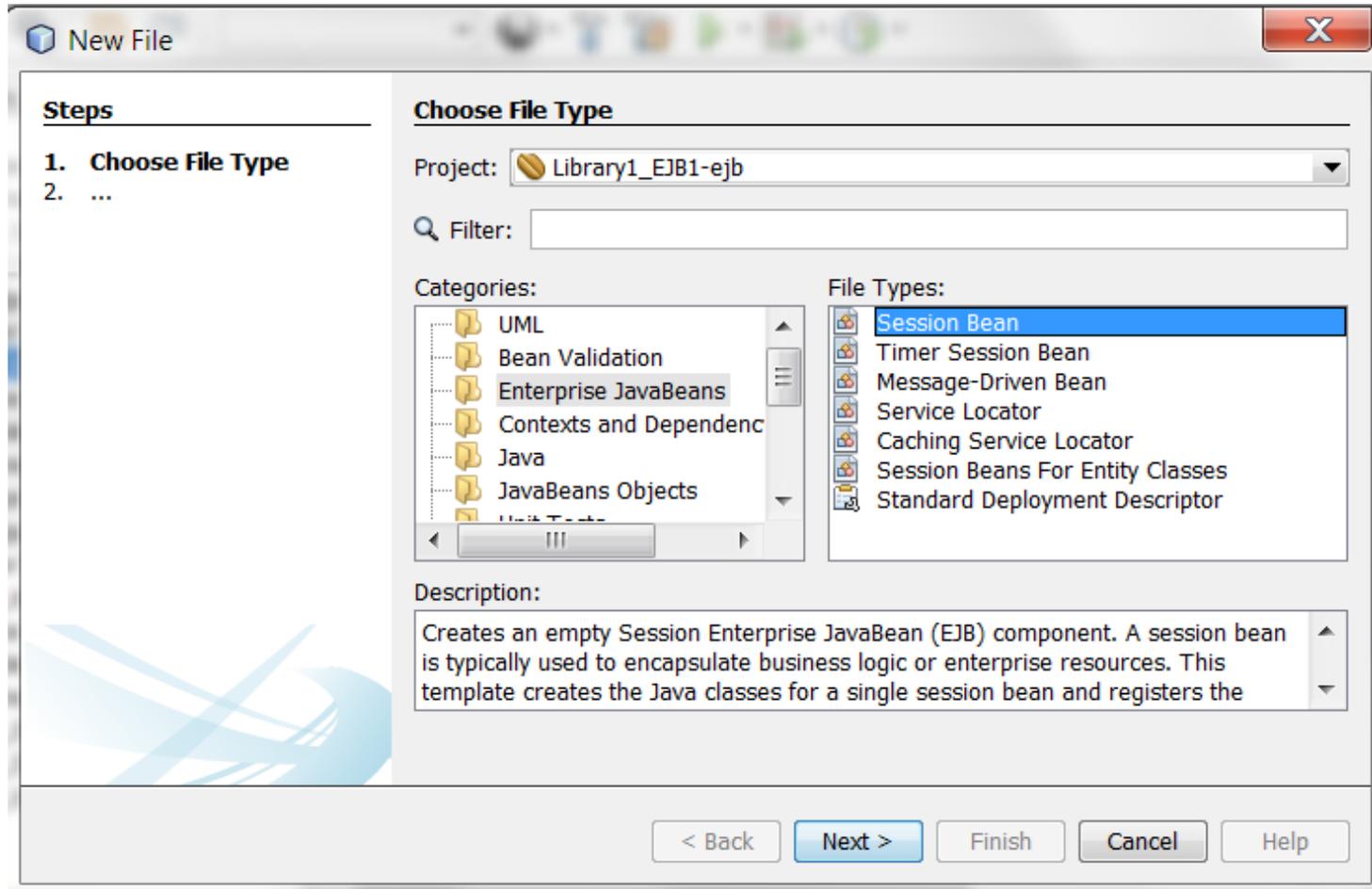


### 3. / 3.1. Adding the EJB as the Session Facade – Right click/New/Other





### 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

**Steps**

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

**Name and Location**

EJB Name:

Project:

Location:

Package:

Session Type:

Stateless

Stateful

Singleton

Create Interface:

Local

Remote in project:

< Back   Next >   Finish   Cancel   Help



3.4. The new empty EJB and its empty interface

The screenshot shows the NetBeans IDE interface for a project named 'Library1\_EJB1-ejb'. The left-hand 'Projects' pane displays a tree view of the project structure. A new package 'business\_tier' has been created under 'Library1\_EJB1-ejb'. Inside this package, two files are visible: 'Facade.java' and 'FacadeRemote.java'. Blue arrows point from the text '3.4. The new empty EJB and its empty interface' to these two files. The 'Facade.java' file is currently open in the editor, showing the following code:

```

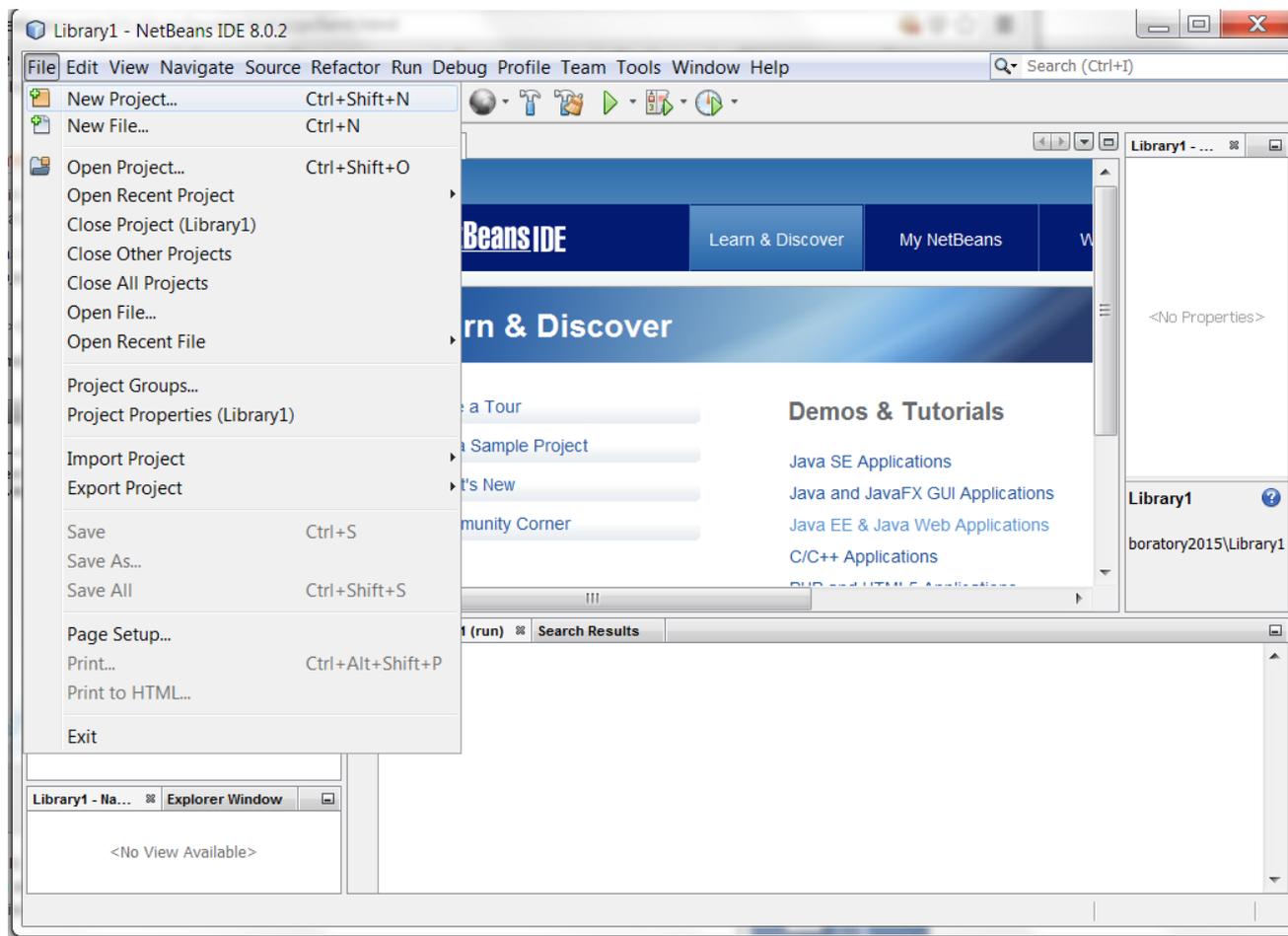
1  /**
2   * To change this license header, choose License Headers in
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package business_tier;
7
8  import javax.ejb.Stateless;
9
10 /**
11  *
12  * @author Zofia
13  */
14 @Stateless
15 public class Facade implements FacadeRemote {
16
17     // Add business logic below. (Right-click in editor and
18     // "Insert Code > Add Business Method")
19 }
20

```

The right-hand side of the IDE shows the 'Properties' window for 'Facade.java', displaying details such as Name, Extension, File Size, and Modification. Below the editor is the 'Output - Library1 (run)' window, which is currently empty.

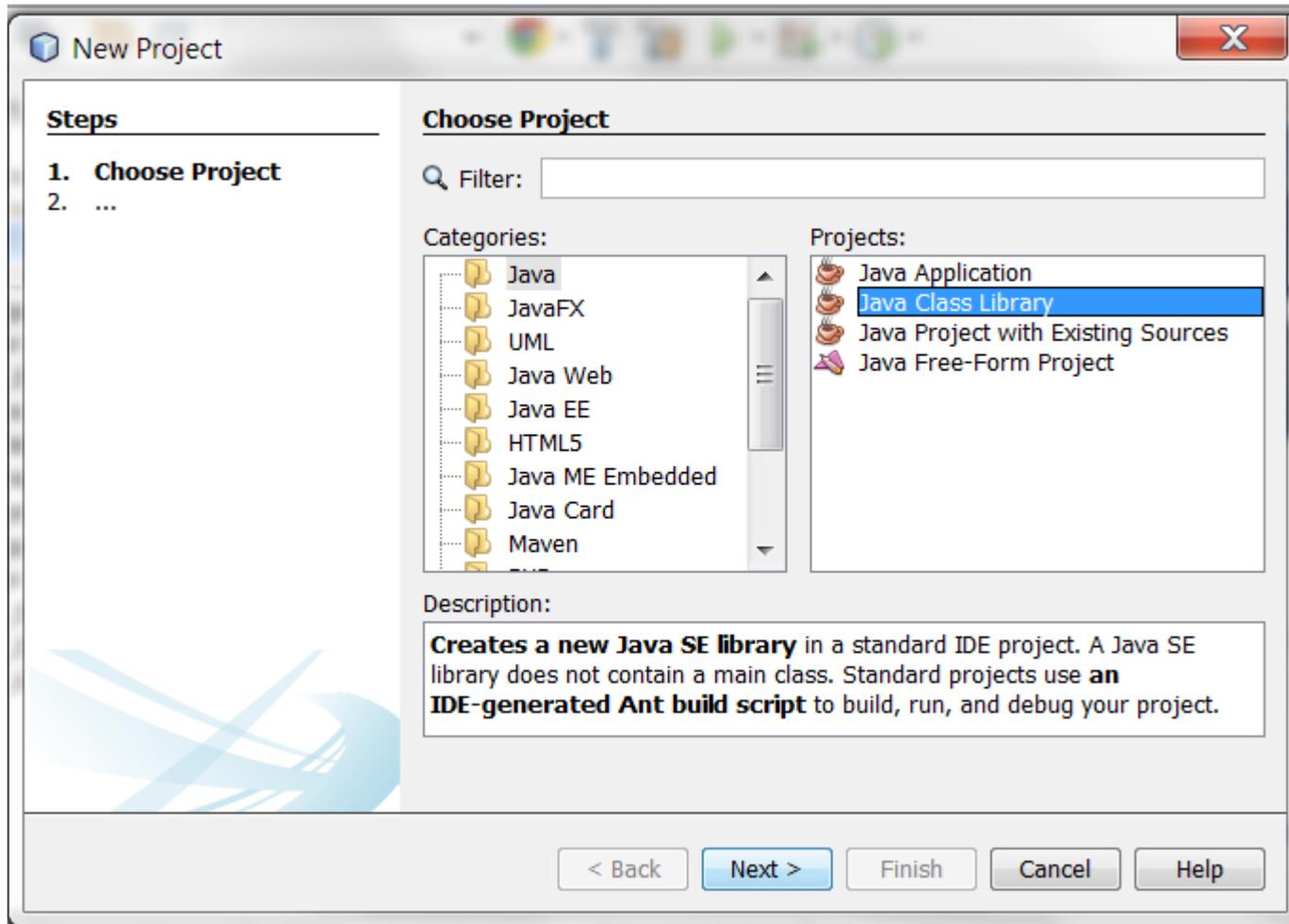


## 4. / 4.1./4.1.1. Creation the library project for code from lab1-2





#### 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

**Steps**

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

**Name and Location**

Project Name:

Project Location:

Project Folder:

Use Dedicated Folder for Storing Libraries

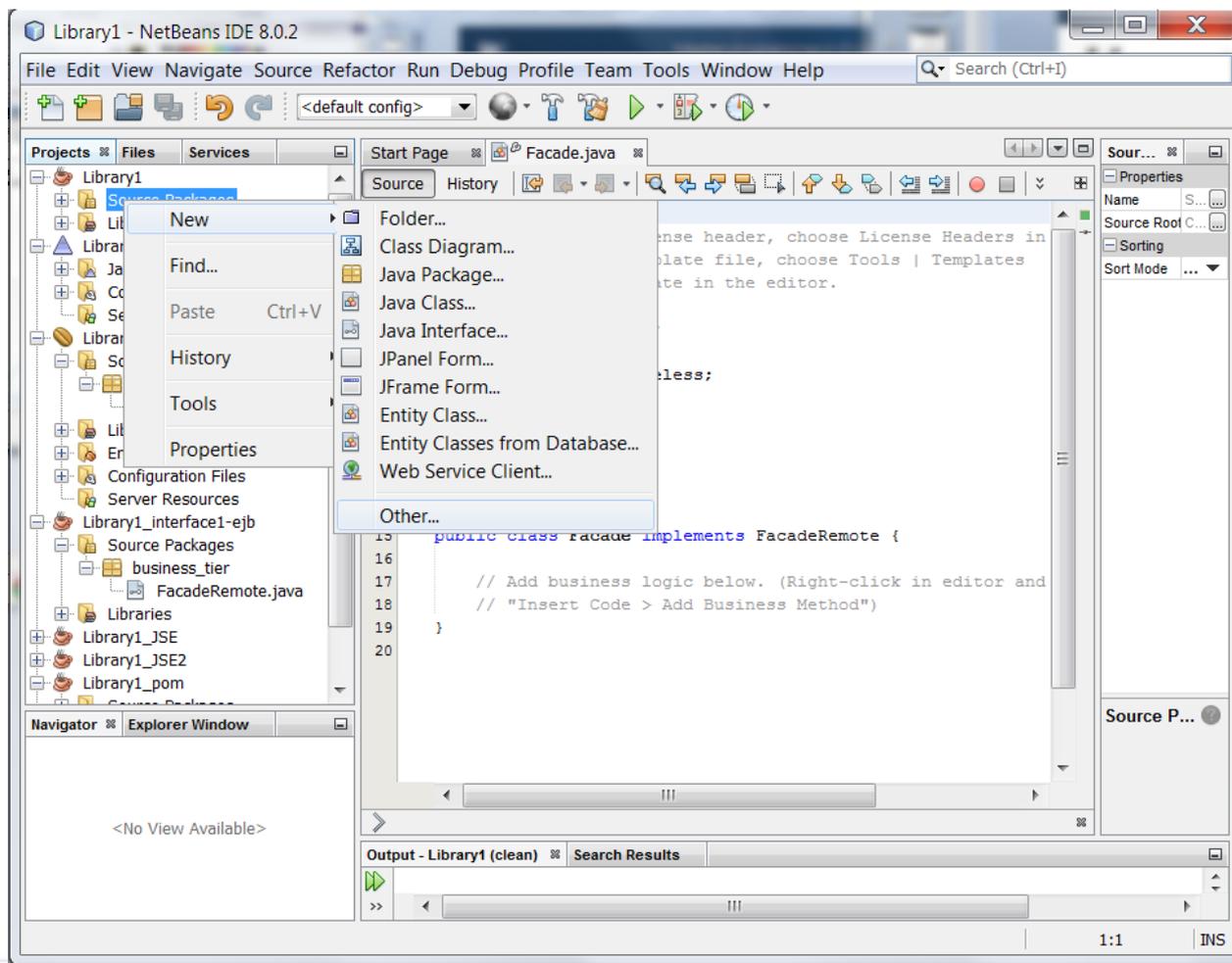
Libraries Folder:

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

< Back   Next >   **Finish**   Cancel   Help

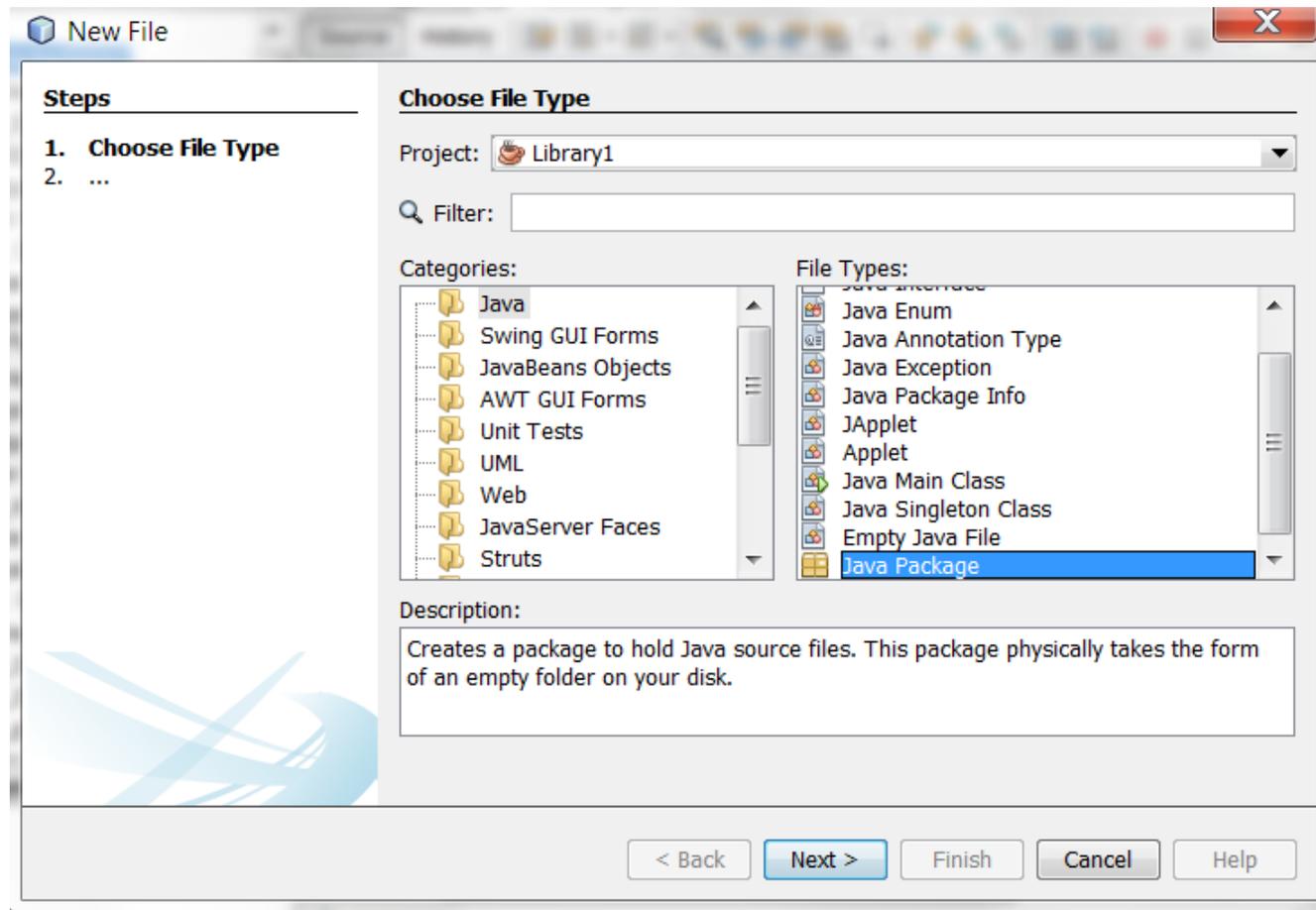


## 4.2. /4.2.1 Creation the package for TFacade and TFactory classes



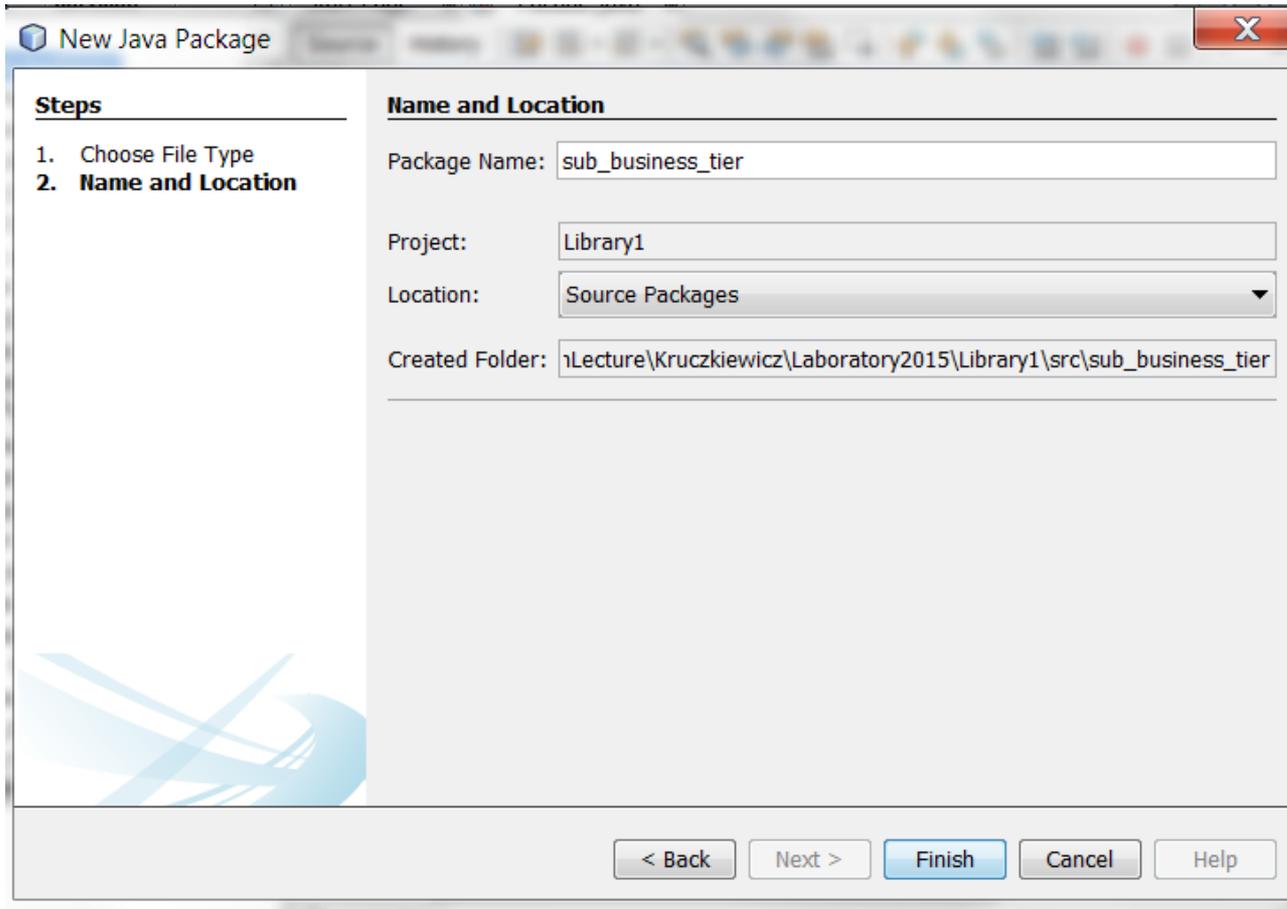


## 4.2.2. Creation the package for TFacade and TFactory classes – select Java/ Java Package



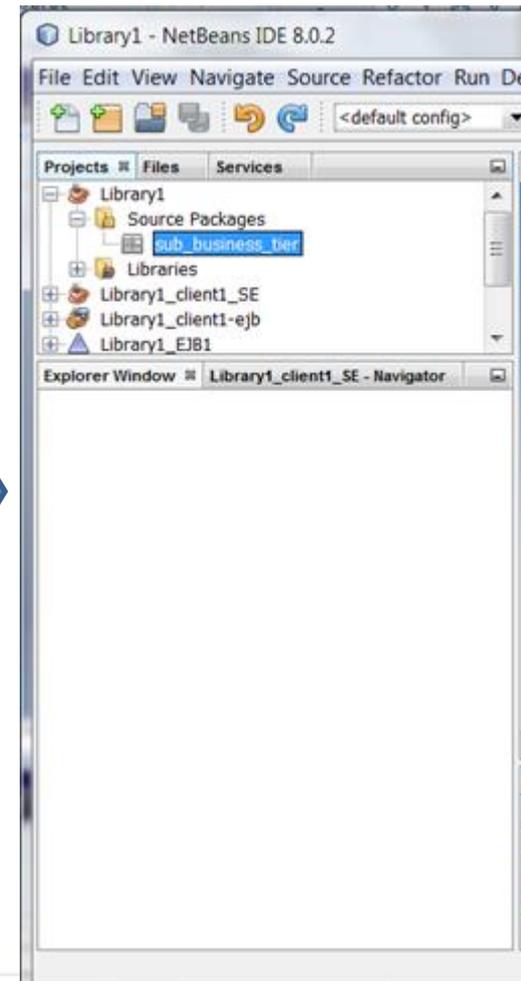


### 4.2.3. Creation the package for TFacade and TFactory classes – fill the name of package and select location . The result on the right.



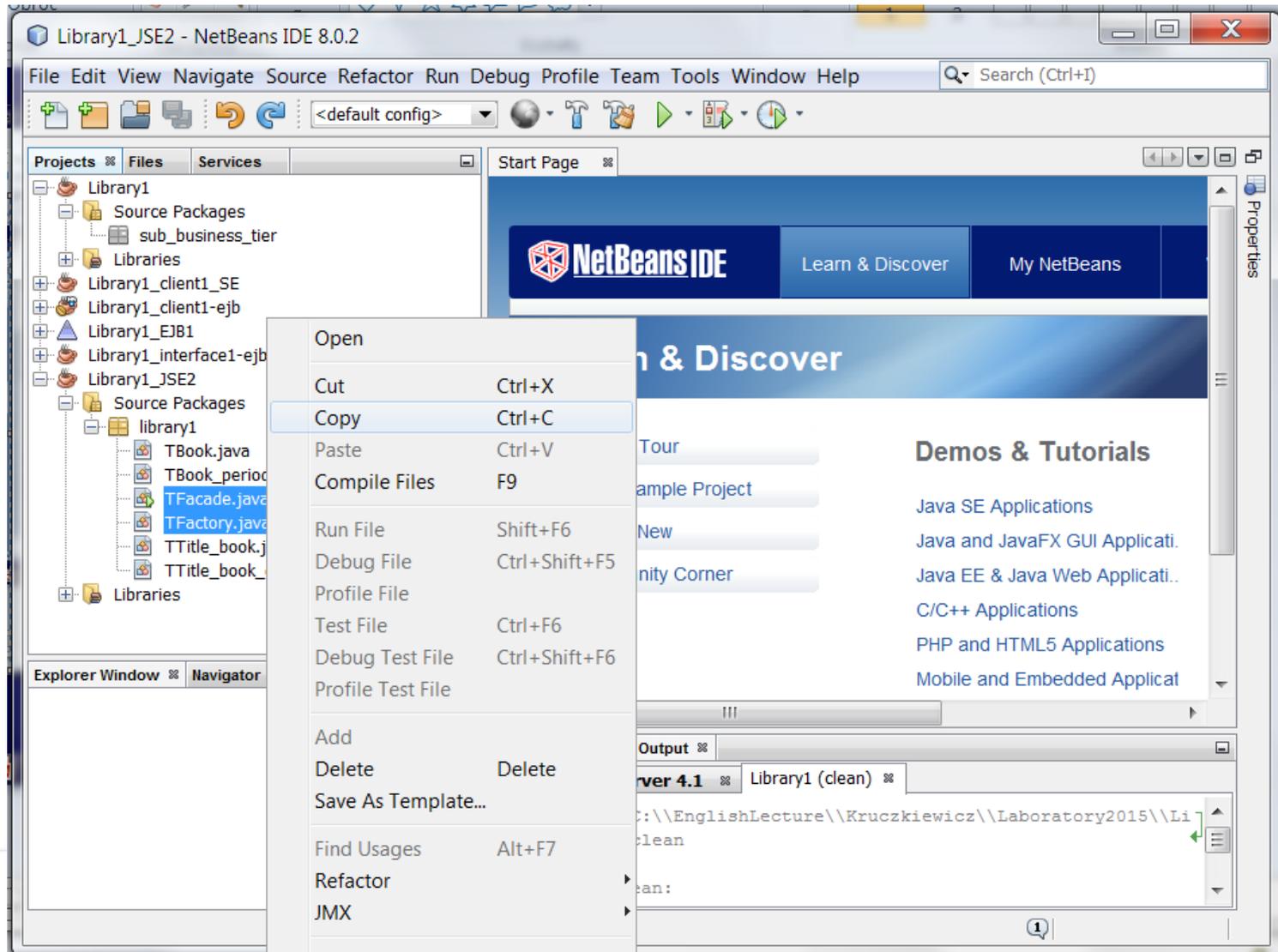
The dialog box 'New Java Package' is shown with the following fields and values:

- Steps:**
  1. Choose File Type
  2. **Name and Location**
- Name and Location:**
  - Package Name:
  - Project:
  - Location:
  - Created Folder:
- Buttons:** < Back, Next >, Finish, Cancel, Help



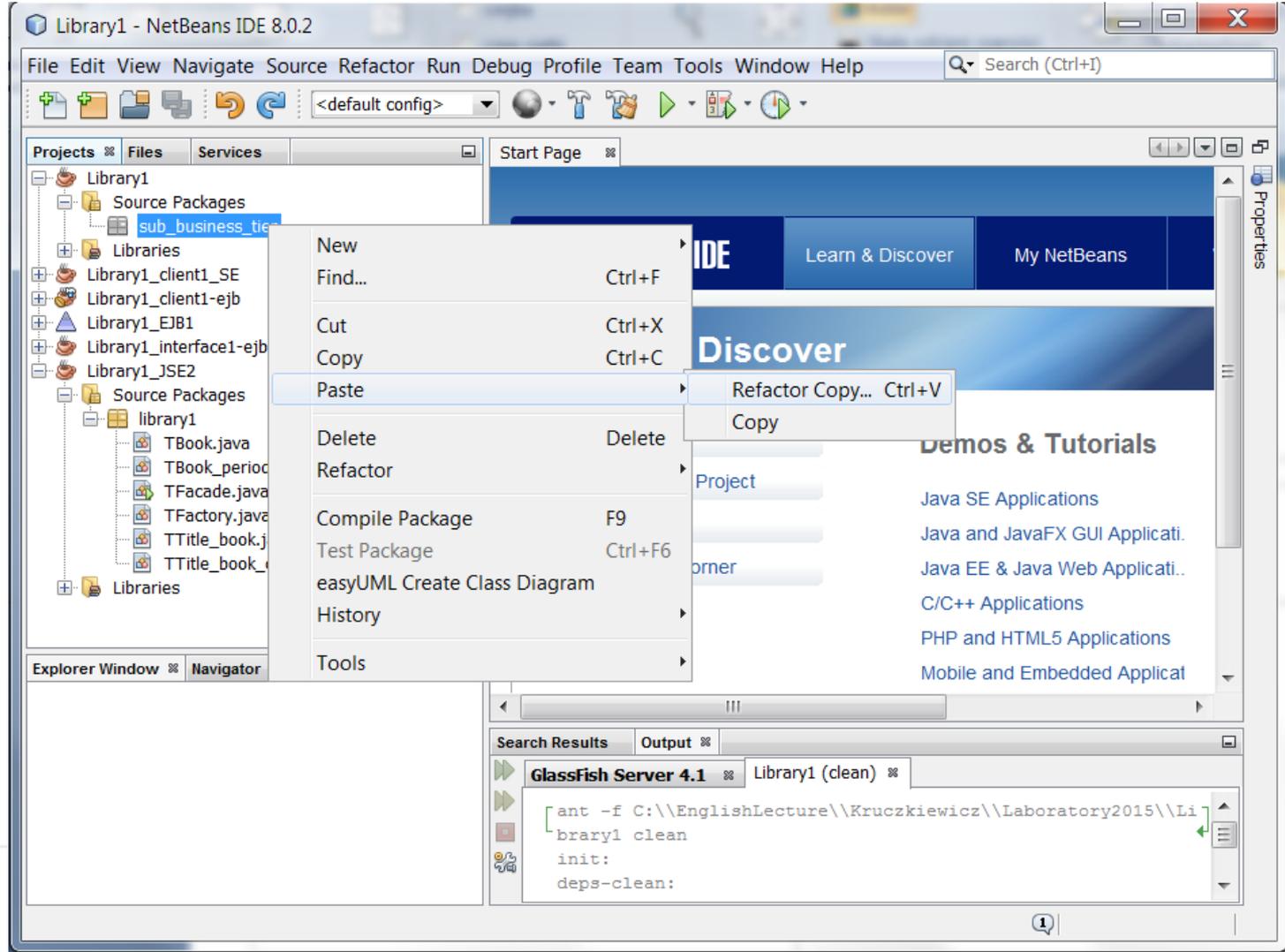


### 4.3. /4.3.1. Copying of TFacade and TFactory classes



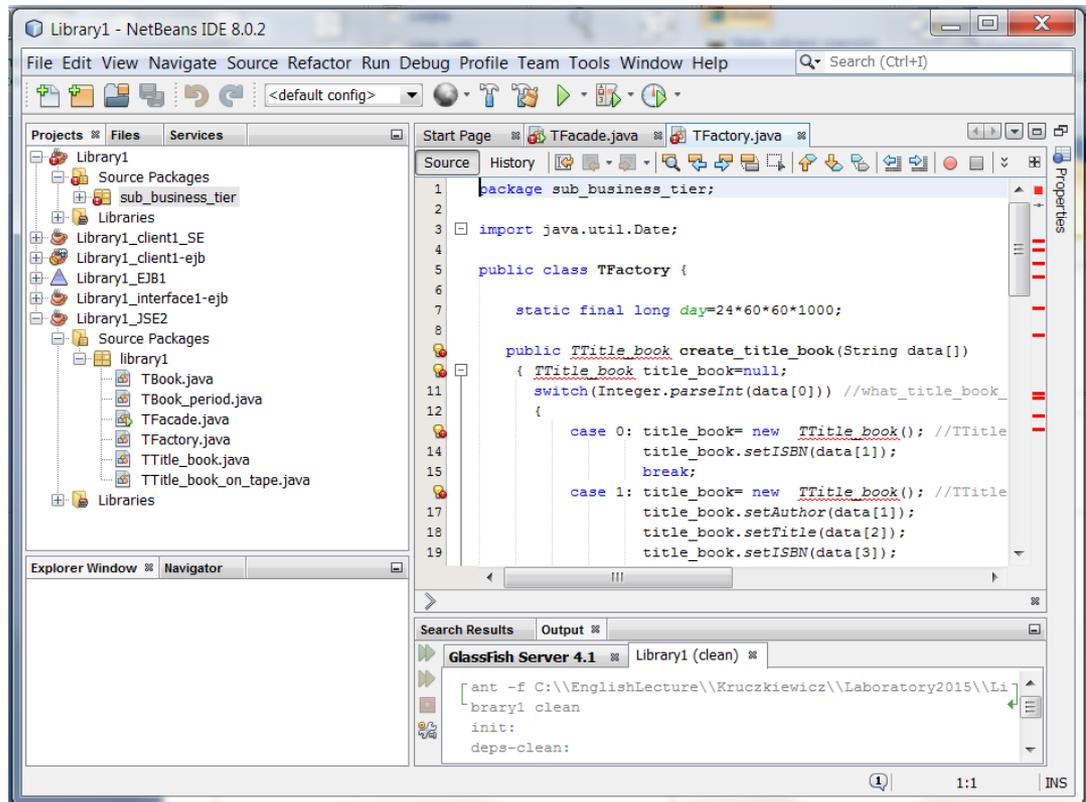
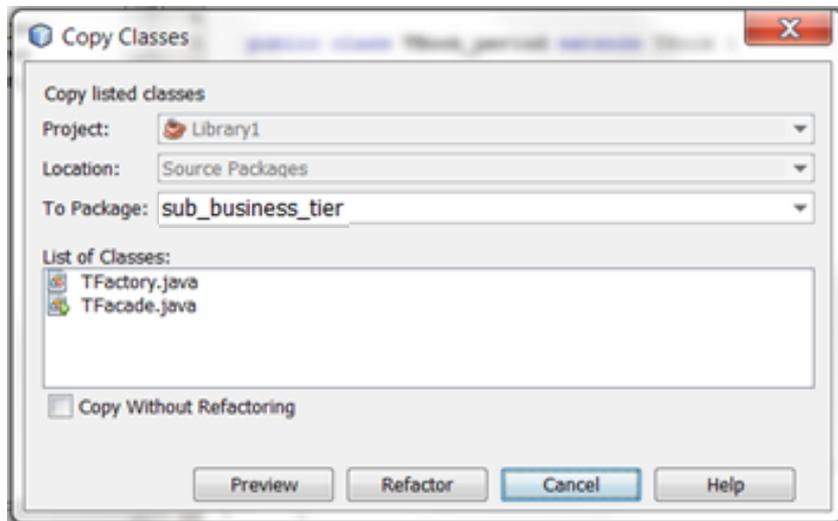


### 4.3.2. Paste the copies of the TFacade and TFactory classes from the lab1-2 project to the Library1 project (select the **sub\_business\_tier** package).





### 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

The screenshot shows the NetBeans IDE 8.0.2 interface. The main window displays the source code for `TFactory.java` in the `sub_business_tier` package. The code is as follows:

```
1 package sub_business_tier;
2
3 import java.util.Date;
4
5 public class TFactory {
6
7     static final long day=24*60*60*1000;
8
9     public TTitle_book create_title_book(String data[])
10    { TTitle_book title_book=null;
11      switch(Integer.parseInt(data[0])) //what_title_book_
12    {
13
14        case 0: title_book= new TTitle_book(); //TTitle
15              title_book.setISBN(data[1]);
16              break;
17
18        case 1: title_book= new TTitle_book(); //TTitle
19              title_book.setAuthor(data[1]);
20              title_book.setTitle(data[2]);
21              title_book.setISBN(data[3]);
22    }
23    }
```

The IDE interface includes a project explorer on the left showing the `Library1` project structure, a code editor in the center, and a console window at the bottom right showing the output of a `clean` command:

```
ant -f C:\\EnglishLecture\\Kruczkiewicz\\Laboratory2015\\Li
brary1 clean
init:
deps-clean:
```



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

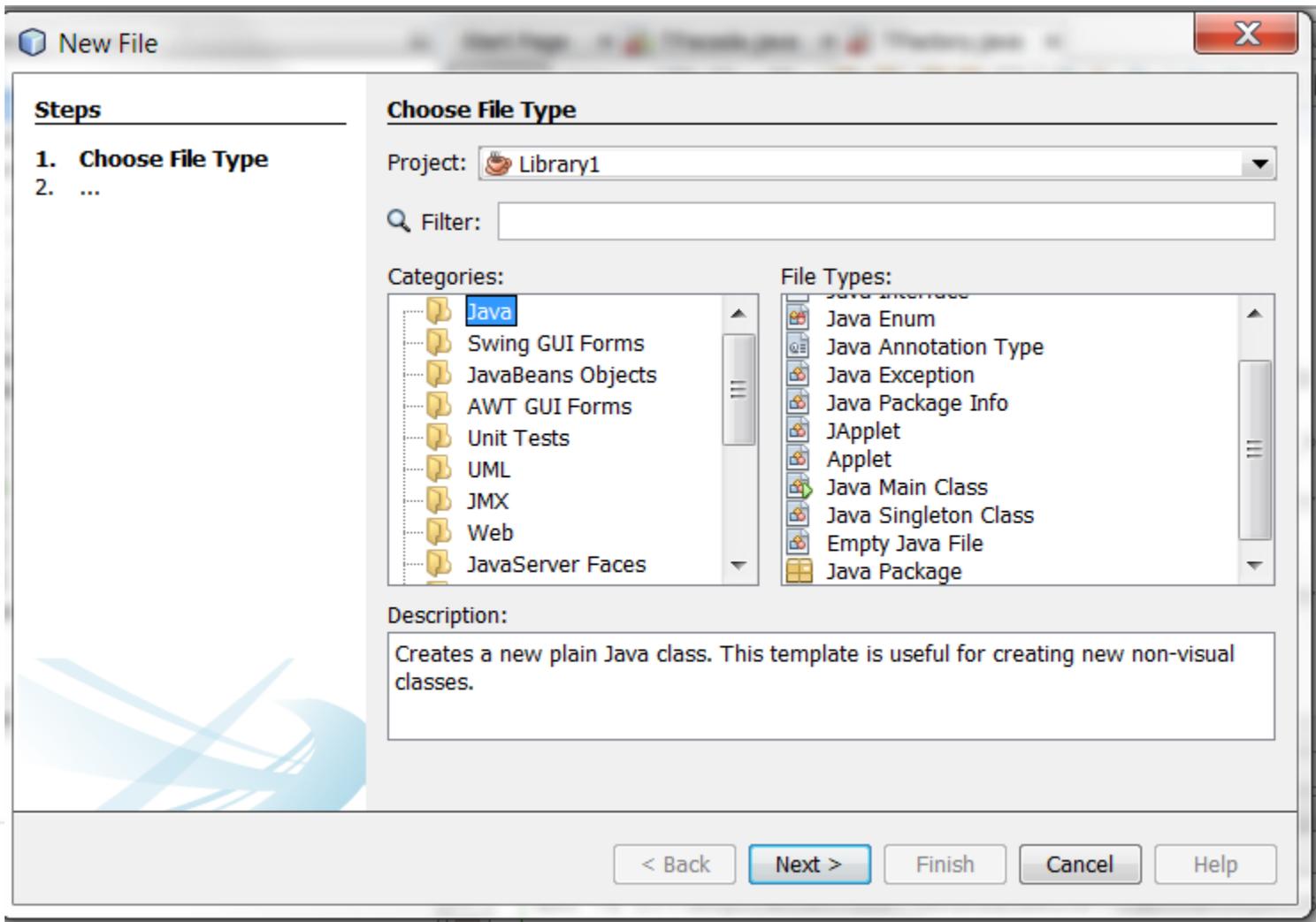
The screenshot shows the NetBeans IDE 8.0.2 interface. The 'New' menu is open, and 'Java Package...' is selected. The Explorer window shows a project structure with 'library1' selected. The editor shows code for TFactory.java. The Output window shows the command 'ant -f C:\EnglishLecture\Kruczkiewicz\Laboratory2015\Library1 clean'.

```
tier;  
;  
{  
    day=24*60*60*1000;  
    create_title_book(String data[])  
    le_book=null;  
    parseInt(data[0])) //what_title_book_  
  
    le_book= new TTitle_book(); //TTitle  
    le_book.setISBN(data[1]);  
    ak;  
    le_book= new TTitle_book(); //TTitle  
    le_book.setAuthor(data[1]);  
    le_book.setTitle(data[2]);  
    le_book.setISBN(data[3]);
```

```
ant -f C:\EnglishLecture\Kruczkiewicz\Laboratory2015\Library1 clean  
init:  
deps-clean:
```



### 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

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
  - Libraries
  - Library1\_client1\_SE
  - Library1\_client1-ejb
  - Library1\_EJB1
  - Library1\_interface1-ejb
  - Library1\_JSE2
  - Source Packages
    - library1
      - TBook.java
      - TBook\_period.java
      - TFacade.java
      - TFactory.java
      - TTitle\_book.java
      - TTitle\_book\_on\_tape.java

Start Page TFacade.java TFactory.java

Source History

```
2
3 import java.util.ArrayList;
4 import java.util.Arrays;
5 import java.util.List;
6
7
8 public class TFacade {
9
10     // List<TTitle_book> mTitle_books;
11     List<TTitle_book> mTitle_books;
12
13     public TFacade() {mTitle_books = new ArrayList<>();
14     }
15
16     public List<TTitle_book> getmTitle_books() {
17         return mTitle_books;
18     }
19
20     void setmTitle_books(List<TTitle_book> title_books) {
21         mTitle_books = title_books;
22     }
23 }
```

Search Results Output

GlassFish Server 4.1 Library1 (run)

6:1 INS



#### 4.5. /4.5.1. Copying of the TTitle\_book and other classes as data classes from lab1-2 project

The screenshot shows the NetBeans IDE 8.0.2 interface. The Explorer window on the left displays a project structure with a 'library1' package containing several Java files. A context menu is open over the 'TTitle\_book.java' file. The menu items include: Open, Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V), Compile Files (F9), Run File (Shift+F6), Debug File (Ctrl+Shift+F5), Profile File, Test File (Ctrl+F6), Debug Test File (Ctrl+Shift+F6), Profile Test File, Add, Delete, Save As Template..., Find Usages (Alt+F7), Refactor, and JMX. The main editor window shows the source code of 'TFacade.java' with the following imports:

```
2  
3 import java.util.ArrayList;  
4 import java.util.Arrays;  
5 import java.util.List;
```

The Properties window on the right is empty. The status bar at the bottom indicates '6:1' and 'INS'.

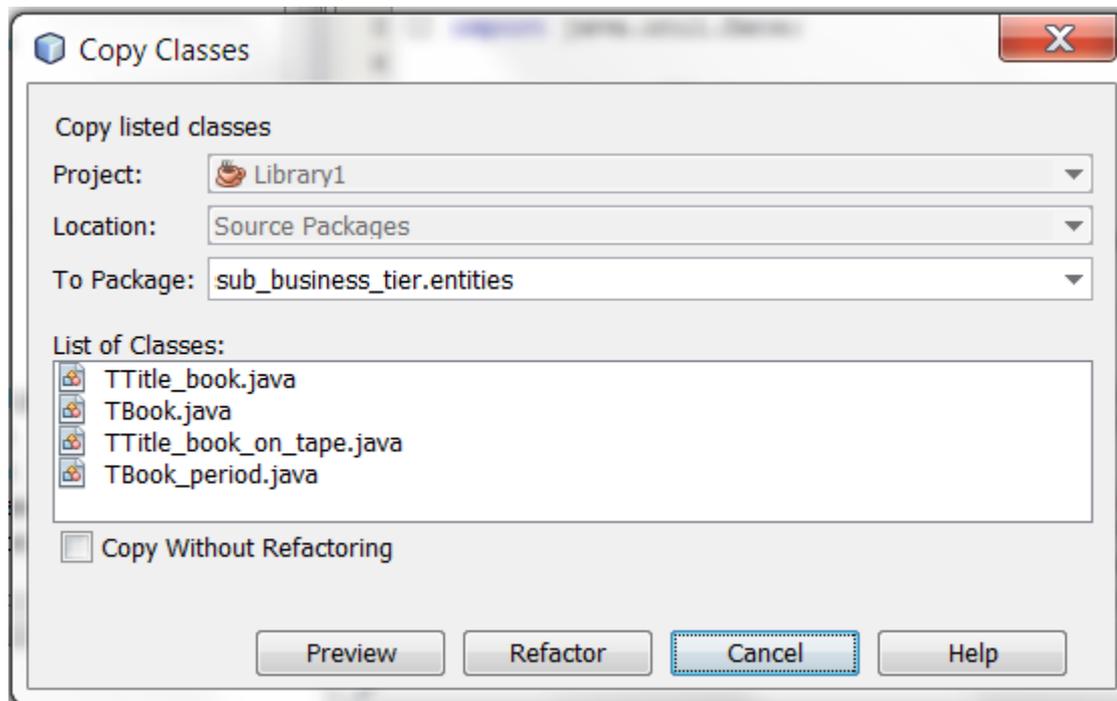


### 4.5.2 Paste the copies of the TTitle\_book and other classes as data classes from the lab1-2 project to the Library1 project (select the **sub\_business\_tier.entities** package).

The screenshot displays the NetBeans IDE 8.0.2 interface. The Explorer Window on the left shows the project structure for 'Library1', with the 'sub\_business\_tier.entities' package selected. A context menu is open over this package, showing options like 'New', 'Find...', 'Cut', 'Copy', 'Paste', 'Delete', 'Refactor', 'Compile Package', 'Test Package', 'easyUML Create Class Diagram', 'History', and 'Tools'. The 'Paste' option is highlighted, and a sub-menu is visible with 'Refactor Copy...' and 'Copy' options. The main editor shows the code for 'TFacade.java' and 'TFactory.java'. The Search Results window at the bottom shows 'GlassFish Server 4.1' and 'Library1 (run)'. The status bar at the bottom right indicates '6:1' and 'INS'.



**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).**





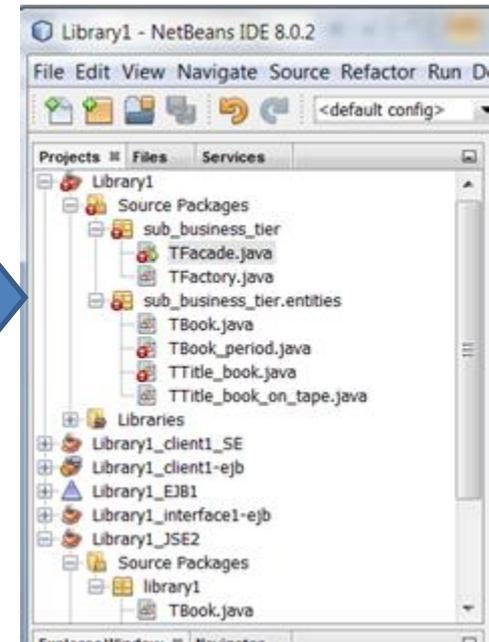
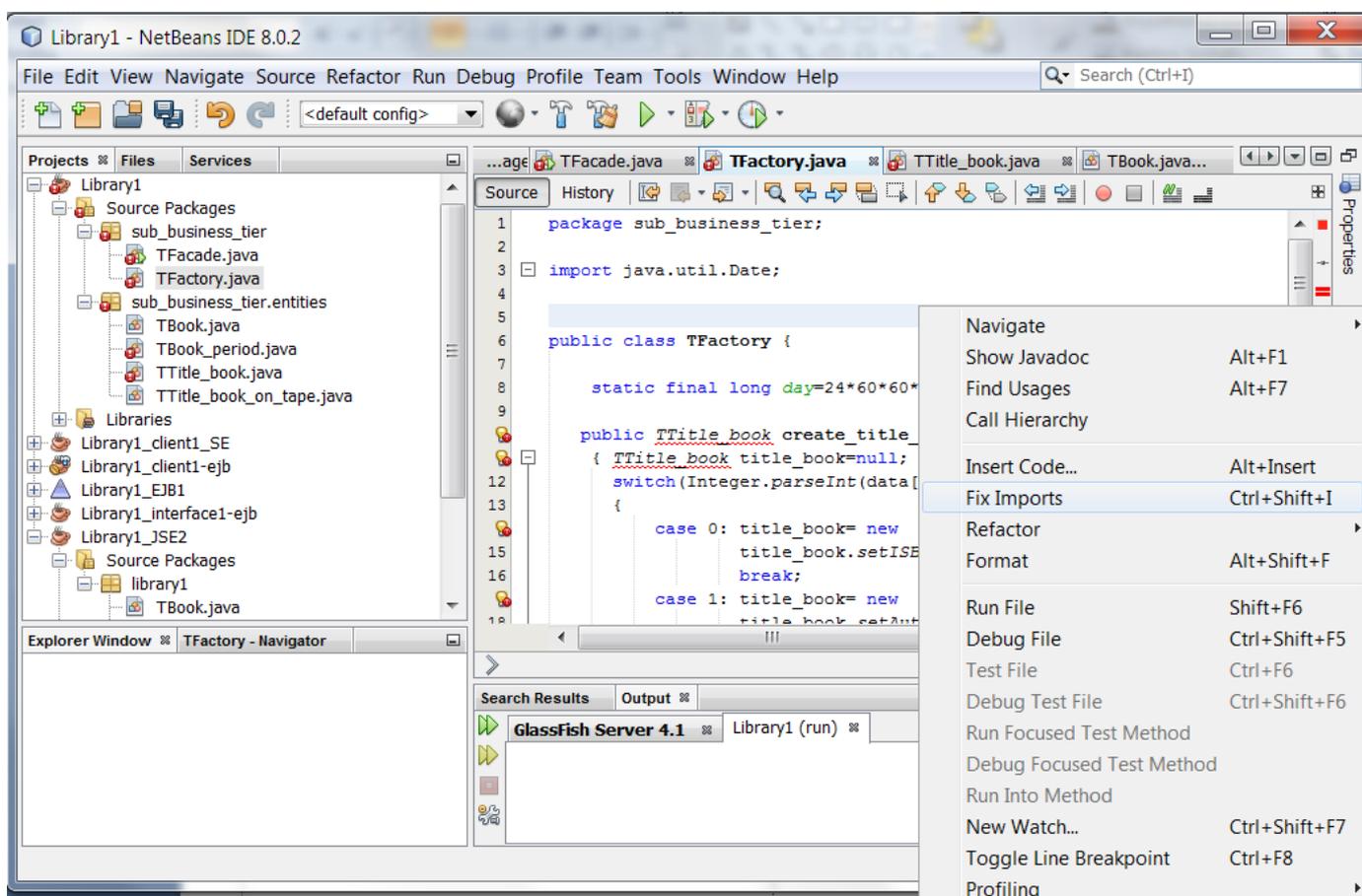
### 4.5.4. Paste the copies of the TTitle\_book and other classes as data classes from the lab1-2 project to the Library1 project - result

The screenshot shows the NetBeans IDE interface for the 'Library1' project. The 'Projects' window on the left displays the project structure, with the 'sub\_business\_tier.entities' package highlighted. The 'Source' window in the center shows the code for 'TBook\_period.java', which is a class extending 'TBook' with a 'period' attribute and methods 'getPeriod()' and 'setPeriod()'. The 'Search Results' window at the bottom shows the 'GlassFish Server 4.1' and 'Library1 (run)' tabs.

```
1 package sub_business_tier.entities;
2
3 import java.util.Date;
4
5 public class TBook_period extends TBook {
6
7     private Date period;
8
9     @Override
10    public Date getPeriod() {
11        return period;
12    }
13
14    @Override
15    public void setPeriod(Date date) {
16        period = date;
17    }
18
19    @Override
```



### 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

The screenshot shows the NetBeans IDE interface. The main editor window displays the source code for the `TFacade` class in the `sub_business_tier` package. The code includes imports for `java.util.ArrayList`, `java.util.Arrays`, and `java.util.List`. A context menu is open over the code, with the **Fix Imports** option selected. The menu items and their shortcuts are as follows:

Navigate	
Show Javadoc	Alt+F1
Find Usages	Alt+F7
Call Hierarchy	
Insert Code...	Alt+Insert
<b>Fix Imports</b>	<b>Ctrl+Shift+I</b>
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



The screenshot shows the project tree in NetBeans IDE 8.0.2. The `TFacade.java` file is now listed under the `sub_business_tier` source package, indicating that the 'Fix Imports' action has been successfully executed.



### 4.6.3. „Fix Imports” process for adding the lacking imports of classes – the TTitle\_book class

The screenshot shows the NetBeans IDE interface. The Explorer Window on the left displays the project structure with 'TTitle\_book.java' selected under 'sub\_business\_tier.entities'. The Source Editor in the center shows the code for 'TFacade.java' with a yellow lightning bolt icon indicating a missing import for 'TTitle\_book'. A context menu is open over the code, with 'Fix Imports' (Ctrl+Shift+I) highlighted. The menu also includes options like 'Navigate', 'Show Javadoc', 'Find Usages', 'Call Hierarchy', 'Refactor', 'Format', 'Run File', 'Debug File', 'Test File', 'Debug Test File', 'Run Focused Test Method', 'Debug Focused Test Method', 'Run Into Method', 'New Watch...', and 'Toggle Line Breakpoint'.



The screenshot shows the NetBeans IDE interface after the 'Fix Imports' process. The Explorer Window on the left now shows 'TTitle\_book.java' as a separate file under 'sub\_business\_tier.entities', indicating that the import has been successfully added to the code.



### 4.6.4. „Fix Imports” process for adding the lacking imports of classes – the TBook\_period class

The screenshot shows the NetBeans IDE interface. The Explorer window on the left displays the project structure with 'TBook\_period.java' selected. The main editor shows the source code of 'TBook\_period.java' with several red squiggly lines under the 'TFactory' and 'Date' identifiers, indicating missing imports. A context menu is open over the code, with 'Fix Imports' highlighted. The menu items include: 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), and Toggle Line Breakpoint (Ctrl+F8).



The screenshot shows the same NetBeans IDE interface after the 'Fix Imports' action. The Explorer window now shows 'TBook\_period.java' with a green checkmark icon, indicating that the missing imports have been successfully added to the file.



### 4.6.5. The result

The screenshot displays the NetBeans IDE 8.0.2 interface. The main window shows the source code for `TBook_period.java` in the `sub_business_tier.entities` package. The code includes the following methods:

```
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 IDE interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help), a toolbar, and a project explorer on the left. The bottom status bar shows "TBook\_period.java saved." and the system tray displays the time "20:14" and "INS".



## 4.7. Build and Run the Library1 Project

The screenshot displays the NetBeans IDE interface for the 'Library1' project. The left sidebar shows the project structure with source packages and libraries. The main editor window shows the source code for 'TBook\_period.java', which includes methods for getting and setting a period, and a period\_pass method. The bottom output window shows the results of running the application on GlassFish Server 4.1, displaying a list of book titles and their associated metadata.

```
public Date getPeriod() {
    return period;
}

@Override
public void setPeriod(Date date) {
    period = date;
}

@Override
public boolean period_pass(Object data) {
    sub_business_tier.entities.TBook_period > period_pass >
}
```

```
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]
```



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

The screenshot shows the NetBeans IDE interface for a project named 'Library1\_EJB1-ejb'. The 'Projects' window on the left displays a tree view of the project structure, including source packages like 'business\_tier'. A context menu is open over the 'Libraries' folder, with options: 'Add Project...', 'Add Library...', 'Add JAR/Folder...', and 'Properties'. The 'Add Library...' option is highlighted. The main editor window shows the source code for 'TFacade.java' in the 'sub\_business\_tier' package. The code includes imports for 'java.util.ArrayList', 'java.util.Arrays', 'java.util.List', and 'sub\_business\_tier.entities.TTitle\_book'. A 'public class TFacade' is defined with a 'List<TTitle\_book> mTitle\_books' attribute and a constructor that initializes 'mTitle\_books' as a new 'ArrayList<>()'. The 'Output - Library1 (run)' window at the bottom shows the execution results, displaying a list of book titles and authors, and a list of books with their titles, authors, ISBNs, publishers, and numbers.

```
package sub_business_tier;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import sub_business_tier.entities.TTitle_book;

public class TFacade {

    List<TTitle_book> mTitle_books;

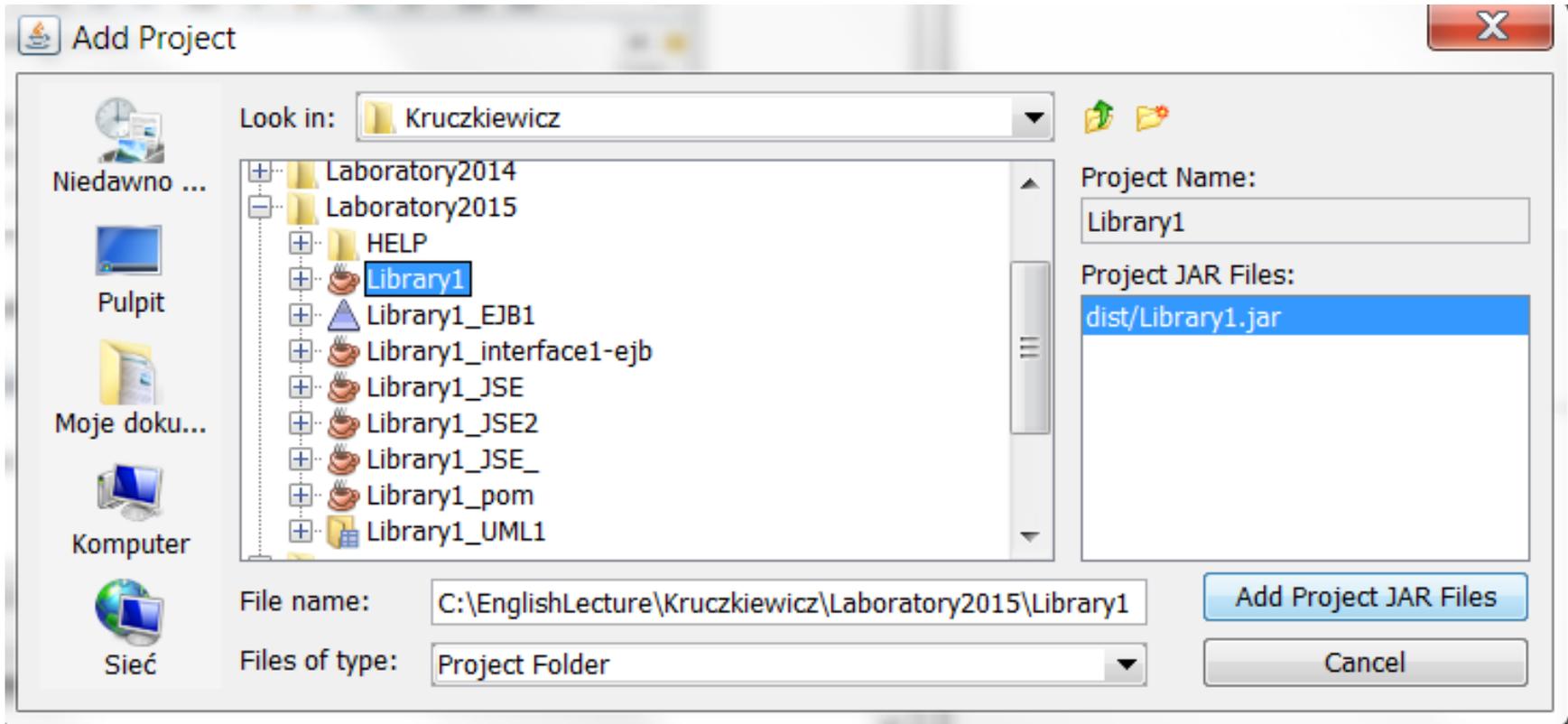
    public TFacade() {mTitle_books = new ArrayList<>();
    }
}
```

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: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4]
```

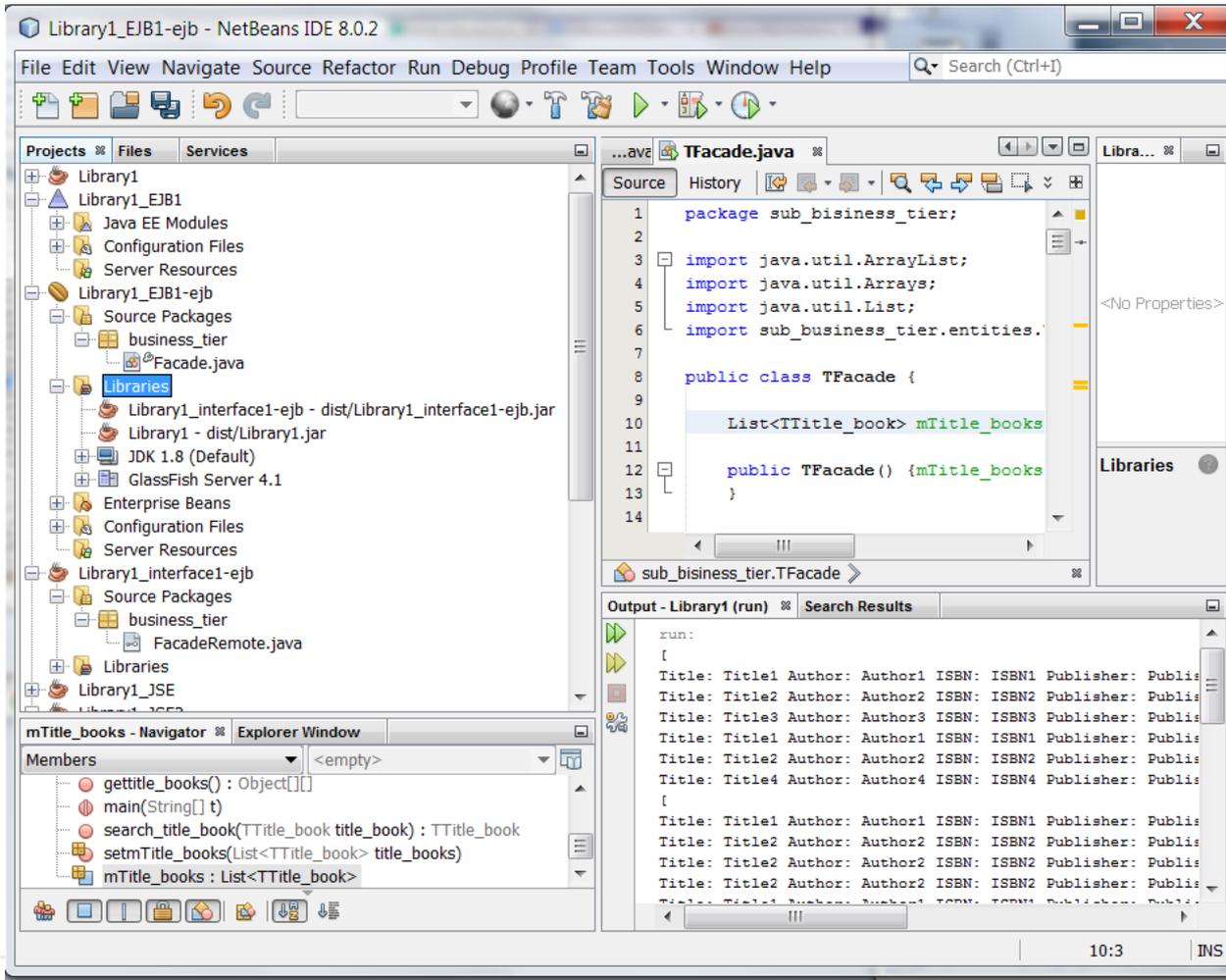


### 5.1.2. Adding the library project (p.4) to the EJB module of EE project - select the Library1 project and click the Add Project JAR Files





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



The screenshot shows the NetBeans IDE interface for a project named 'Library1\_EJB1-ejb'. The 'Projects' window on the left displays the project structure, including 'Library1', 'Library1\_EJB1', and 'Library1\_EJB1-ejb'. The 'Libraries' section is expanded, showing 'Library1 - dist/Library1.jar' and 'JDK 1.8 (Default)'. The 'mTitle\_books - Navigator' window shows the members of the 'mTitle\_books' list, including 'getTitle\_books()', 'main()', 'search\_title\_book()', 'setmTitle\_books()', and 'mTitle\_books'.

The main editor window displays the source code for 'TFacade.java' in the 'sub\_business\_tier' package. The code includes imports for 'java.util.ArrayList', 'java.util.Arrays', 'java.util.List', and 'sub\_business\_tier.entities'. The class 'TFacade' has a constructor that takes a 'List<TTitle\_book>' parameter and assigns it to 'mTitle\_books'.

```
1 package sub_business_tier;
2
3
4 import java.util.ArrayList;
5 import java.util.Arrays;
6 import java.util.List;
7 import sub_business_tier.entities.*;
8
9 public class TFacade {
10     List<TTitle_book> mTitle_books;
11
12     public TFacade() {mTitle_books
13     }
14 }
```

The 'Output - Library1 (run)' window shows the results of a run, displaying a list of titles and authors:

```
run:
[
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publis
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publis
Title: Title3 Author: Author3 ISBN: ISBN3 Publisher: Publis
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publis
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publis
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publis
[
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publis
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publis
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publis
]
```



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

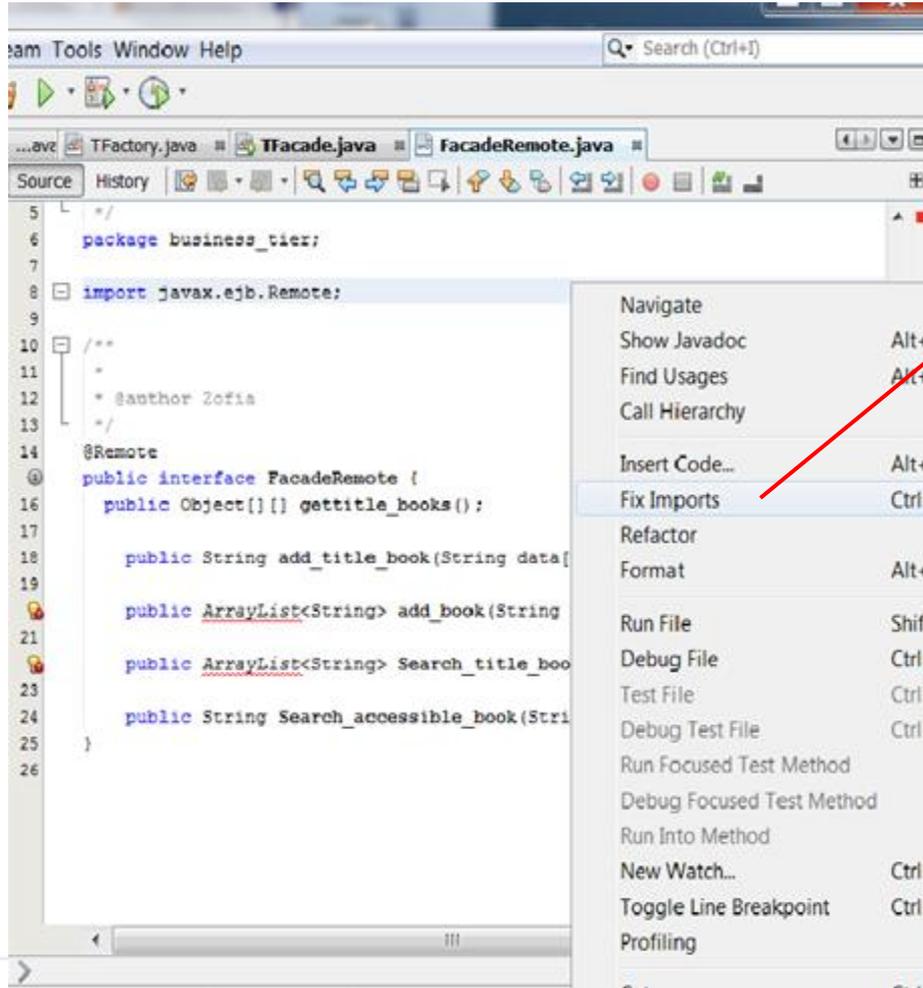
The screenshot shows the NetBeans IDE 8.0.2 interface. The main editor window displays the source code for `FacadeRemote.java`. The code is as follows:

```
5  /*  
6  package business_tier;  
7  
8  import javax.ejb.Remote;  
9  
10 /*  
11  *  
12  * @author Zofia  
13  */  
14 @Remote  
15 public interface FacadeRemote {  
16     public Object[][] gettitle_books();  
17  
18     public String add_title_book(String data[]);  
19  
20     public ArrayList<String> add_book(String data1[], String data2[]);  
21  
22     public ArrayList<String> Search_title_book(String data[]);  
23  
24     public String Search_accessible_book(String data1[], Object data2);  
25 }  
26
```

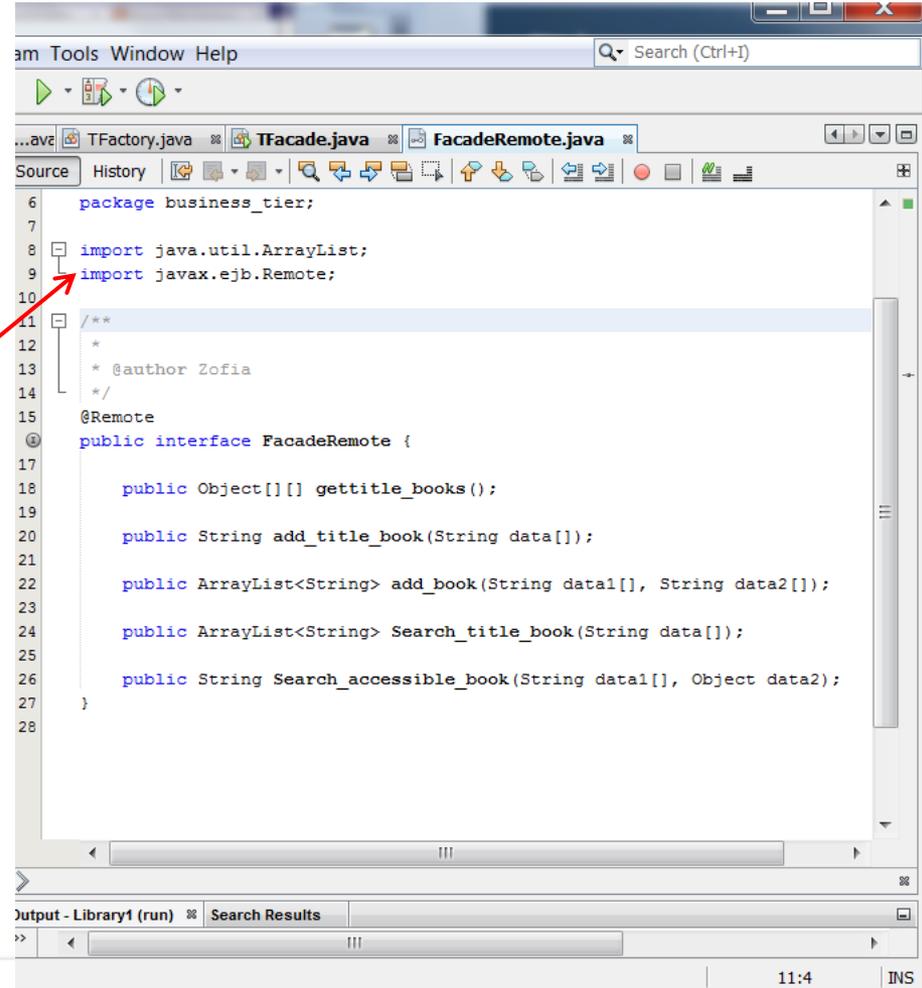
The left sidebar shows the project structure for `Library1`, with `FacadeRemote.java` selected under the `business_tier` package. The bottom-left pane shows the `Members` view for `FacadeRemote`, listing the methods: `Search_accessible_book`, `Search_title_book`, `add_book`, `add_title_book`, and `gettitle_books`. The bottom-right pane shows the `Output` window for `Library1 (run)`.



### 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 Librrary\_EJB1 – ejb module (as the adapter of methods of the POJO TFacade class) – before Fix Imports

The screenshot shows the NetBeans IDE interface. The left sidebar displays the project structure for 'Library1\_EJB1-ejb', with 'Facade.java' selected under the 'business\_tier' package. The main editor window shows the source code of 'Facade.java'.

```

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 code defines a `Facade` class that implements the `FacadeRemote` interface. It uses a `TFacade` object to delegate method calls. The methods include `gettitle_books`, `add_title_book`, `add_book`, `Search_title_book`, and `Search_accessible_book`.

At the bottom of the IDE, the 'Output - Library1 (run)' window is visible, showing 'Search Results'.



### 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 shows the NetBeans IDE interface with the following components:

- Project Explorer:** Shows the project structure for 'Library1', including sub-packages like 'business\_tier' and 'Facade.java'.
- Source Editor:** Displays the code for 'Facade.java' with the following content:
 

```

1  ...5 lines
6  package business_tier;
7
8  import java.util.ArrayList;
9  import javax.ejb.Stateless;
10 import sub_business_tier.TFacade;
11
12 @Stateless
13 public class Facade implements FacadeRemote {
14     TFacade facade = new TFacade();
15     @Override
16     public Object[][] gettitle_books() {
17         return facade.gettitle_books();
18     }
19     @Override
20     public String add_title_book(String data[]) {
21         return facade.add_title_book(data);
22     }
23     @Override
24     public ArrayList<String> add_book(String data1[], String data2[]) {
25         return facade.add_book(data1, data2);
26     }
27     @Override
28     public ArrayList<String> Search_title_book(String data[]) {
29         return facade.Search_title_book(data);
30     }
31     @Override
32     public String Search_accessible_book(String data1[], Object data2) {
33         return facade.Search_accessible_book(data1, data2);
34     }
35 }
      
```
- Facade - Navigator:** Shows the 'FacadeRemote' interface and the 'Search\_accessible\_book' method signature.
- Output Window:** Shows 'Library1 (run)' and 'Search Results'.



```
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);    }  
}
```

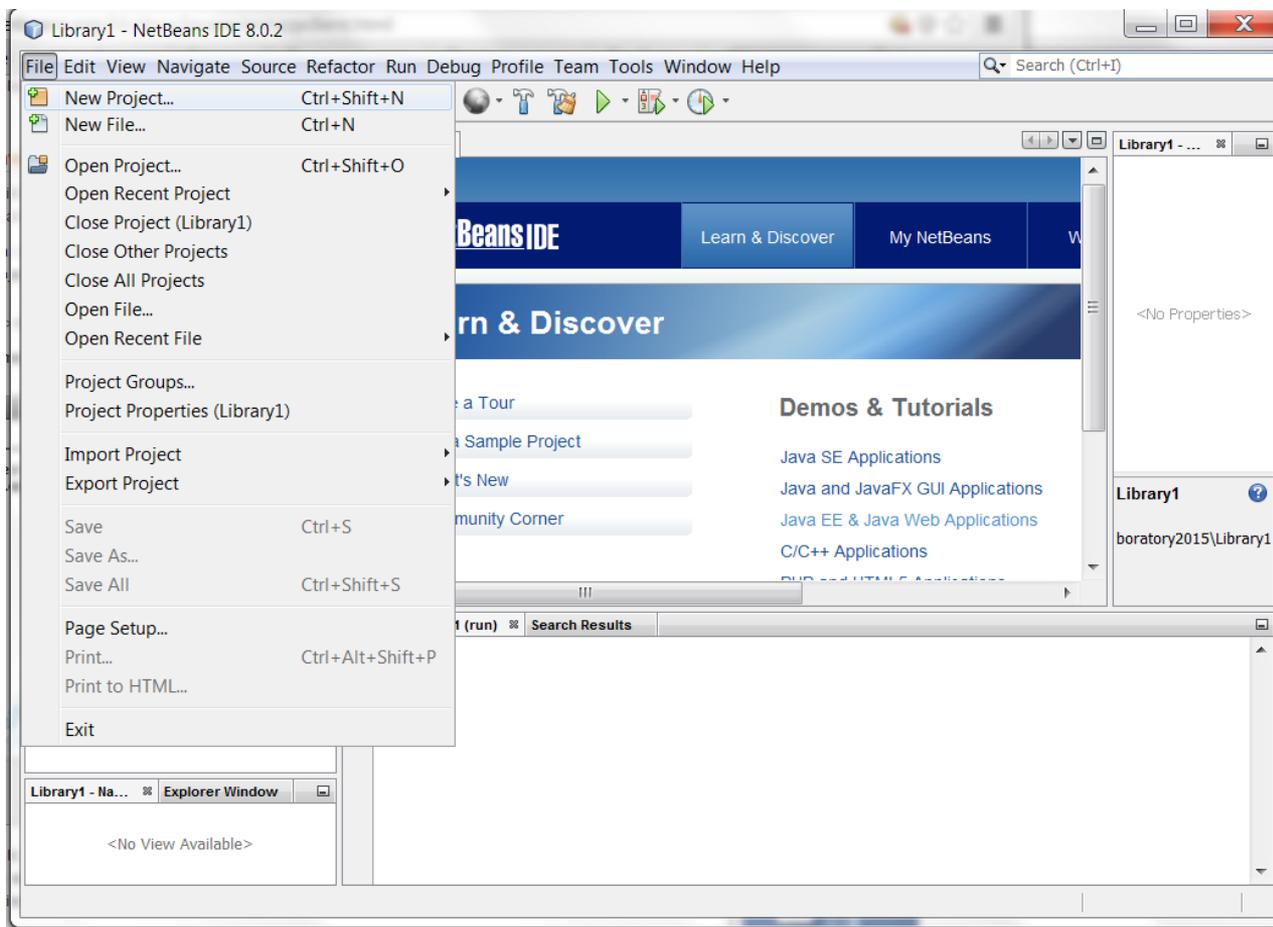
```
    @Override
```

```
    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.**

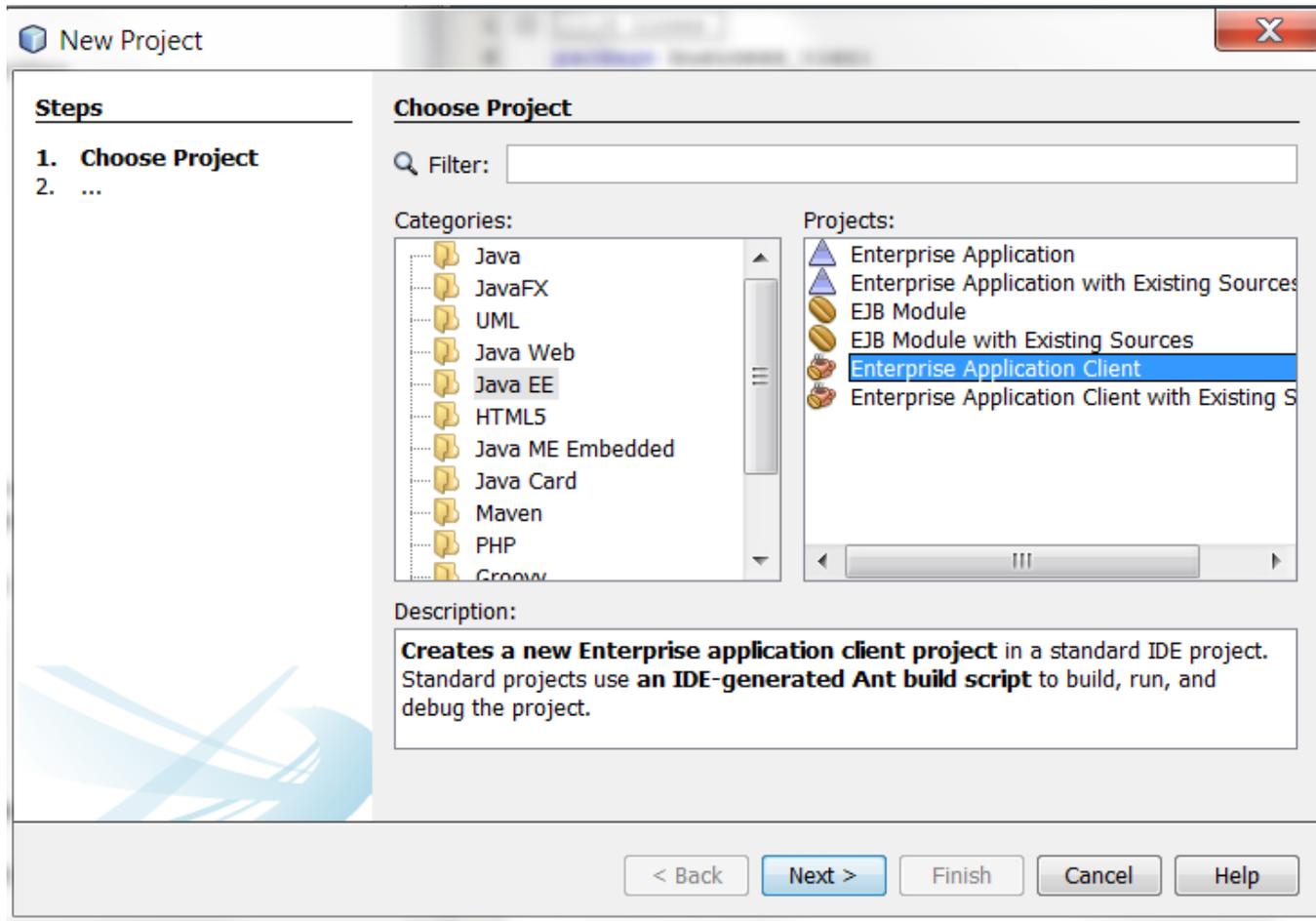


## 6. /6.1. /6.1.1. Creation of the EE client tier – click File/New Project...





### 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.

**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).

< Back   Next >   Finish   Cancel   Help



### 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 interface. The main editor displays the source code of `Client.java` in the `client_tier` package. The code 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 context menu is open over the code, with the following items:

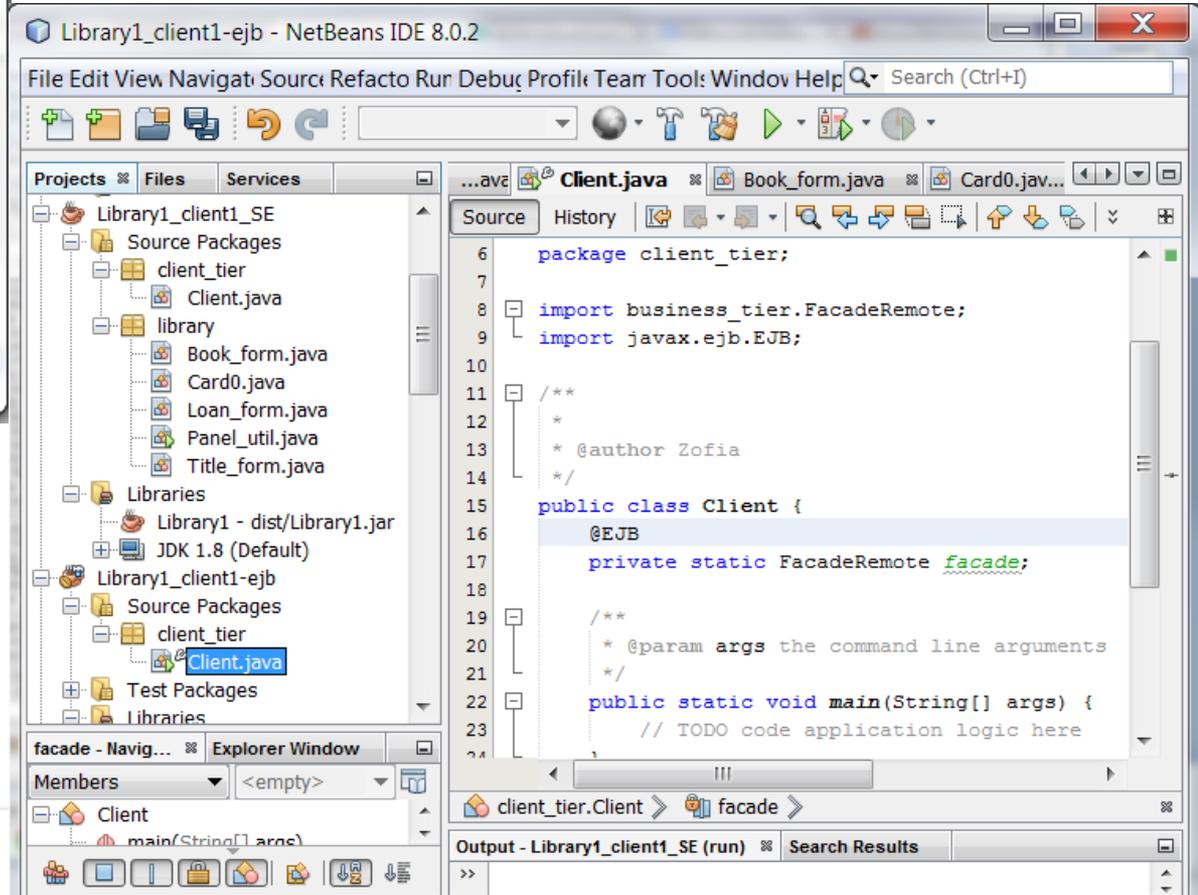
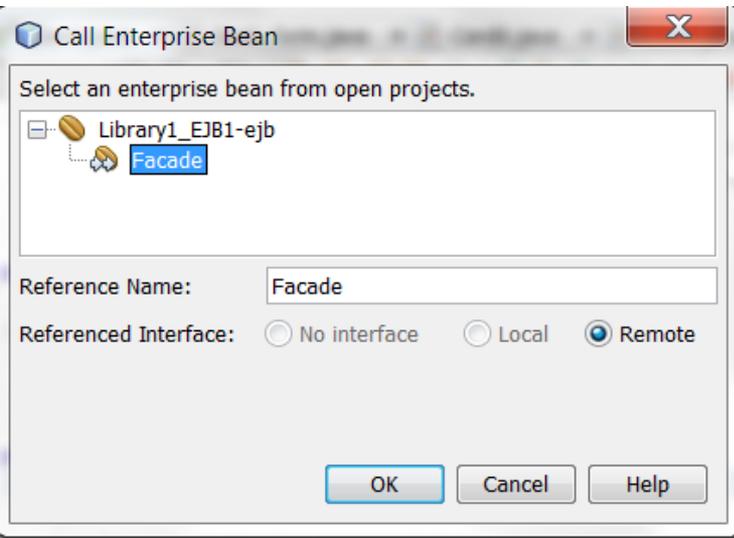
- Navigate
- Show Javadoc (Alt+F1)
- Find Usages (Alt+F7)
- Call Hierarchy
- Insert Code... (Alt+Insert)**
- Fix Imports (Ctrl+Shift+I)
- Refactor
- Format (Alt+Shift+F)
- Run File (Shift+F6)
- Debug File (Ctrl+Shift+F5)
- Test File (Ctrl+F6)
- Debug Test File (Ctrl+Shift+F6)
- Run Focused Test Method
- 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 **Generate** submenu is also visible, with the following items:

- 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...



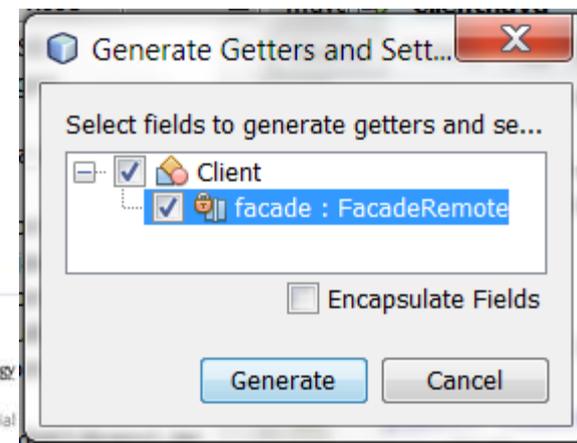
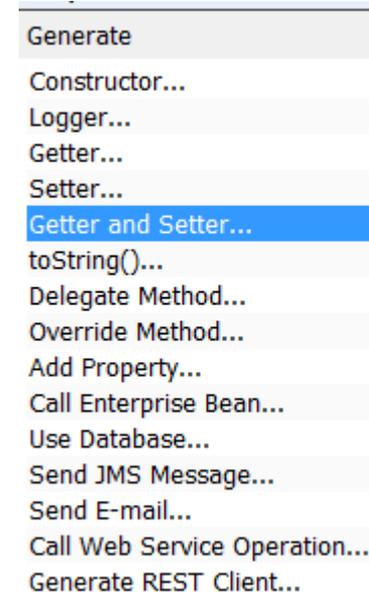
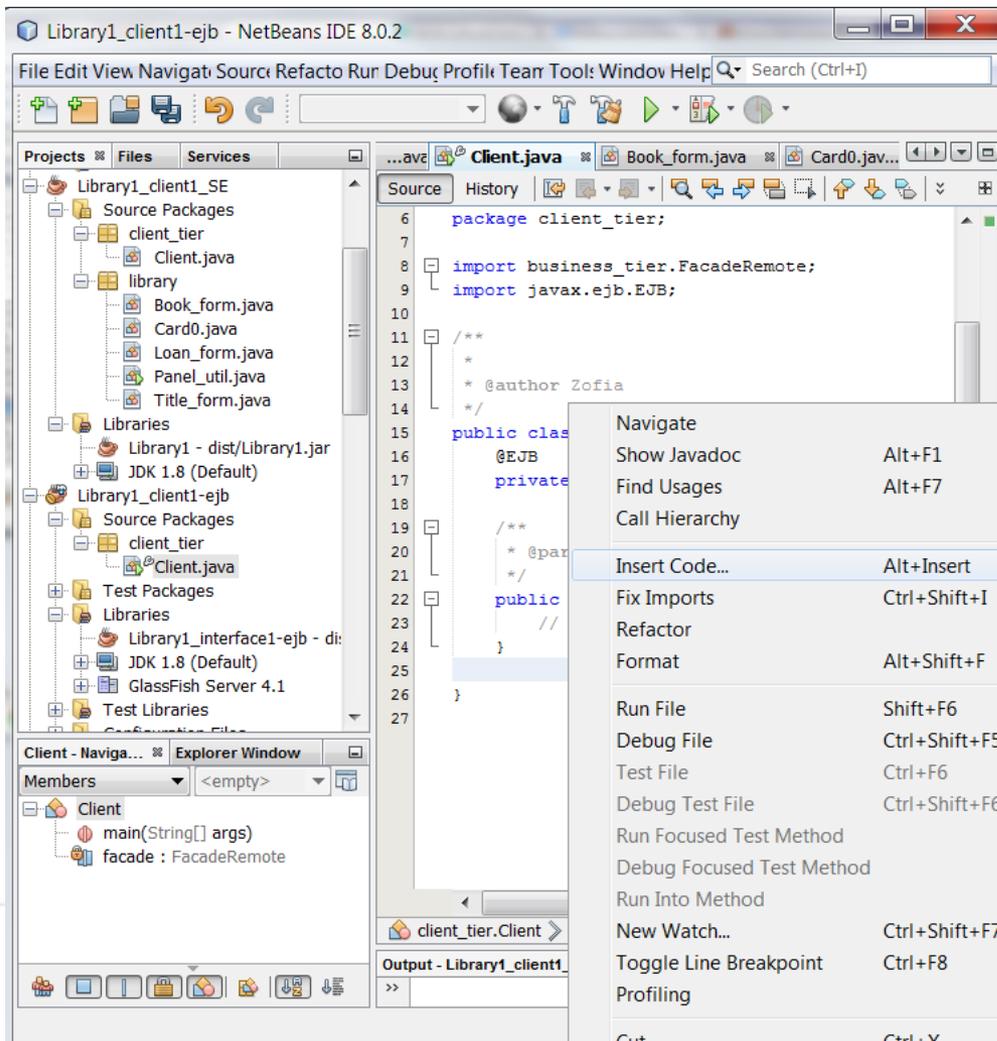
### 6.2.2. Creation of connection from the Facade Session Bean of Library1\_EJB1-ejb module to the Client class of the Library\_client-ejb project - select the Facade EJB from the Library1\_EJB1-ejb module of Library1\_EJB1 project



### 6.2.3. The result of creation the connection to the Facade SessionBean of the Library1\_EJB1-ejb module.

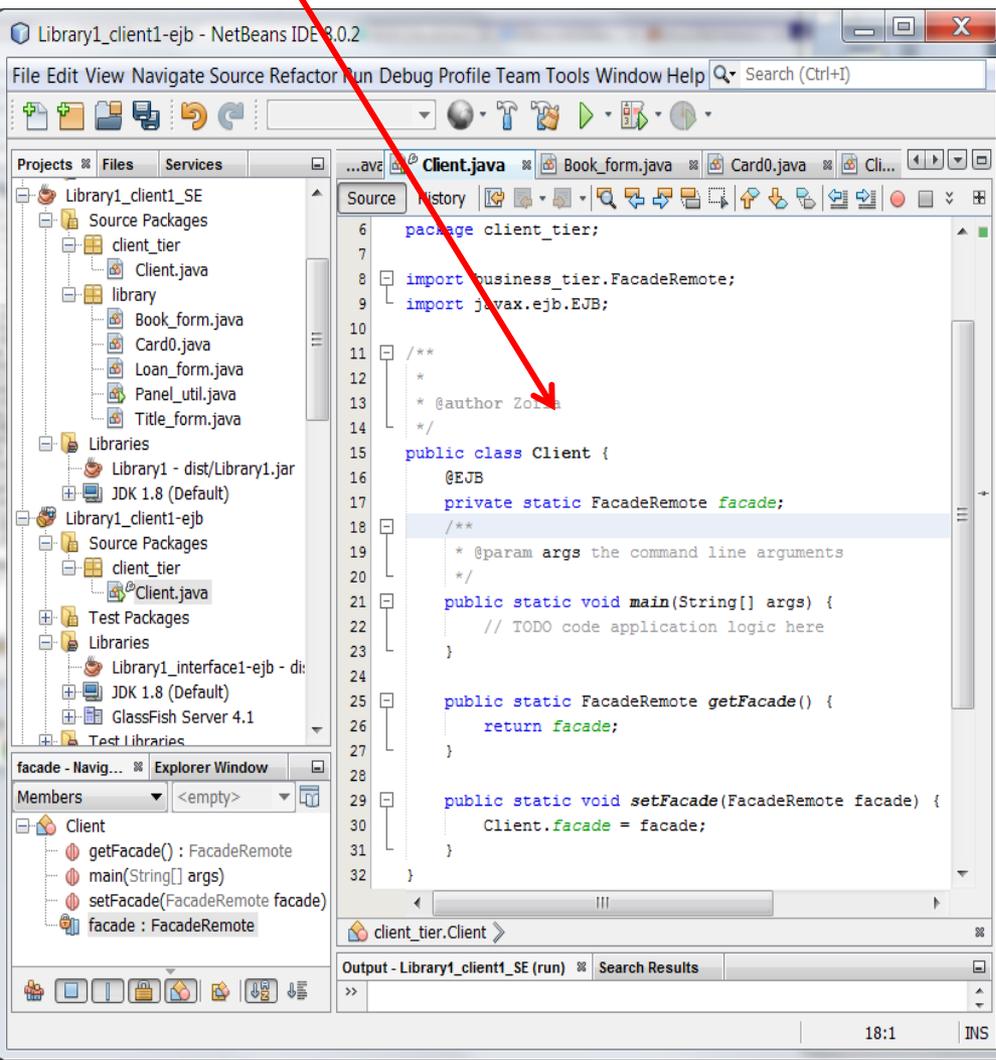


### 6.3. /6.3.1. Adding the getter and setter methods to the facade attribute as the reference to the Facade SessionBean by using the „Insert Code” process – select Insert Code/Getter and Setter... items to open the Generate Getters and Setters dialog and select the facade attribute





### 6.3.2. The result



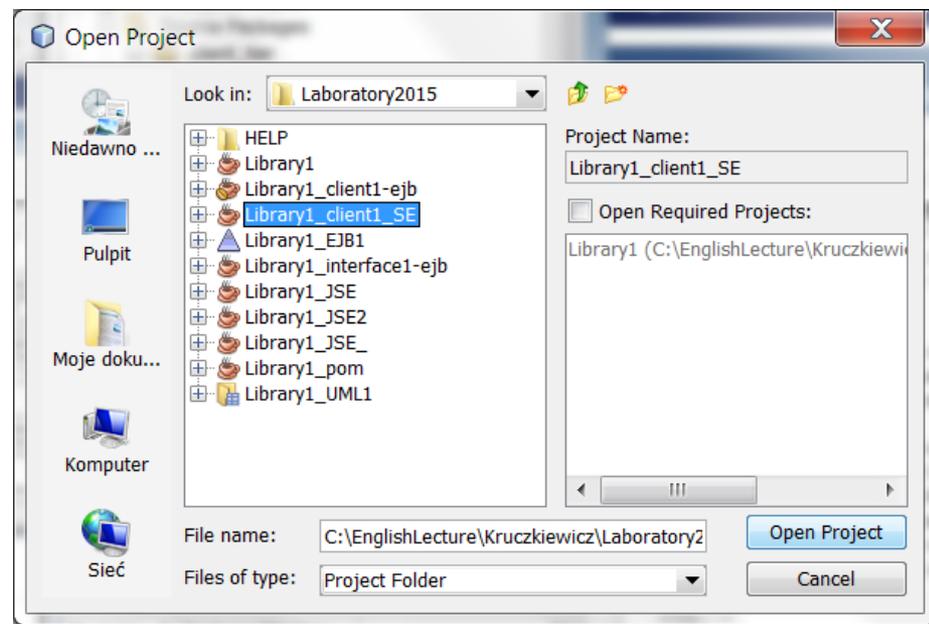
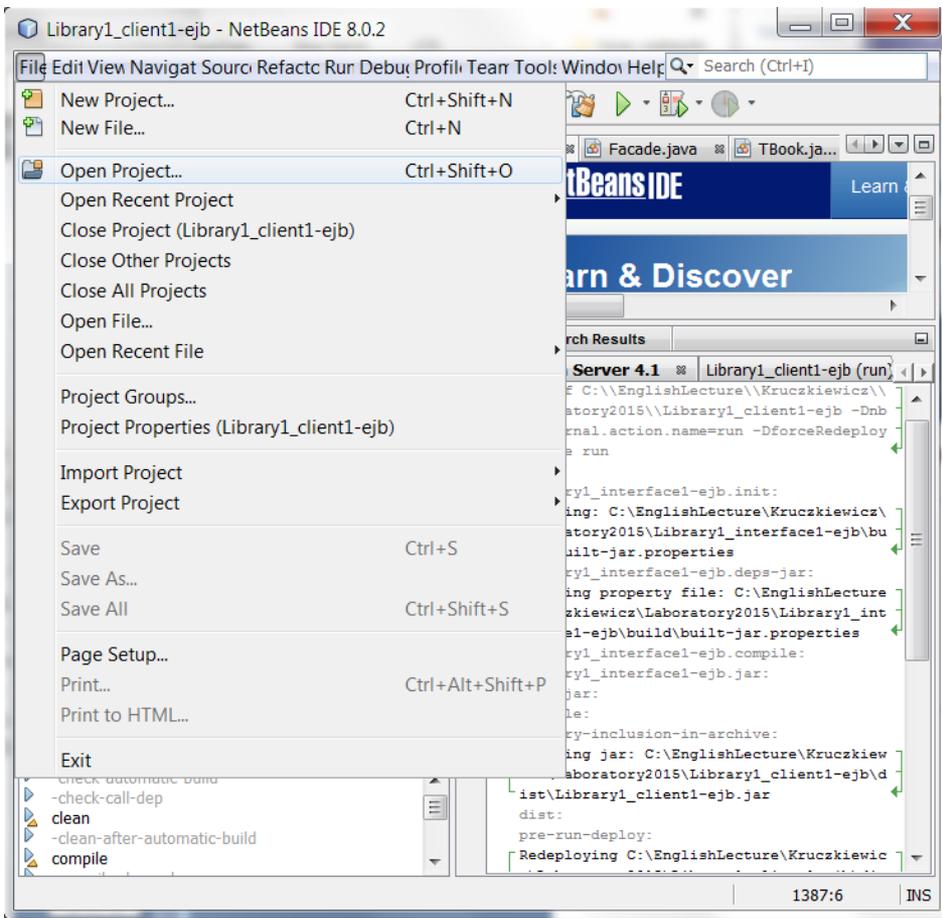
### 6.3.3. Define the body of main method based on code of the Client class main method from the Standard Edition client project named Library1\_client1\_SE

```
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

The screenshot shows the NetBeans IDE interface for the 'Library1\_client1\_SE' project. The 'Projects' window on the left displays a tree view with a context menu open over the 'lib' folder. The 'Copy' option is highlighted. The main editor window shows the source code of 'Panel\_util.java', which includes imports for various Java Swing classes and the definition of a 'Panel\_util' class implementing 'ActionListener'.

### 6.4.3. Paste the library package from the Library1\_client1\_SE to the Library1\_client1-ejb type Enterprise Application Client project

The screenshot shows the NetBeans IDE interface for the 'Library1\_client1-ejb' project. The 'Projects' window on the left displays a tree view with a context menu open over the 'library' folder. The 'Paste' option is highlighted. The main editor window shows the source code of 'Panel\_util.java', which is identical to the one in the previous screenshot, but now includes the 'package library;' declaration at the top.



**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**

The screenshot shows the NetBeans IDE interface. On the left, the 'Projects' window displays a tree view of the project structure. The main editor window shows the source code for 'Panel\_util.java'. The code includes several imports for Java AWT and Swing classes, followed by a class declaration and some static constants.

```

4  /*
5  package library;
6
7
8  import java.awt.BorderLayout;
9  import java.awt.CardLayout;
10 import java.awt.Container;
11 import java.awt.event.ActionEvent;
12 import java.awt.event.ActionListener;
13 import java.awt.event.KeyEvent;
14 import javax.swing.JFrame;
15 import javax.swing.JMenu;
16 import javax.swing.JMenuBar;
17 import javax.swing.JMenuItem;
18 import javax.swing.JPanel;
19 import javax.swing.KeyStroke;
20
21
22 /**
23  * @author Zofia
24  */
25
26 public class Panel_util implements A
27
28     JPanel cards; //a panel that use
29     final static String NOTHING1 = "
30     final static String TITLE = "Tit
31     final static String BOOK = "Book
32     final static String LOAN = "Loan
33
34     public JMenuBar createMennBar()

```

At the bottom of the IDE, the 'Output' window shows the status of the 'GlassFish Server 4.1' and the 'Library1\_client1\_SE (run)' process.



## 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;

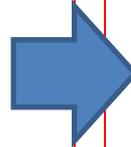
public class Client {

    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



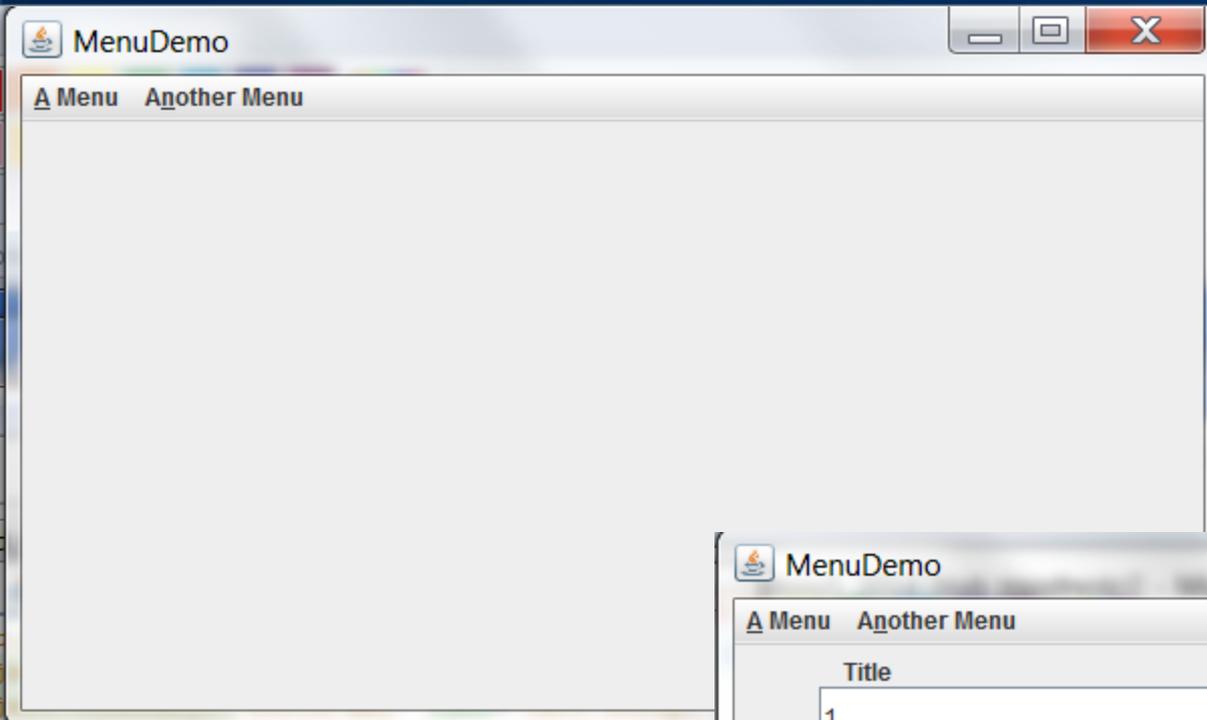
## **6.7. You must add the new panels with forms for supported the new functions (the reservation or the loan of books)**

- 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)**

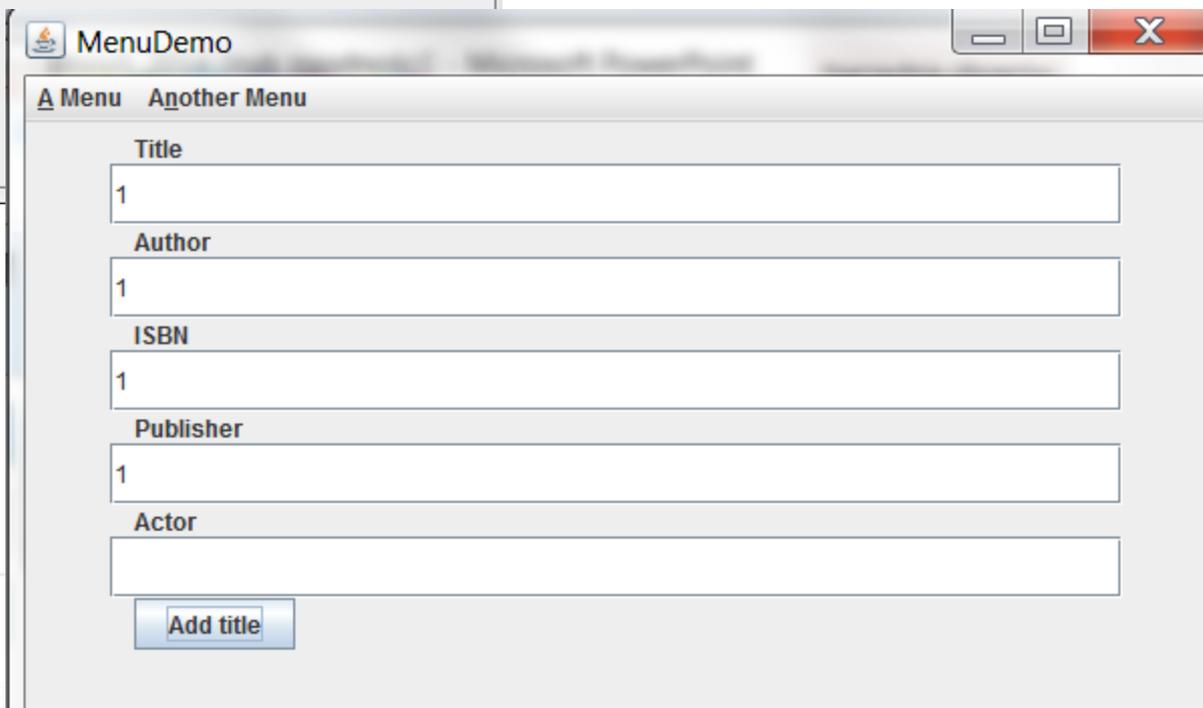


## 7.0. Running the program

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.





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

MenuDemo

A Menu Another Menu

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 the following data:

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

Below the table are three text input fields labeled "Number of a book", "Period of a book", and "Add book" button. At the bottom, there is a list box titled "Books" containing three entries:

- 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 contains a table of contents with the following items:

- 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

The right pane displays the article titled "Using Java EE". The text reads:

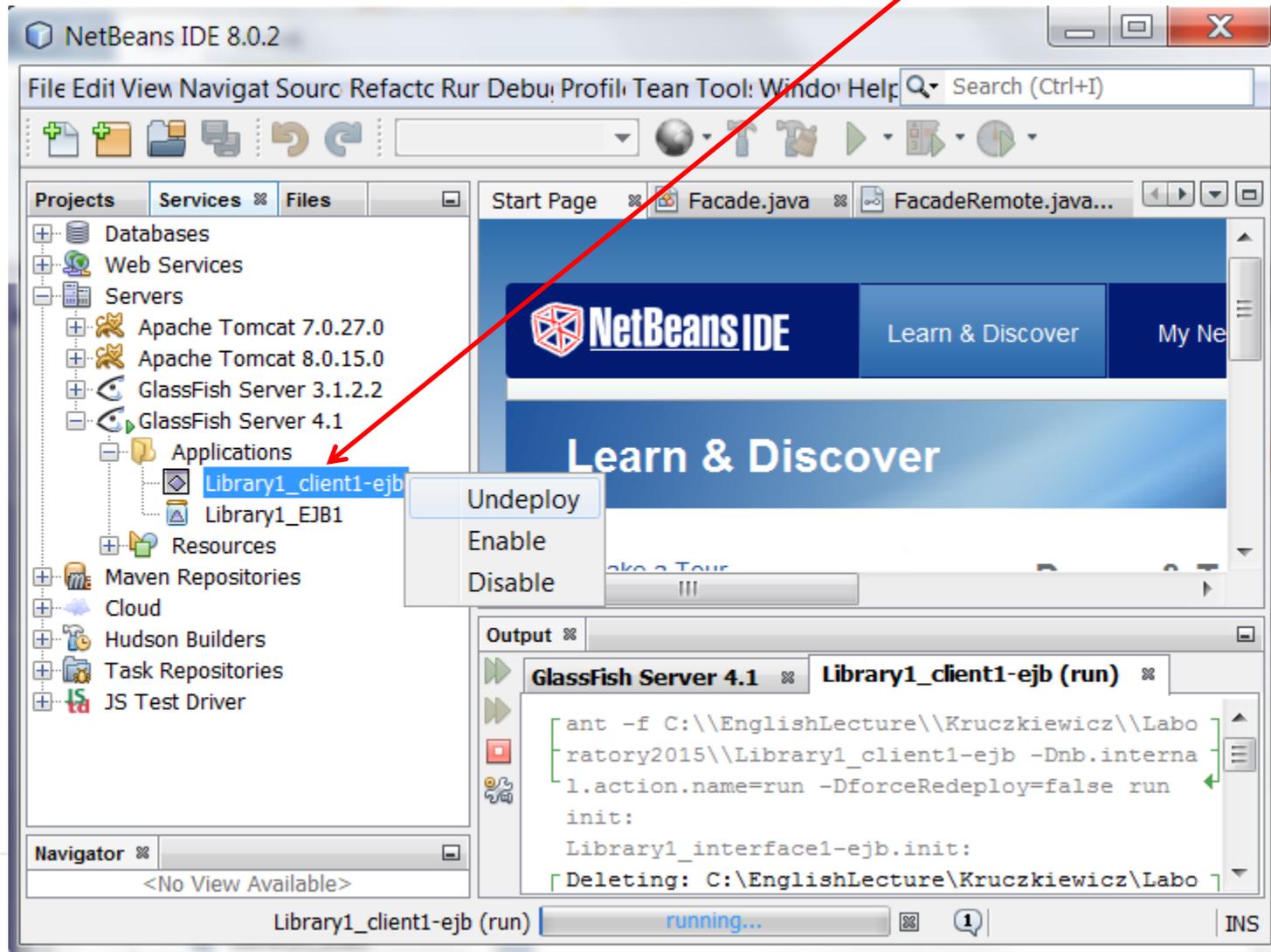
For information on working with Java Enterprise Edition (Java EE) in the IDE, see the following chapter in the User's Guide:

*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

The screenshot shows the NetBeans IDE 8.0.2 interface. The left sidebar contains a tree view of project components. A red arrow points to the 'Applications' folder under 'GlassFish Server 4.1'. The main editor area displays the 'Start Page' with a 'Learn & Discover' banner and buttons for 'Take a Tour', 'Try a Sample Project', and 'What's New'. The bottom output window shows the following log messages:

```
INFO: visiting unvisited references  
[Info: Java Web Start services started for the app client Library1_client1-ejb (contextRoot: /Library1_client1-ejb)  
[Info: Library1_client1-ejb was successfully deployed in 1 084 millise  
conds.  
[Info: Java Web Start services stopped for the app client Library1_client1-ejb
```