Internet Engineering

Tomasz Babczyński, Zofia Kruczkiewicz
Tomasz Kubik

Information systems modelling– UML and service description languages

Laboratory 1

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

Design patterns used to build the Service Subtier of the BusinessTier

D.Alur, J.Crupi, D. Malks, Core J2EE. Desin Patterns

Outline of creating the Library Catalogue Java Application

1. Description of Business
2. The formulation of functional requirements and non-functional system
3. Model analysis of the entire system based on use case diagram
4. Design model of the business service sub-tier based on the class diagram and sequence diagram, created by iteratively driven development of use cases
5. Implementation of the business service sub-tier is created in a series of iteration driven development project model

1. Description of Business

1. **Description of human resources**
   - An library employee may add new titles to the catalog of titles. Each title is represented by the following data: title, author, publisher, ISBN, and the number of copies and their storage place and is present in the library as a single piece of information for each title.
   - Some books are recorded on the tape, so their data of a title also includes additional data, eg name of the actor.
   - Each piece, whether it is a book or a cassette is described in a separate copy number, and also data indicating (it applies to discrete units) information on the number of days on which you can borrow a copy.
   - The numbers of book can be repeated among books of different titles.
   - A librarian can add new titles and copies and search them, and the customer can only browse titles and accessible copies of selected titles.
2. **Provisions**
   - The employee is responsible for the accuracy of the data - is responsible for material non-compliance with the state of the rental.
3. **Technical data**
   - The customer can view data of library through the website or directly through a special program. A library employee can also insert, modify and delete data of titles and copies. It is assumed that customers, searching at the same time the catalogue of the library, may be more than 1,000 and the library can contain some of thousands of titles and at least twice as many copies. The library consists of several centers in various cities across the country (list of cities is included in the contract). It is recommend to use Java technology.

2. List of requirements of the catalog titles and books

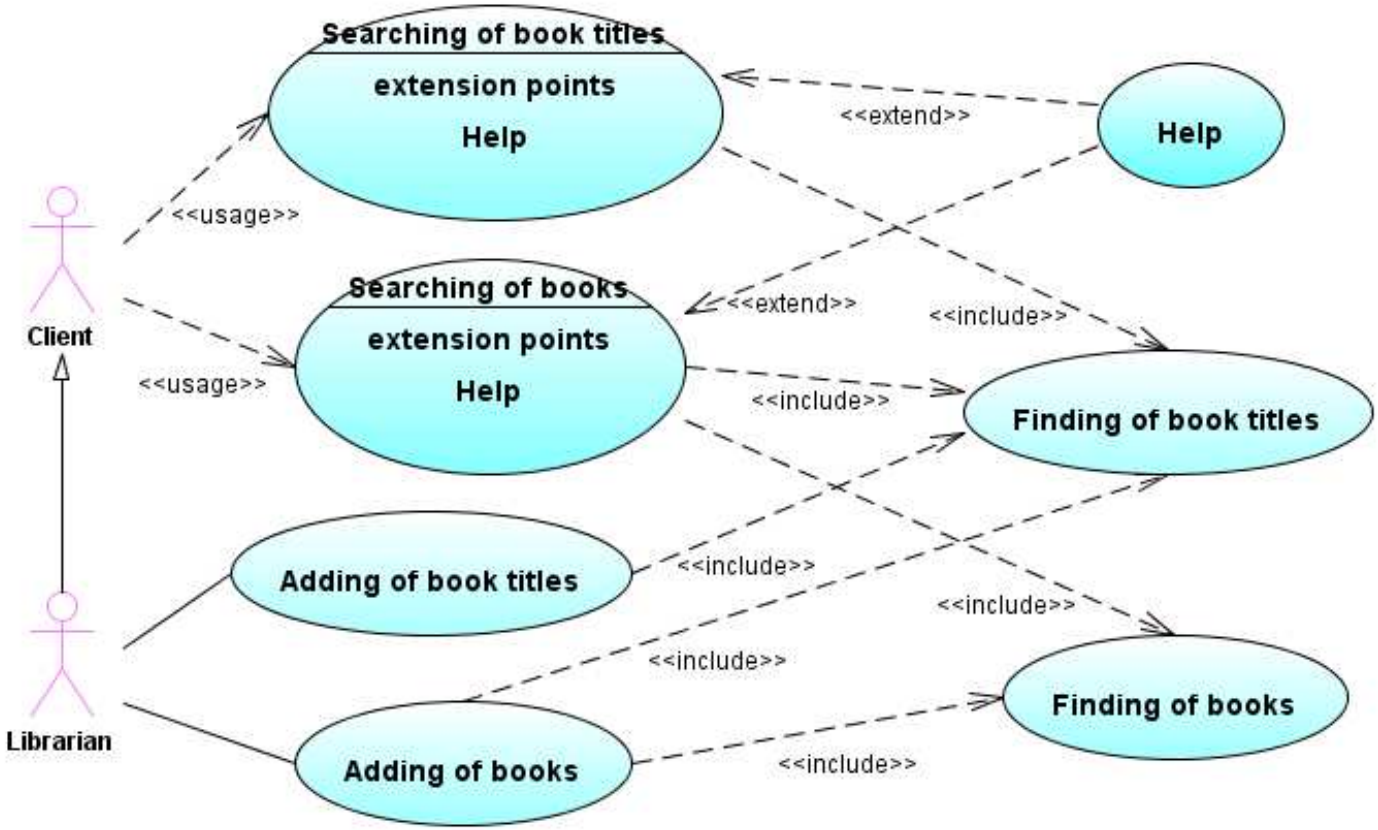
2.1.functional

- The system includes a catalog of titles
- The system includes two types of copies such as books and cassettes with recordings of audio books.
- Each copy contains the title, author name, ISBN, publisher, if this is the book, plus the name of the actor, if it is a sound recording.
- It can be a lot of copies of books and tapes with the same title. Each copy of the book or the cassette has the nonrepeating number of ISBN or the pair data such as ISBN, and the actor's name among all copies.
- In order to select the correct copy, has to give the ISBN, if it is a book and also the name of the actor, when it is a cassette
- Both copies of a book or a cassette, can be spent on borrow at the obligatory period and for a specific period

2.2. nonfunctional

- Inserting data of titles and their copies may be made only by authorized persons
- Searching for information can be done by the client
- Operations management and information retrieval can be made via the Internet or by an application that runs without browser.

3. UseCase Diagram of Library Catalogue



Actor	Description	Use cases
Librarian	*The librarian is responsible for maintaining the resource of the library (inserting and deleting: titles of books, copies of books). It may also browse the directory resources of titles and copies of books*	•Adding of book titles •Adding of book •Searching of titles •Searching of books
Client	*The client can only search the resources of the catalogue of titles and books*	• Searching of titles • Searching of books

UC Finding of book titles

DESCRIPTION
GOAL: Searching of a title
PC (preconditions): initialization by running the program (such as opening a web page, start the application)
FC (final conditions) It gets the new title, or provides information about your lack of result
Scenario

1. Searching for the proceeds according to attributes: ISBN (mandatory) and actor (if required) according to the data given to the use case
2. If there is a title of a given attributes, it returns this title, otherwise it is returned information for lack of a title.

UC Searching of book titles

DESCRIPTION
GOAL: Search titles
PC (preconditions): initialization by running the program (such as opening a web page, start the application)
FC (final conditions) searching titles with the given value of a mandatory attribute ISBN or ISBN, and actor in the case of a cassette with the audio book, or information regarding the lack of result
Scenario:

1. It must be specified the attributes of the title: ISBN as the mandatory value plus actor, if it is looking for a cassette with the audio book. It creates a standard title to search for the real one.
2. It must call **the Searching of book titles use case** to make sure that the title of the specified attributes already exists. If not, the use case finishes without information about the title, otherwise it delivers the title.

UC Finding of books

DESCRIPTION

GOAL: Looking for a book

PC (preconditions): initialization by running the program (such as opening a web page, start the application)

FC (final conditions) providing a book containing the same data as the pattern book or providing information about the lack of the book

SCENARIO:

1. Finding a copy of the proceeds according to the attributes: the number of the book (obligatory) and according to the title given to the use case. It is searching books of the given title.
2. If it finds a book of the specified number, it returns the existing one, otherwise it is returned information of lack of a book.

UC Searching of books

DESCRIPTION

GOAL: Searching for copies of the book with the given title

PC (preconditions): initialization by running the program (such as opening a web page, start the application)

FC (final conditions): It finds a book of the title consistent with the mandatory ISBN attribute, or ISBN and the actor in the case of audio book and the mandatory number of a copy or gives information about the lack of copy

SCENARIO:

1. Please specify the attributes title: ISBN plus actor, if you are looking for the title of the book as the audio book. It creates a standard title to search the real one.
2. It calls **the Finding of book titles use case** to make sure that the title of the specified attributes already exists. If not, it completes the use case not giving information about the title, otherwise it returns the searched title.
3. You must create a pattern of a book which contains the number of the searched book and then you must call **the Finding a book use case**, providing the pattern book. The result given by the executed use case should be given as a final score.

UC Adding of book titles

DESCRIPTION

GOAL: Inserts a new title

PC (preconditions): initialization by running the program (such as opening a web page, start the application)

FC (final conditions): It adds a title which includes the following mandatory attributes: title, author, ISBN, publisher, and if it is an audio book, the actor's name, or information about the existence of such a title

SCENARIO:

1. Provide attributes of the title : title, author, ISBN, publisher, and if it is an audio book, the name of the actor. It creates a title for searching and for the possible insertion.
2. It executes **the Finding od book titles use case** giving the new book title. It makes sure that the title of the specified attribute already exists. If so, it must just finish the use case, otherwise you must insert a new title.

UC Adding of books

DESCRIPTION

GOAL: Adding a new book

PC (preconditions): initialization by running the program (such as opening a web page, start the application)

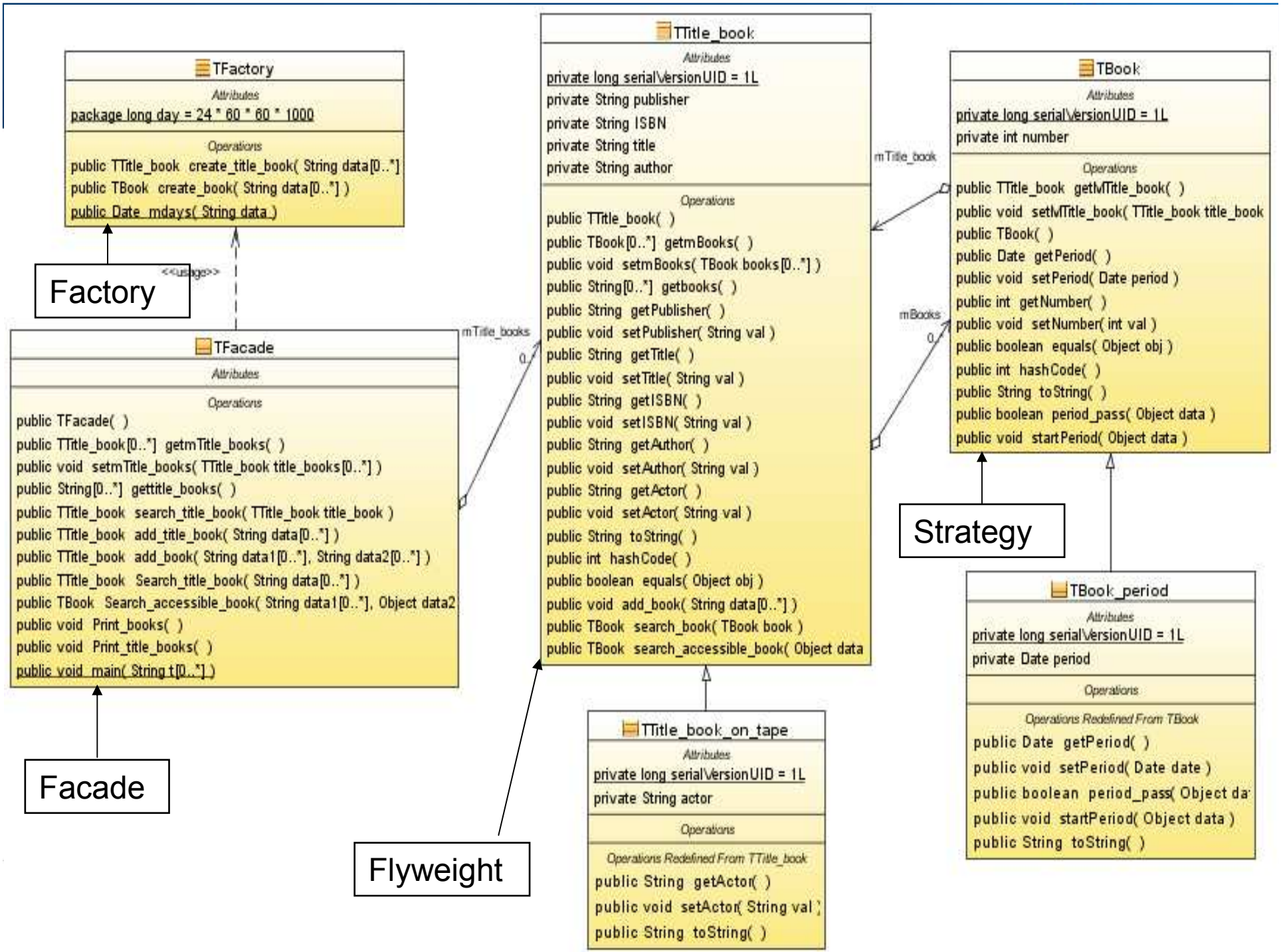
FC (final conditions): It inserts a book consistent with the mandatory attribute ISBN, or ISBN and actor in the case of the audio book and the specified number of the book and possibly the attribute to specifying the date of return, or give information about the existence of such a copy

SCENARIO:

1. It specifies the attributes of the title: ISBN as the mandatory plus actor, if you are looking for is the title of the book as audio book. It creates a pattern title to searching for the real one.
2. It executes **the Finding of book titles use case** providing the pattern title. It makes sure that the title with the specified attribute already exists. If not, it finishes the use case not giving information about the title.
3. Otherwise you must create a book that contains the given number and the date attribute of return, if required, and must pass it to **the Finding of books use case**. If there is not a book with a given number, it adds this book, otherwise you must return the existence of such a copy.

4. An analysis of commonality and variability

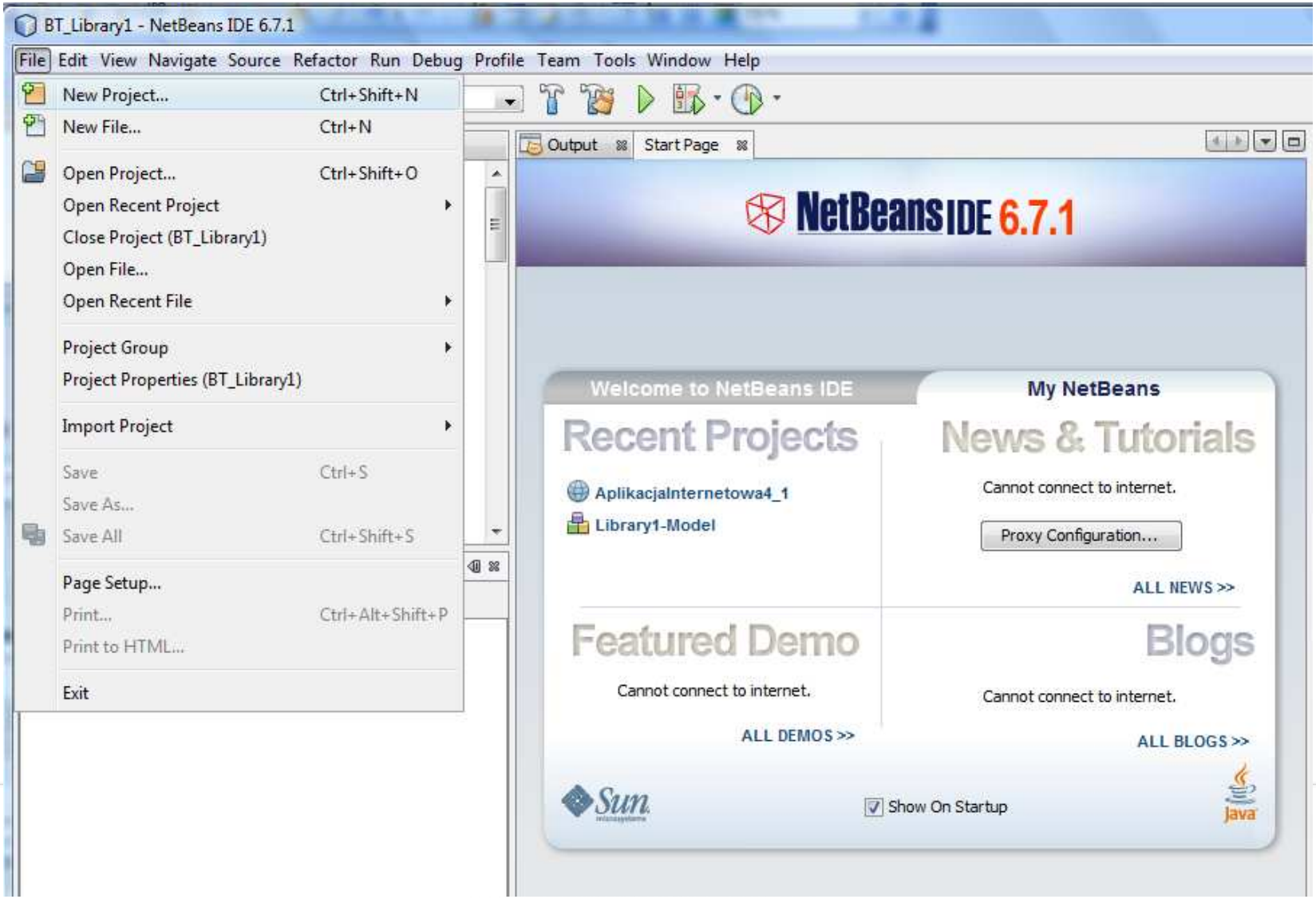
- Detected two main classes of "Entity" for liability reasons: **the TTitle_book class** (it includes attributes of the title, has books -it adds and searches for them) and **the TBook class** (a number). The concepts of the book and the copy of the book are equivalent.
- Inheritance has been detected in the properties of titles that may occur as a simple book or an audio book (**TTitle_book_on_tape class** as the type "Entity", which inherits from **the TTitle_book class**). It is defined a strategy for the storage of the title of multiple copies of books or tapes. **The TBook** object can be borrowed for the standard period and the audio books as **the TBook_period** can be borrowed for specified period, in the period attribute.
- The relationship between **the TTitle_book** and **TBook** objects are in relation 1 to 0 .*. The compound objects **TTitle_book** inherit from **TTitle_book_on_tape** the same relationship 1 to 0 .* From **the TBook class** are inherited **TBook_period class**. Hence the ordinary books can be identified only by numbers or numbers and the date of repayment. This also applies to books in the form of sound recordings.
- Aggregation detects as the strong relationships between the title and a copy - a copy can not exist without title. It can be chosen **the strategy pattern** for the implementation of **TBook** objects.
- **The TFacade class** uses a "Control" as a facade pattern to separate objects of type "Entity" from the rest of the class system and **the TFactory class** as the "Control" object to create different types of titles and copies and to separate the creation and using of objects.


TFactory
Attributes
package long day = 24 * 60 * 60 * 1000
Operations
public TTitle_book create_title_book(String data[0..*])
public TBook create_book(String data[0..*])
public Date mdays(String data)
Factory
<<usage>>
TFacade
Attributes
Operations
public TFacade( )
public TTitle_book[0..*] getmTitle_books( )
public void setmTitle_books(TTitle_book title_books[0..*])
public String[0..*] gettitle_books( )
public TTitle_book search_title_book(TTitle_book title_book)
public TTitle_book add_title_book(String data[0..*])
public TTitle_book add_book(String data1[0..*], String data2[0..*])
public TTitle_book Search_title_book(String data[0..*])
public TBook Search_accessible_book(String data1[0..*], Object data2
public void Print_books( )
public void Print_title_books( )
public void main(String t[0..*])
Facade
mTitle_books
0..*
TTitle_book
Attributes
private long serialVersionUID = 1L
private String publisher
private String ISBN
private String title
private String author
Operations
public TTitle_book( )
public TBook[0..*] getmBooks( )
public void setmBooks(TBook books[0..*])
public String[0..*] getbooks( )
public String getPublisher( )
public void setPublisher(String val)
public String getTitle( )
public void setTitle(String val)
public String getISBN( )
public void setISBN(String val)
public String getAuthor( )
public void setAuthor(String val)
public String getActor( )
public void setActor(String val)
public String toString( )
public int hashCode( )
public boolean equals(Object obj)
public void add_book(String data[0..*])
public TBook search_book(TBook book)
public TBook search_accessible_book(Object data
mTitle_book
TBook
Attributes
private long serialVersionUID = 1L
private int number
Operations
public TTitle_book getMTitle_book( )
public void setMTitle_book(TTitle_book title_book
public TBook( )
public Date getPeriod( )
public void setPeriod(Date period)
public int getNumber( )
public void setNumber(int val)
public boolean equals(Object obj)
public int hashCode( )
public String toString( )
public boolean period_pass(Object data)
public void startPeriod(Object data)
mBooks
0..*
Strategy
TTitle_book_on_tape
Attributes
private long serialVersionUID = 1L
private String actor
Operations
Operations Redefined From TTitle_book
public String getActor( )
public void setActor(String val)
public String toString( )
Flyweight
TBook_period
Attributes
private long serialVersionUID = 1L
private Date period
Operations
Operations Redefined From TBook
public Date getPeriod( )
public void setPeriod(Date date)
public boolean period_pass(Object da
public void startPeriod(Object data)
public String toString( )

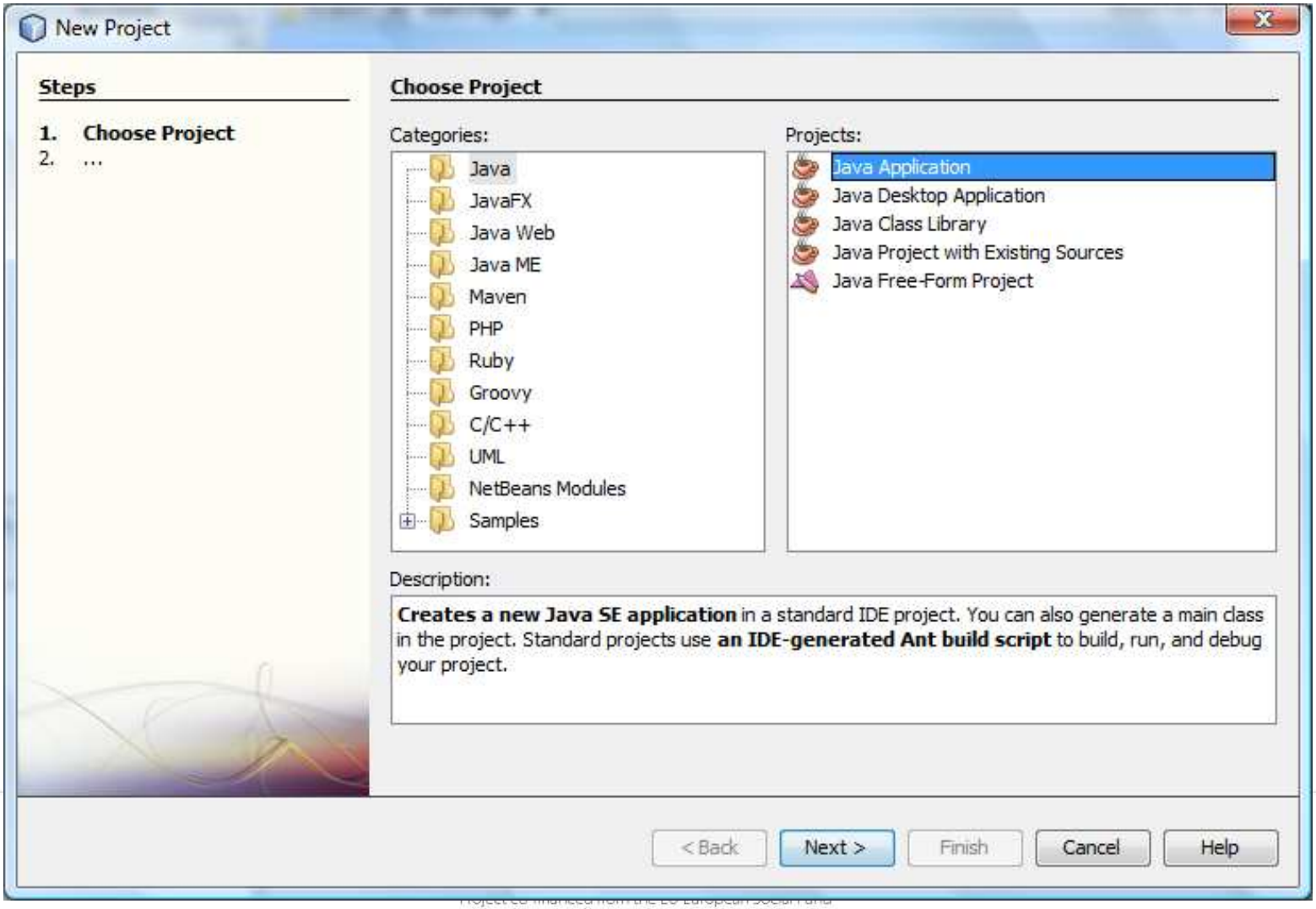
5. Generate code classes based on class diagram developed in the analysis phase (next 9 slides)

1. You must open the UMLproject called BT-Library1-Model2, provided in an annex to the laboratory
2. It must be created a new Java project such as the Java Application project named Code-Library1 (You must chose in the following order: the File, New Project, Java, Java Application, Next, insert the name of the project as the Code-Library1 name, select a folder in the Project Location where the project is BT-Library1-Model2, Finish)
3. You should generate code from the library1 package (in the Project Window right click the library1 in the Model part of the BT-Library1-Model2 and select the Generate Code item from the pop-up menu).
4. In the Generate Code form, you must select the Target Project as the Code-Library1, then select the Backup Existing Source Files, and the Prompt Before Generating Code and press OK.
5. In two files: TFacade.java and TTitle_book you must right click in the Java editor and from the pop-up menu you must click the Fix Import. You can click right in the Java editor and click Format item for organizing the code (indentation, free lines of code blocks, etc).

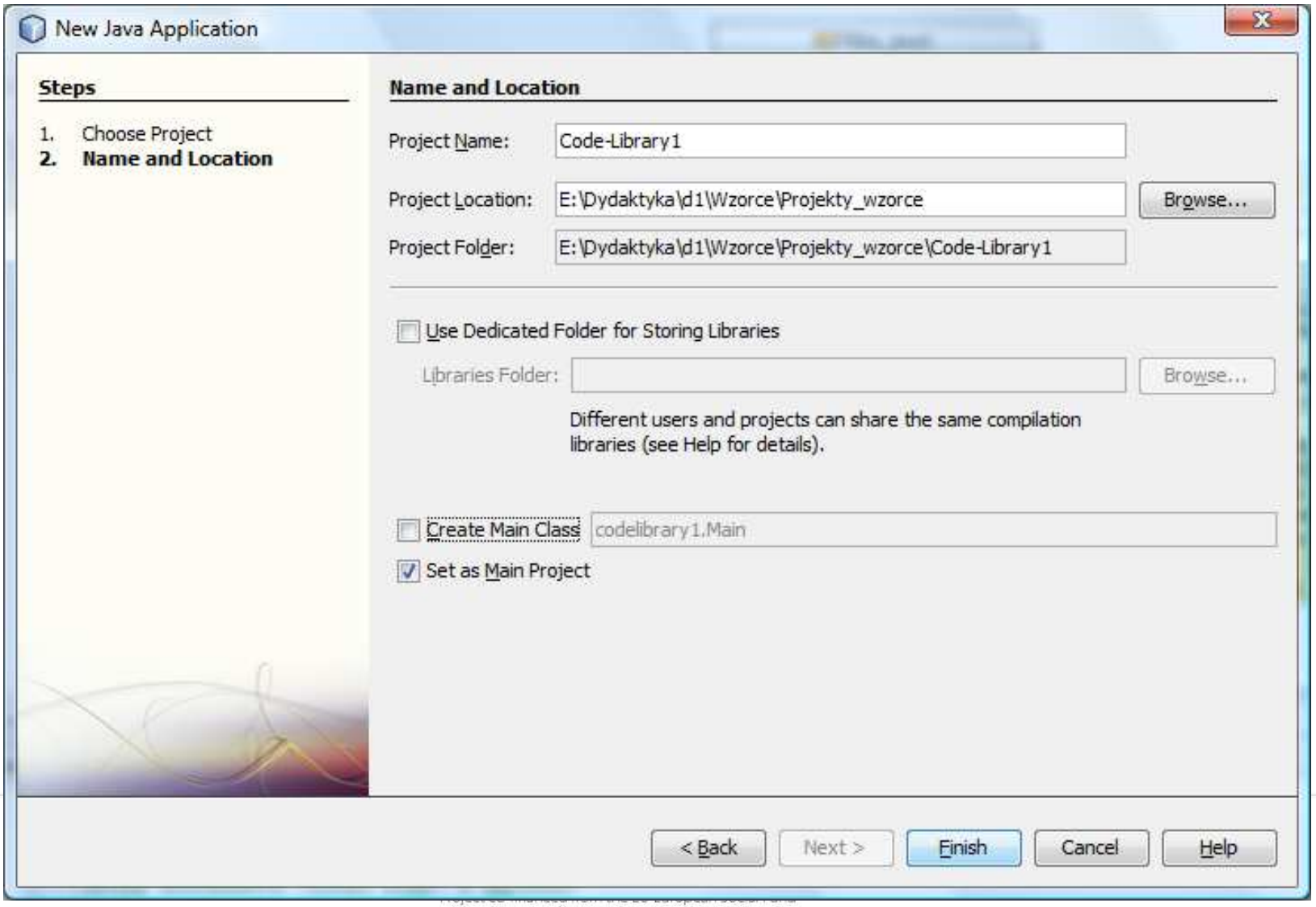
Create empty Java Application Project (2)



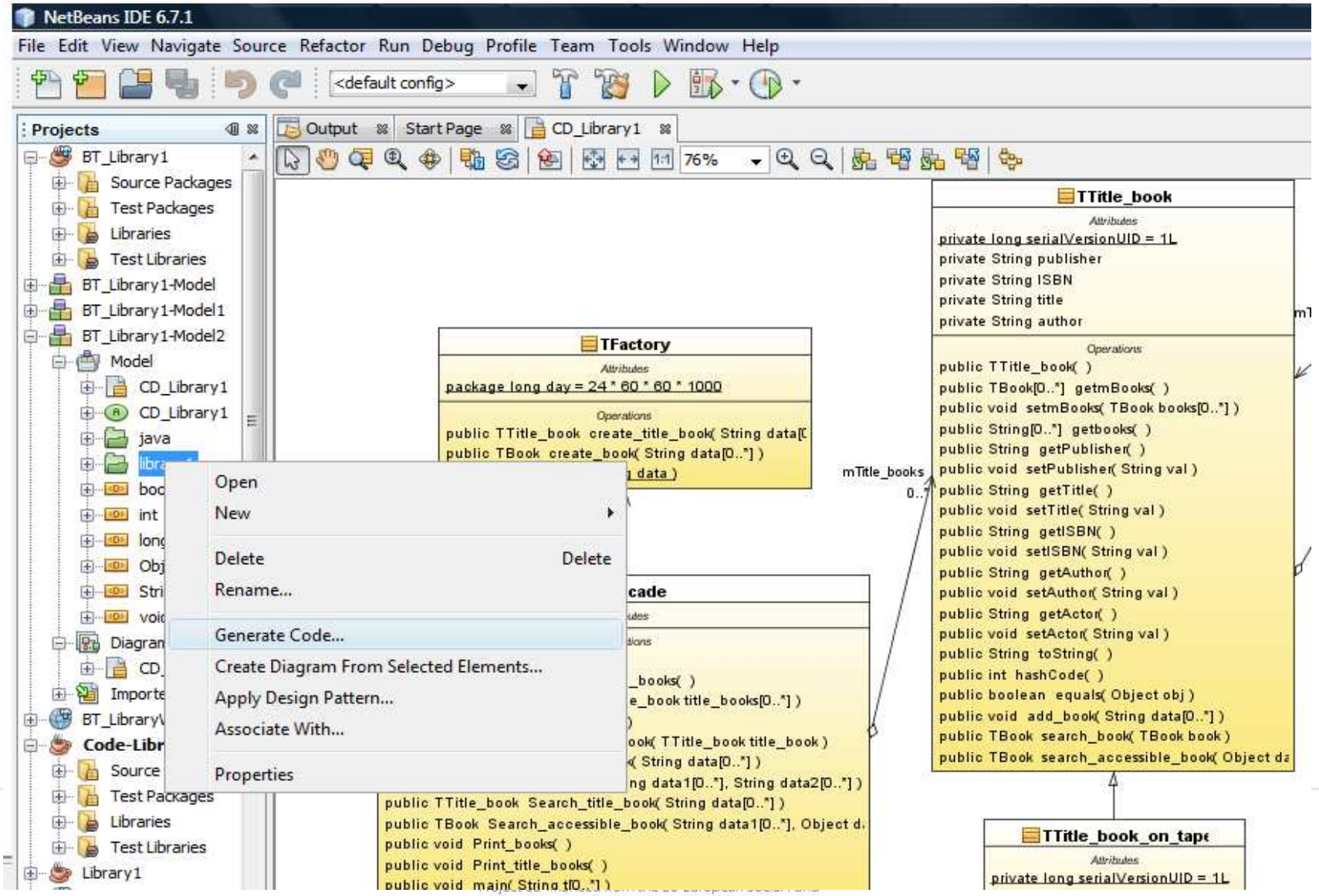
Create empty Java Application Project (2)



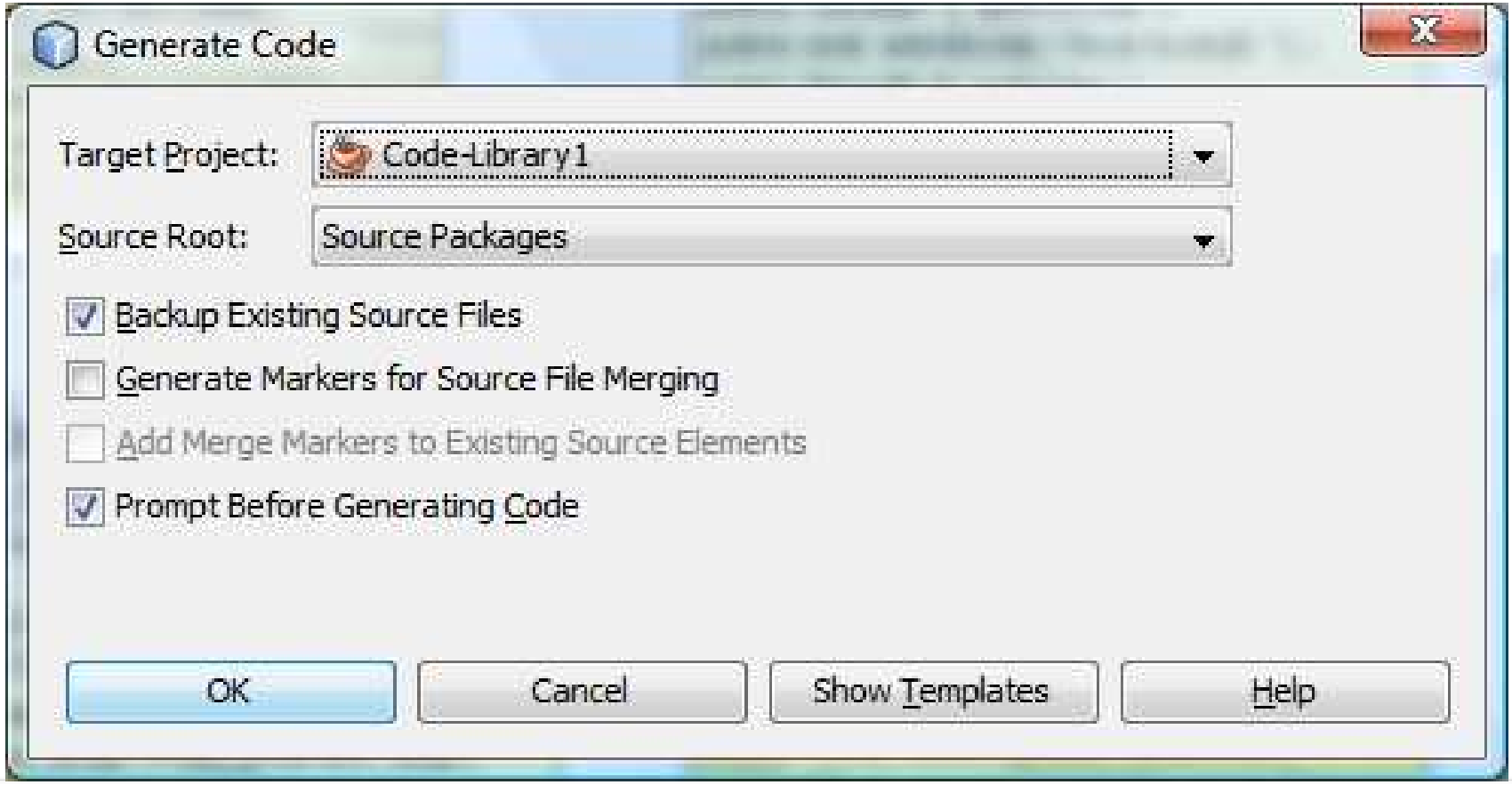
Create empty Java Application Project (2)



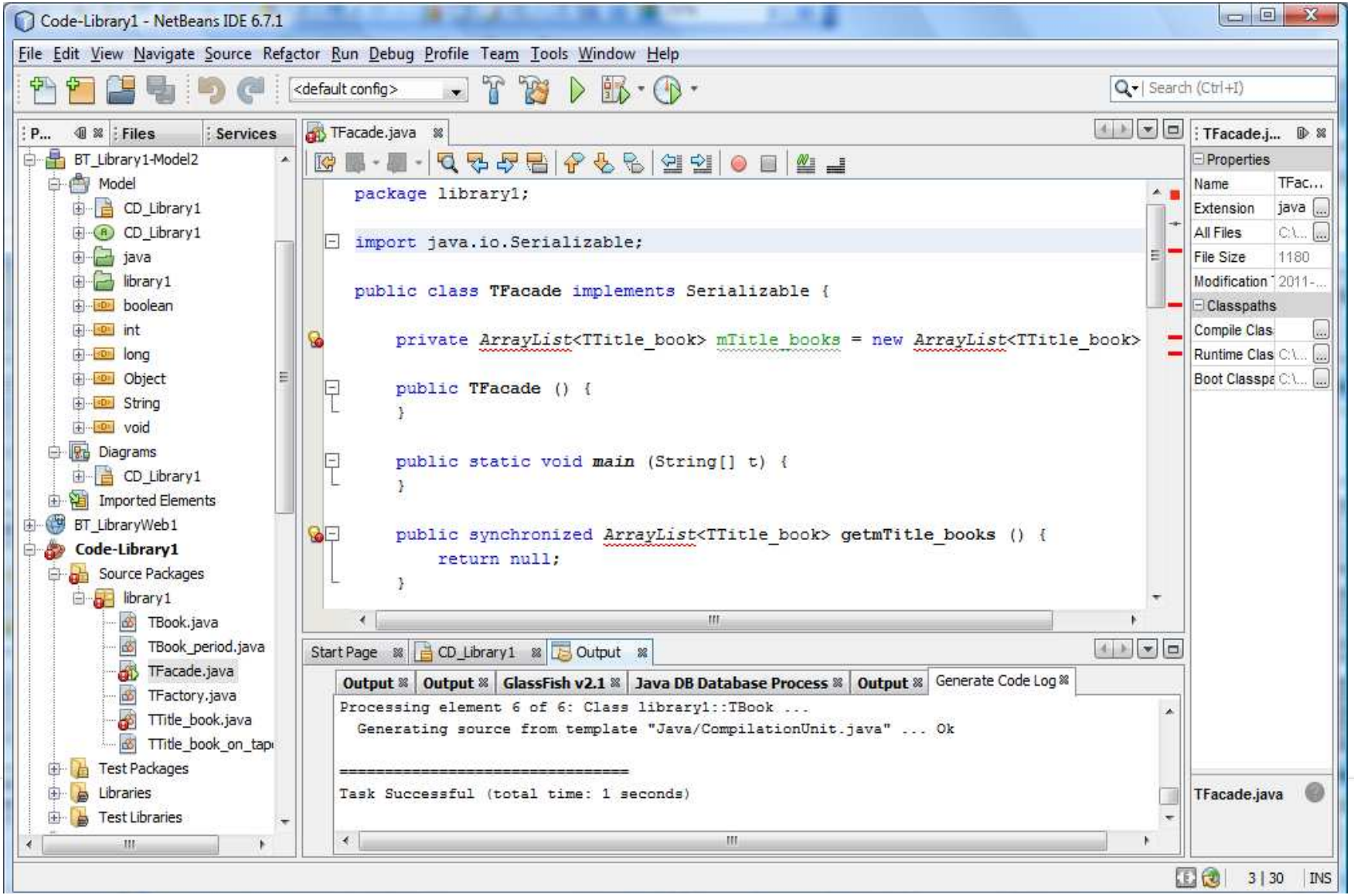
Generating code from the library1 package (3)



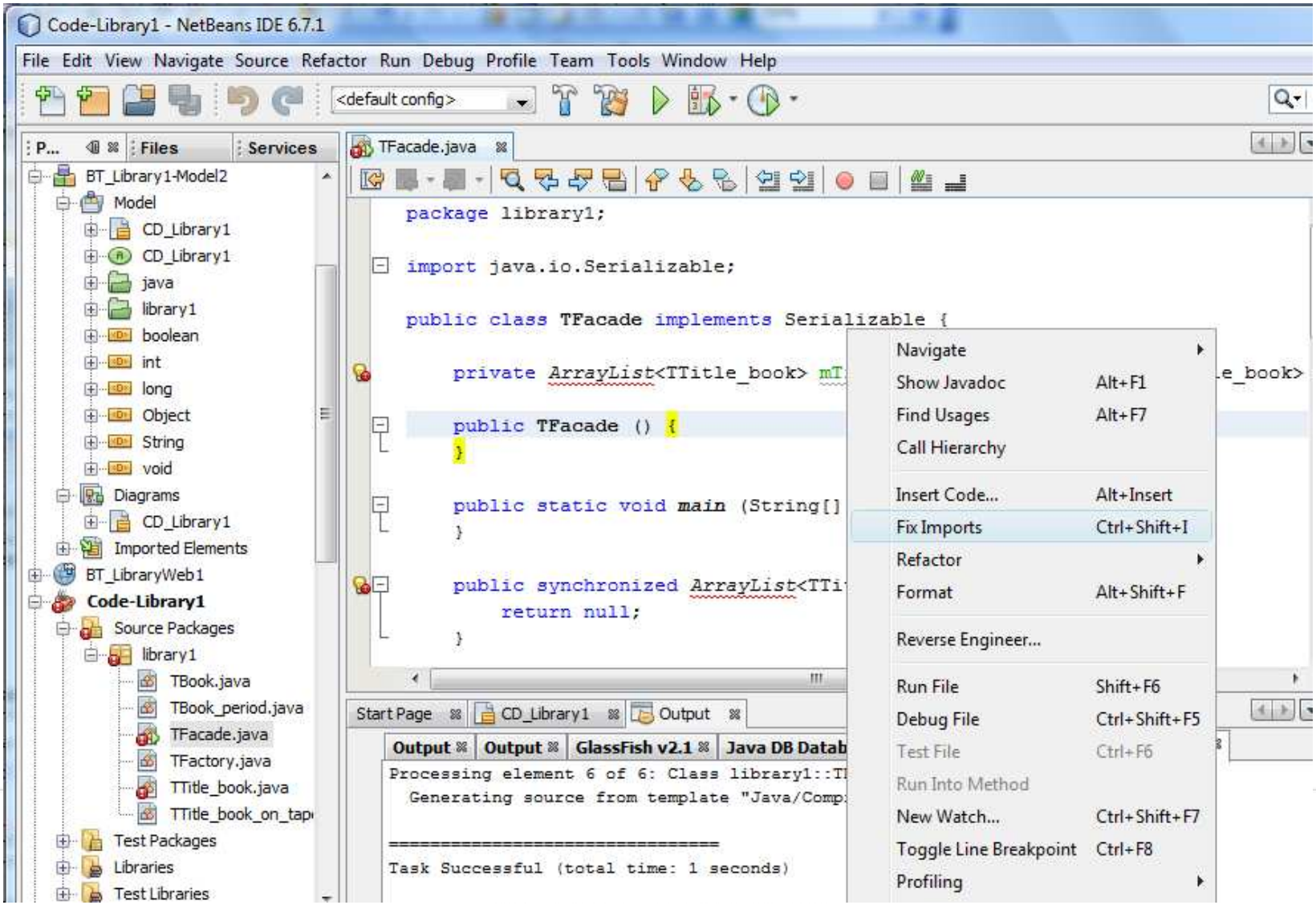
Selection of the Code-library1 as the Java Application project (4)



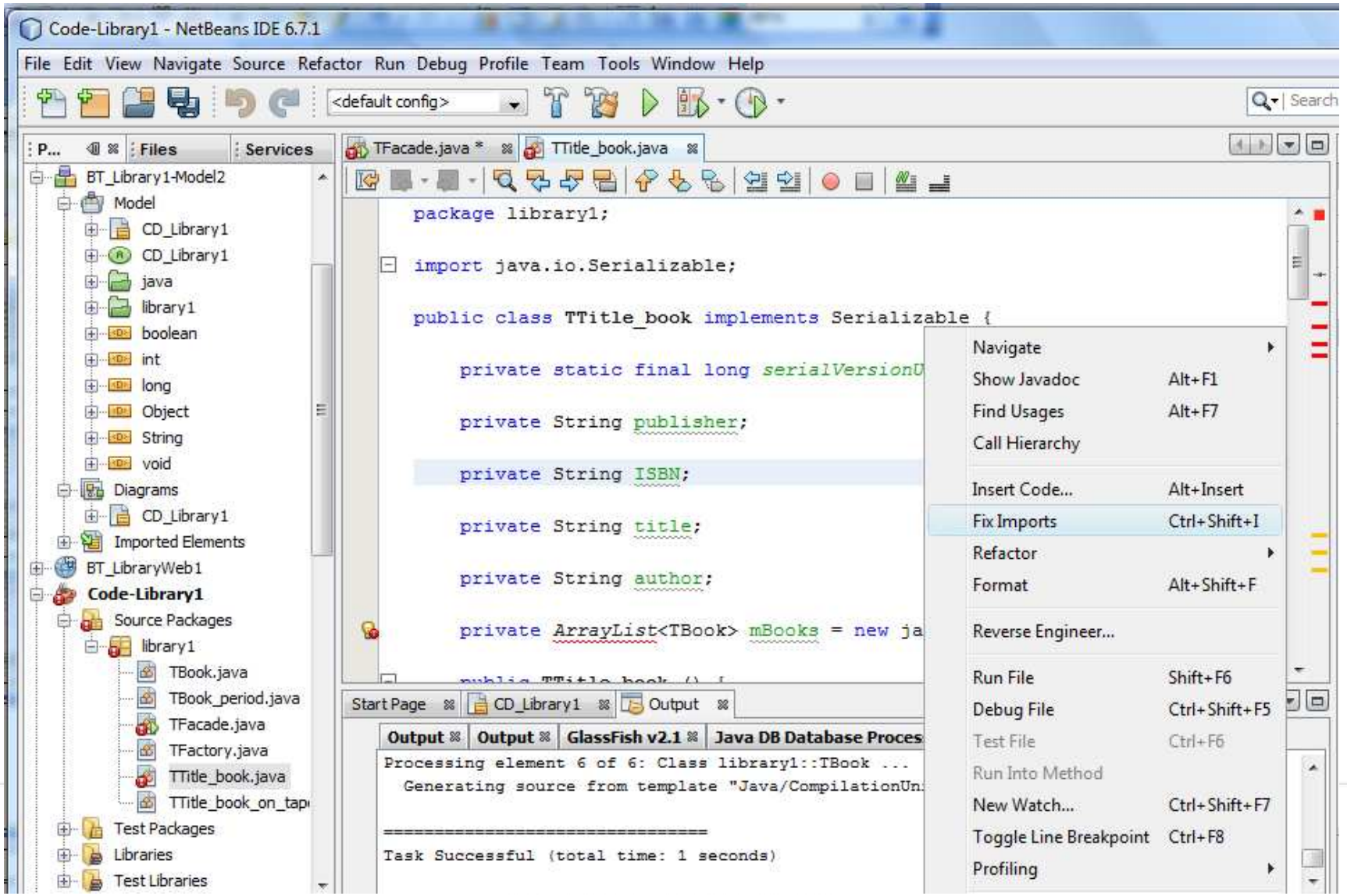
Faults because of lacking of imports of package (next slide)



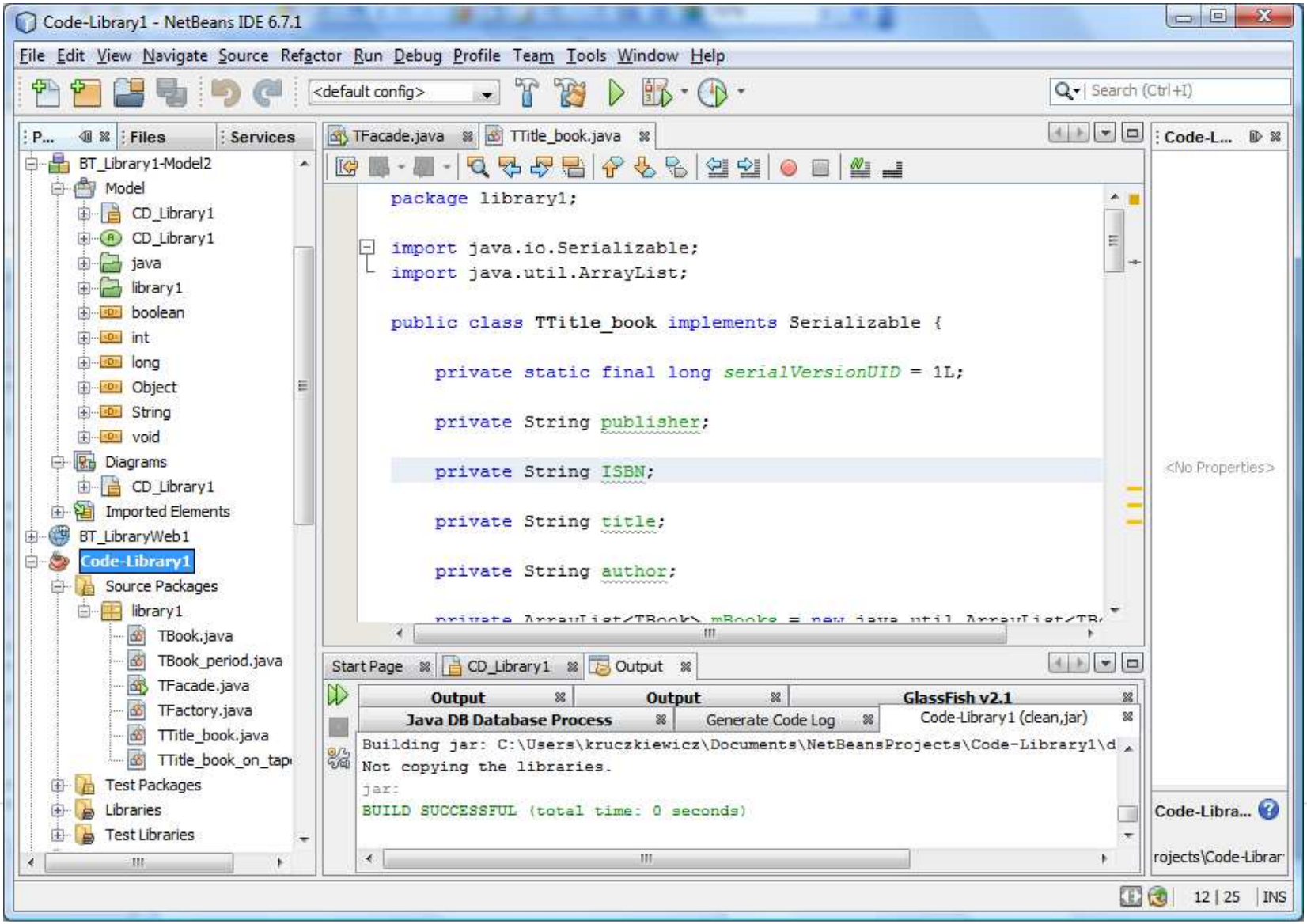
Fix imports in Tfacade.java (5)



Fix imports in Ttitle_book.java (5)



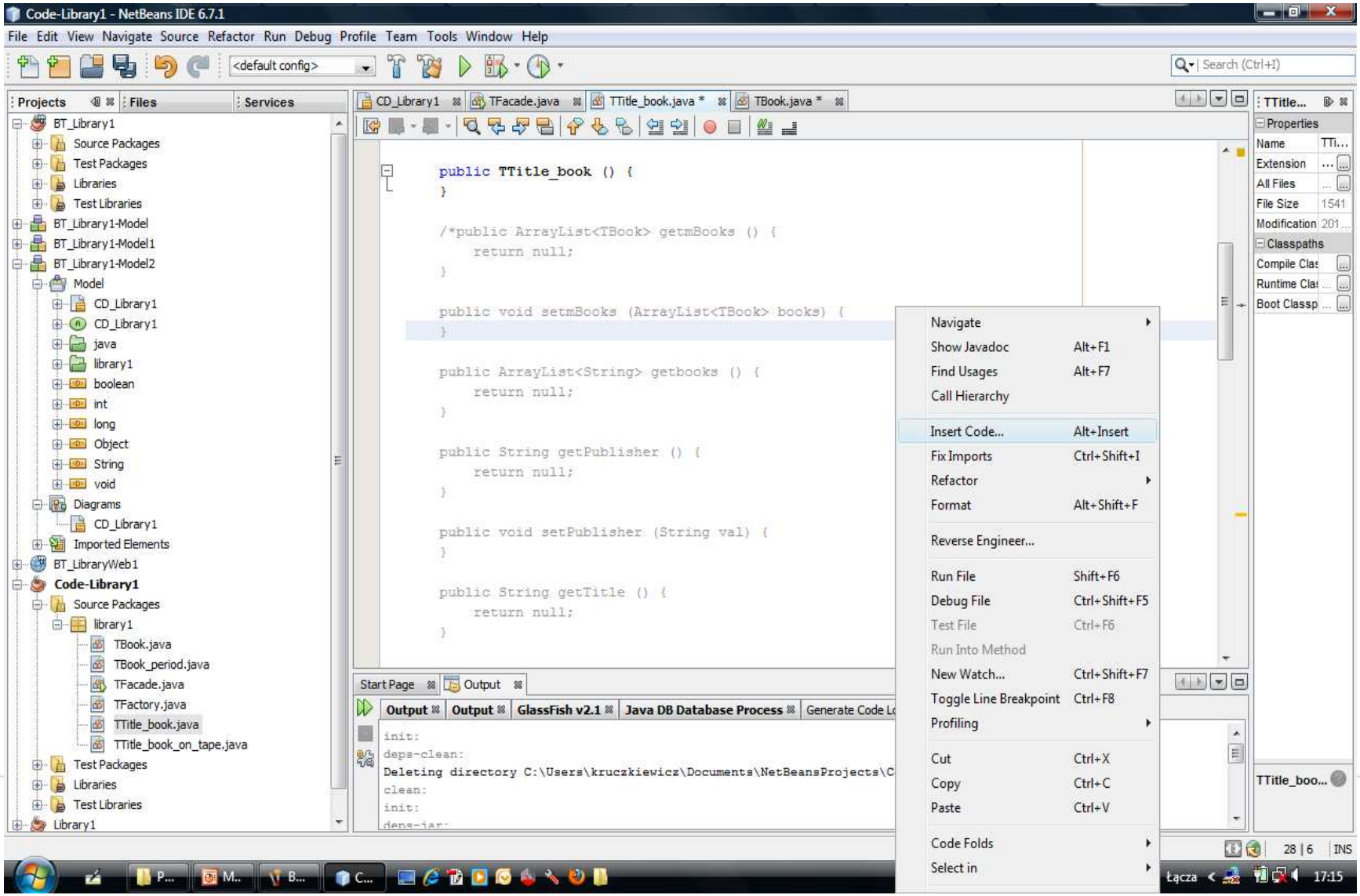
Result project as skeleton of program



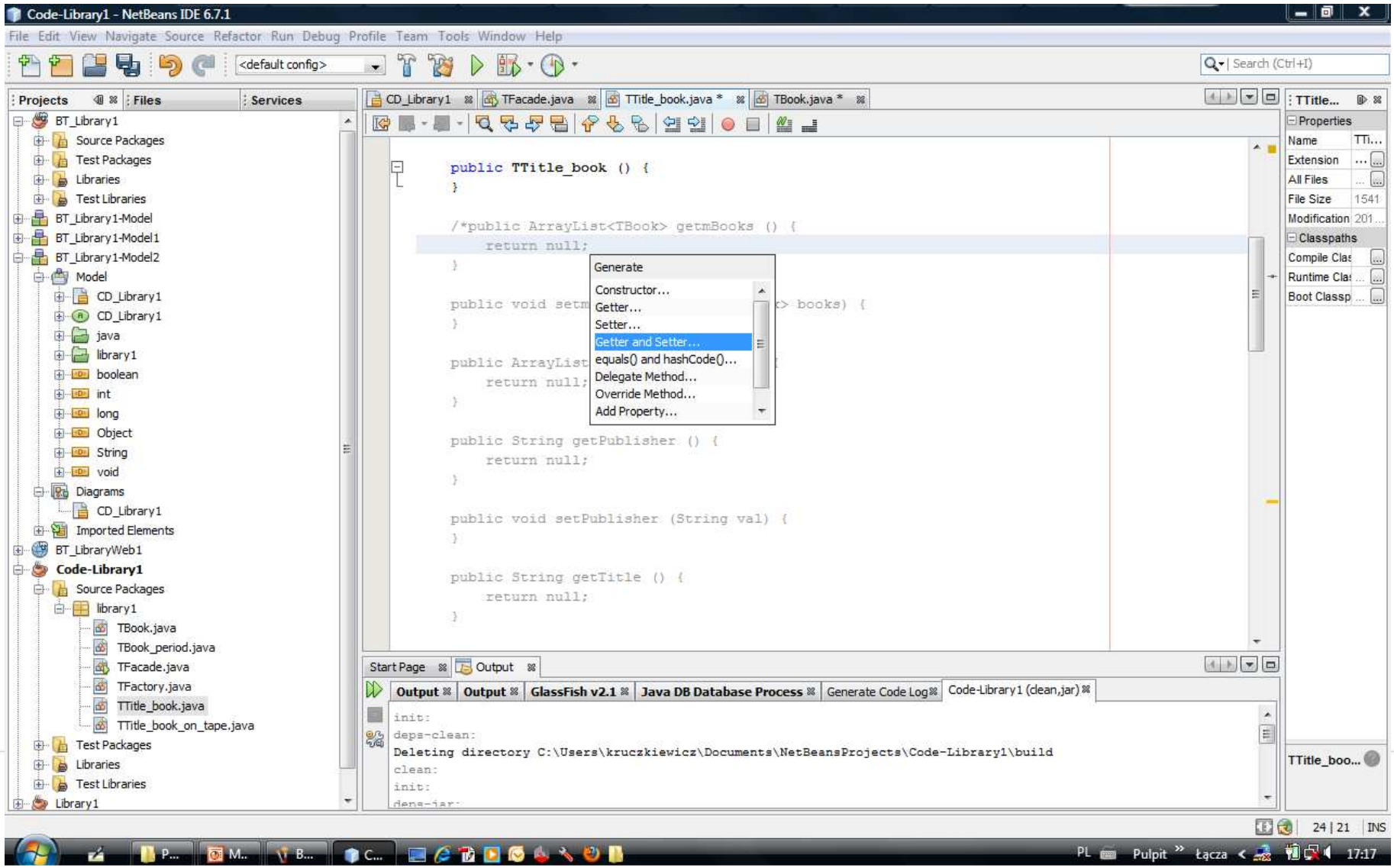
Code of TTitle_book

```
package library1;

import java.io.Serializable;
import java.util.ArrayList;

public class TTitle_book implements Serializable {
   private static final long serialVersionUID = 1L;
   private String publisher;
   private String ISBN;
   private String title;
   private String author;
   private ArrayList<TBook> mBooks = new java.util.ArrayList<TBook>();
   public TTitle_book ()                                          {  }
   public ArrayList<TBook> getmBooks ()              {     return null;   }
   public void setmBooks (ArrayList<TBook> books)    {  }
   public synchronized ArrayList<String> getbooks()               {  }
   public String toString ()                                      {     return null;   }
      public String getPublisher ()                               {     return null;   }
      public void setPublisher (String val)                       {  }
      public String getTitle ()                                   {     return null;   }
      public void setTitle (String val)                                        {  }
      public String getISBN ()                                    {     return null;   }
      public void setISBN (String val)                            {  }
      public String getAuthor ()                                  {     return null;   }
      public void setAuthor (String val)                          {  }
      public String getActor ()                                   {     return null;   }
      public void setActor (String val)                           {  }
      public int hashCode ()                                      {     return 0;   }
      public boolean equals (Object obj)                          {     return true;   }
   public void add_book (String[] data)              {  }
   public TBook search_book (TBook book)             {     return null;   }
   public TBook search_accessible_book(Object data)  {  }
}
```

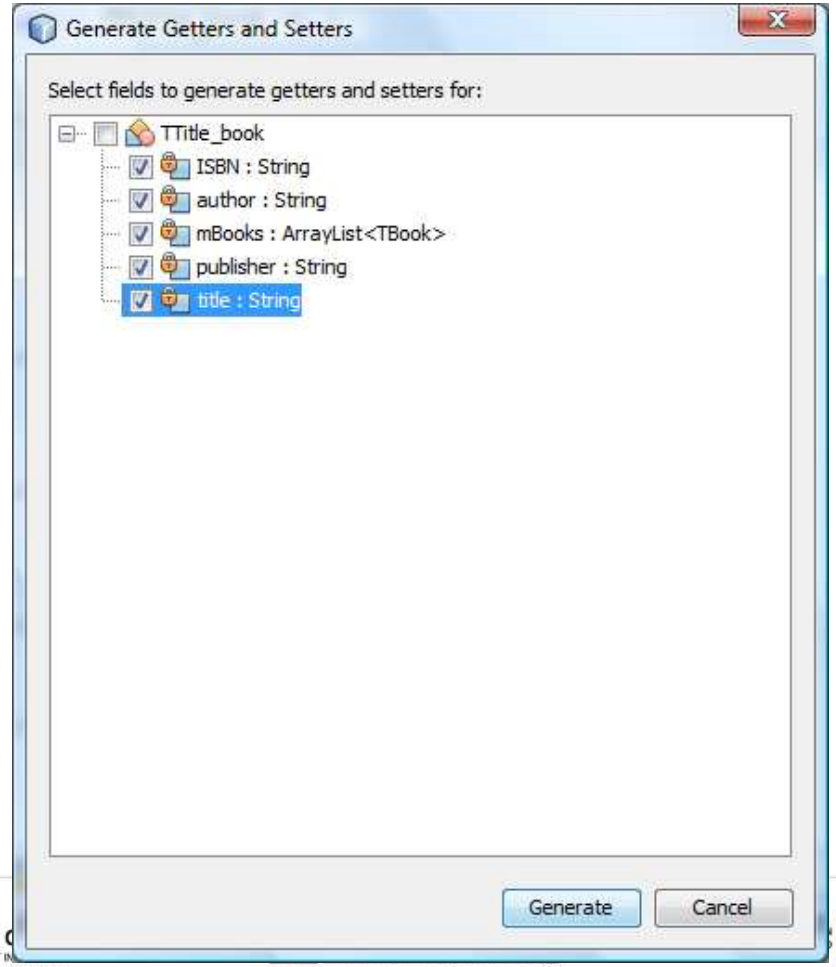
Insert code option from pop-up menu (generating Setter and Getter methods replaced with created from Insert code option) – (1)



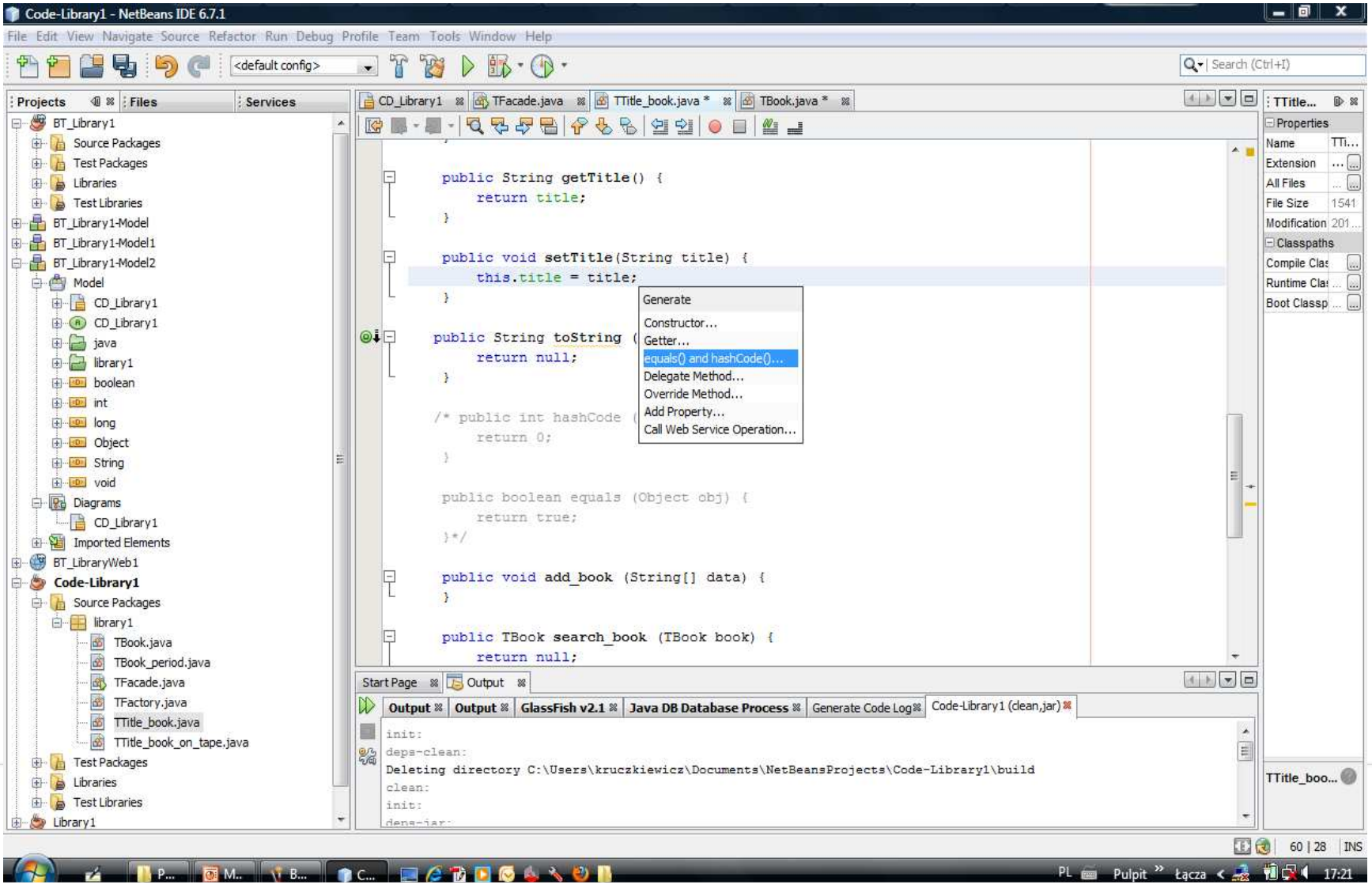
Insert code option from pop-up menu (generating Setter and Getter methods replaced with created from Insert code option) – (2)



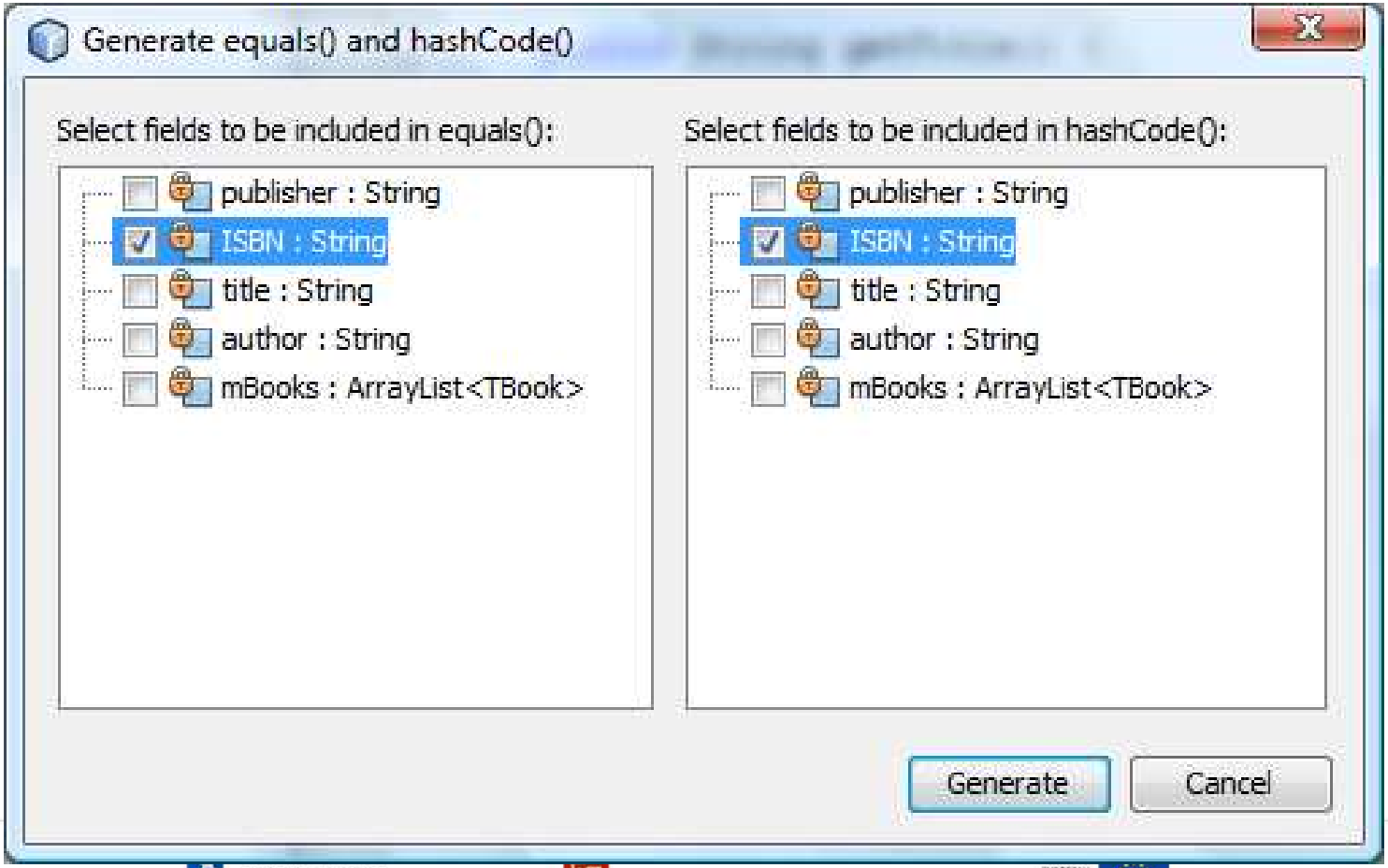
Insert code option from pop-up menu (generating Setter and Getter methods replaced with created from Insert code option) – (3)



Insert code option from pop-up menu (generating equals and hashCode methods replaced with created from Insert code option) –(1)



Insert code option from pop-up menu (generating equals and hashCode methods replaced with created from Insert code option) –(2)



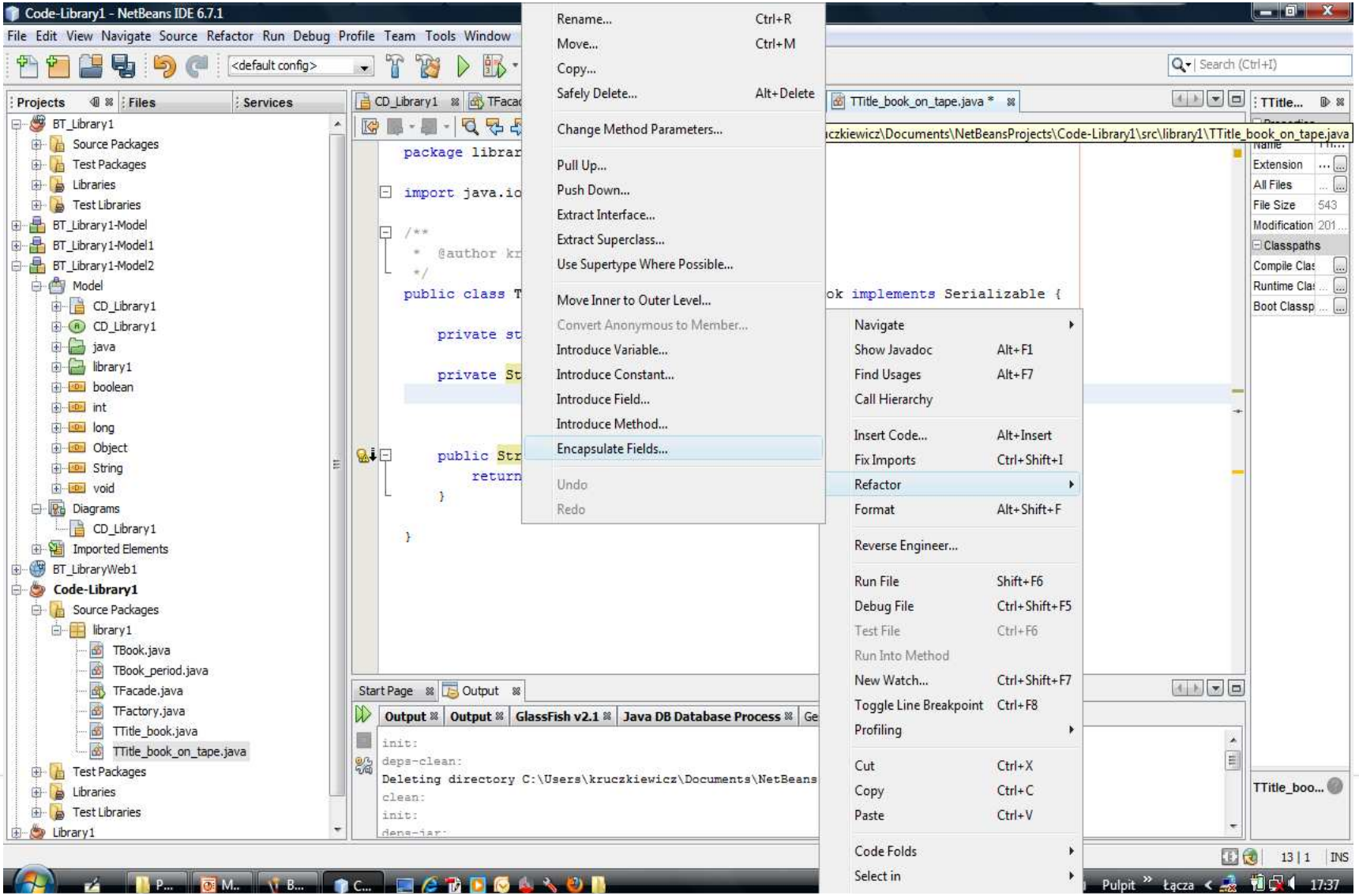
Code of TTitle_book_on_tape

```
package library1;

import java.io.Serializable;

public class TTitle_book_on_tape extends TTitle_book implements Serializable {

   private static final long serialVersionUID = 1L;
   private String actor;

   public String getActor ()                    {     return null;   }
   public void setActor (String val)            {   }
   public String toString ()                    {     return null;   }
}
```

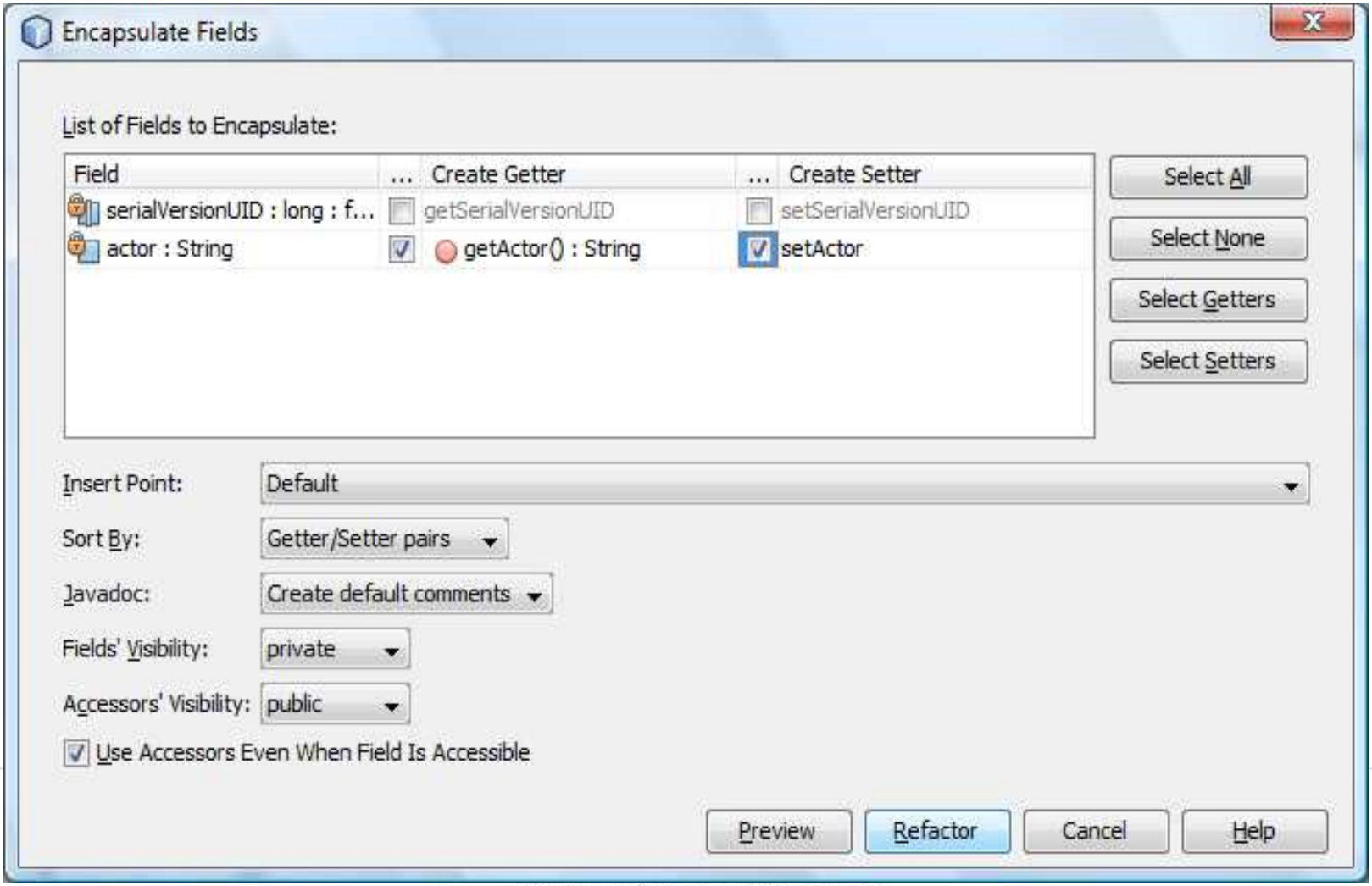
Code of TBook

```
package library1;

import java.io.Serializable;
import java.util.Date;

public class TBook implements Serializable {

   private static final long serialVersionUID = 1L;

   private int number;
   private TTitle_book mTitle_book;

   public TBook ()                                              {   }

   public TTitle_book getmTitle_book ()                {     return null;   }
   public void setmTitle_book (TTitle_book title_book)  {   }

   public Date getPeriod ()                                     {     return null;   }
   public void setPeriod (Date period)                          {   }
   public int getNumber ()                                      {     return 0;   }
   public void setNumber (int val)                              {   }
   public int hashCode ()                                       {     return 0;   }
   public String toString ()                                    {     return null;   }

   public boolean equals (Object obj)                           {     return true;   }
   public boolean period_pass(Object data)                      {     return true;   }
   public void startPeriod(Object data)                {   }
}
```

Refactor>Encapsulation Method> option from pop-up menu (generating Setter and Getter methods replaced with created from Refactor>Encapsulation Method> options) –(1)



Refactor>Encapsulation Method> option from pop-up menu (generating Setter and Getter methods replaced with created from Refactor>Encapsulation Method> options) –(2)



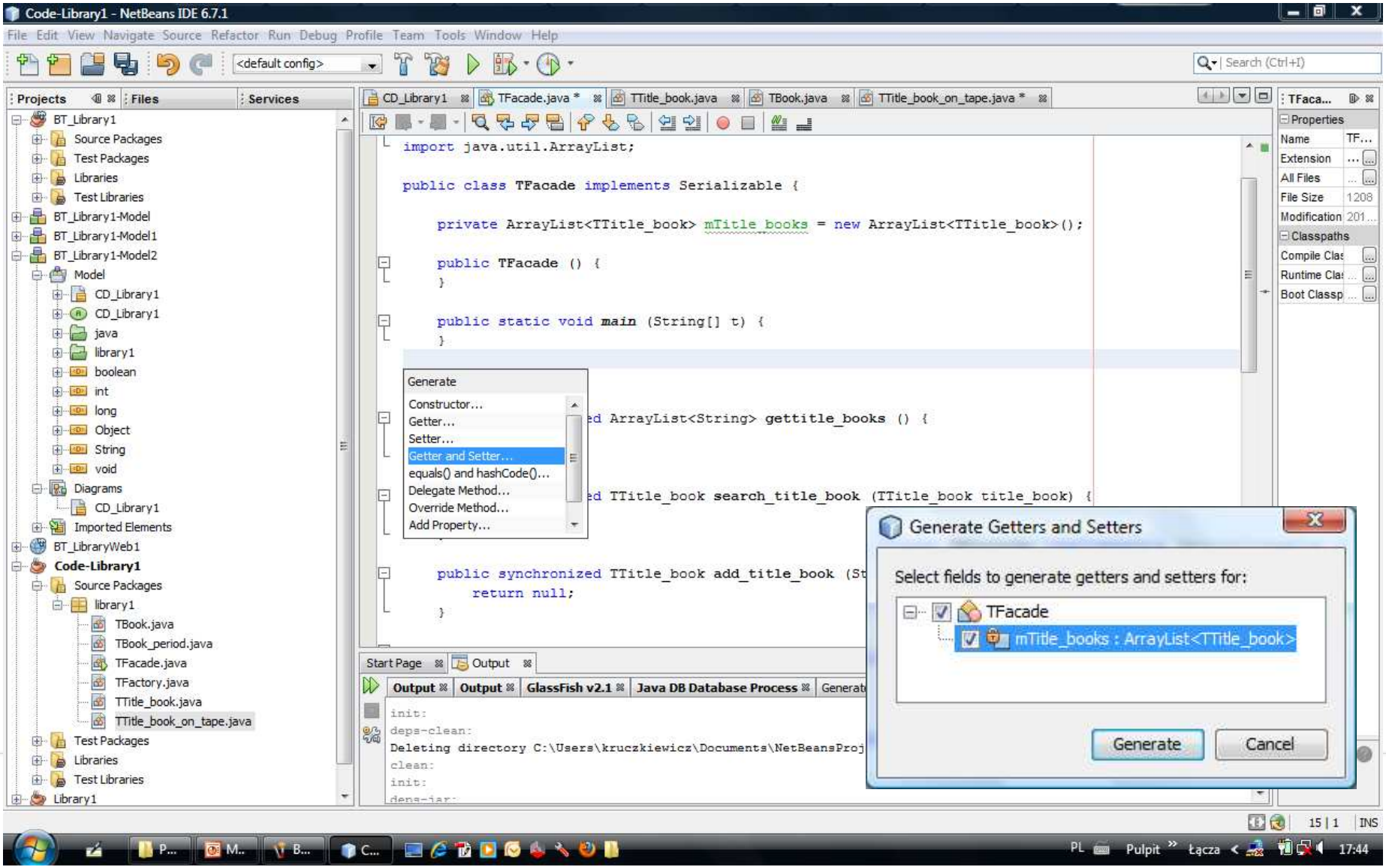
Code of TBook_period

```
package library1;

import java.io.Serializable;
import java.util.Date;

public class TBook_period extends TBook implements Serializable {

   private static final long serialVersionUID = 1L;
   private Date period;

   public Date getPeriod ()                          {     return null;   }
   public void setPeriod (Date date)                 {  }
   public String toString ()                         {     return null;   }
   public boolean period_pass(Object data)           {     return true;   }
   public void startPeriod(Object data)       {  }
}
```

Code of TFacade

```
package library1;

import java.io.Serializable;
import java.util.ArrayList;

public class TFacade implements Serializable {

   private ArrayList<TTitle_book> mTitle_books = new ArrayList<TTitle_book>();

   public TFacade ()                                                                    {   }
   public synchronized ArrayList<TTitle_book> getmTitle_books ()                {     return null;   }
   public synchronized void setmTitle_books (ArrayList<TTitle_book> title_books)

   public synchronized TTitle_book search_title_book (TTitle_book title_book)   {     return null;   }
   public synchronized TTitle_book add_title_book (String[] data)               {     return null;   }
   public synchronized TTitle_book add_book (String[] data1, String[] data2)    {     return null;   }
   public synchronized TTitle_book Search_title_book (String[] data)            {     return null;   }
   public synchronized TBook Search_book (String[] data1, String[] data2)                {     return null;   }
   public synchronized TBook Search_accessible_book(String data1[], Object data2)  {  }

   public synchronized ArrayList<String> gettitle_books()                       {   }
   public synchronized void Print_books ()                                      {   }
   public synchronized void Print_title_books ()                                        {   }
   public static void main (String[] t)                                                 {   }
}
```

Generated code for replacing existing Setter and Getter methods



Code of Factory

```
package library1;

public class TFactory {

   static final long day=24*60*60*1000;
   static public Date mdays(String data)                                    {     }

   public TTitle_book create_title_book (String[] data)     {     return null;   }
   public TBook create_book (String[] data)                 {     return null;   }


}
```

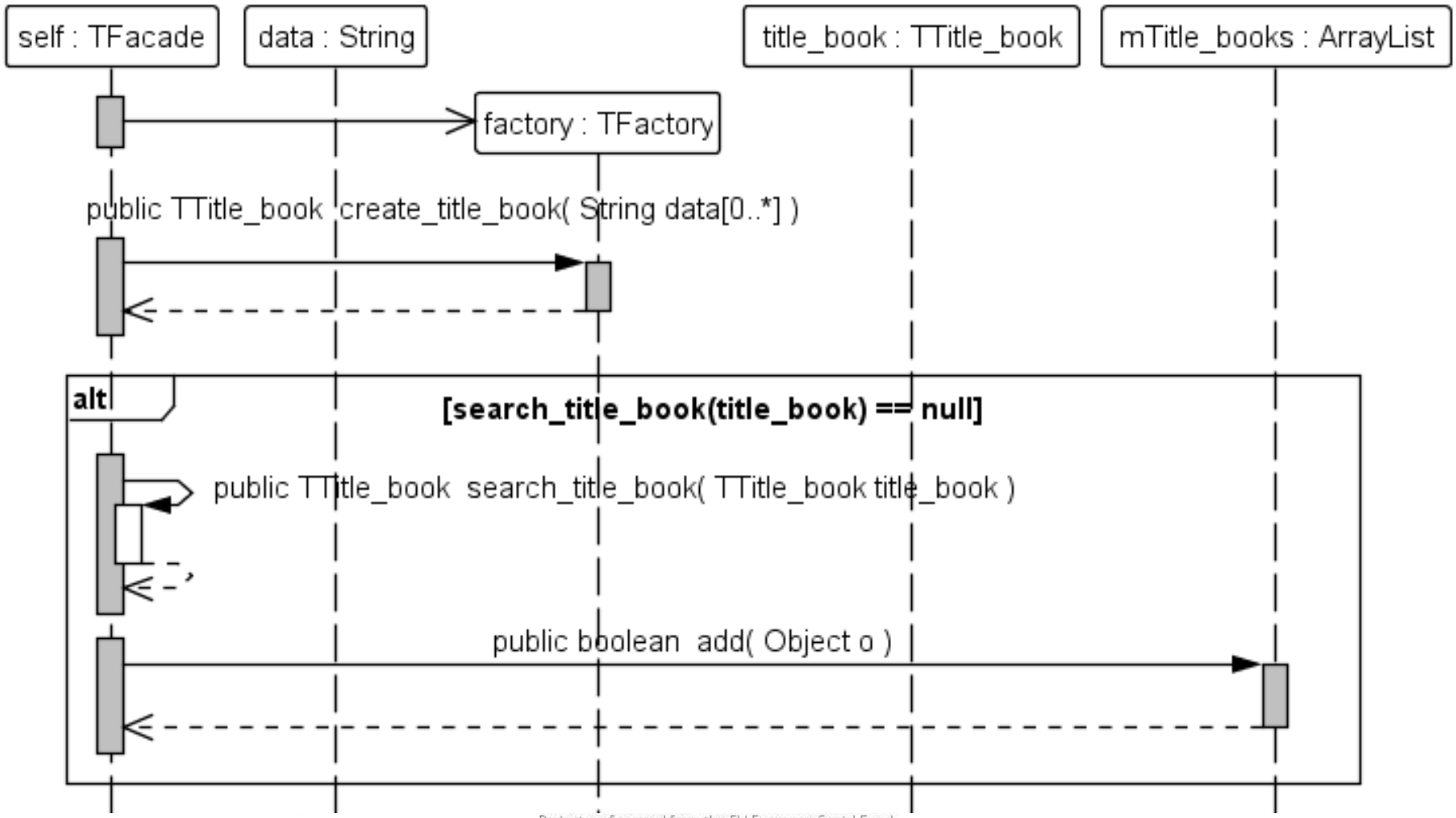
First Iteration

Create the code of methods based on the class diagram and sequence diagrams. Next you must build and run program with the content of the main method of TFacade class defined on the further slide, as the acceptance test.

Applying Facade and Factory design patterns

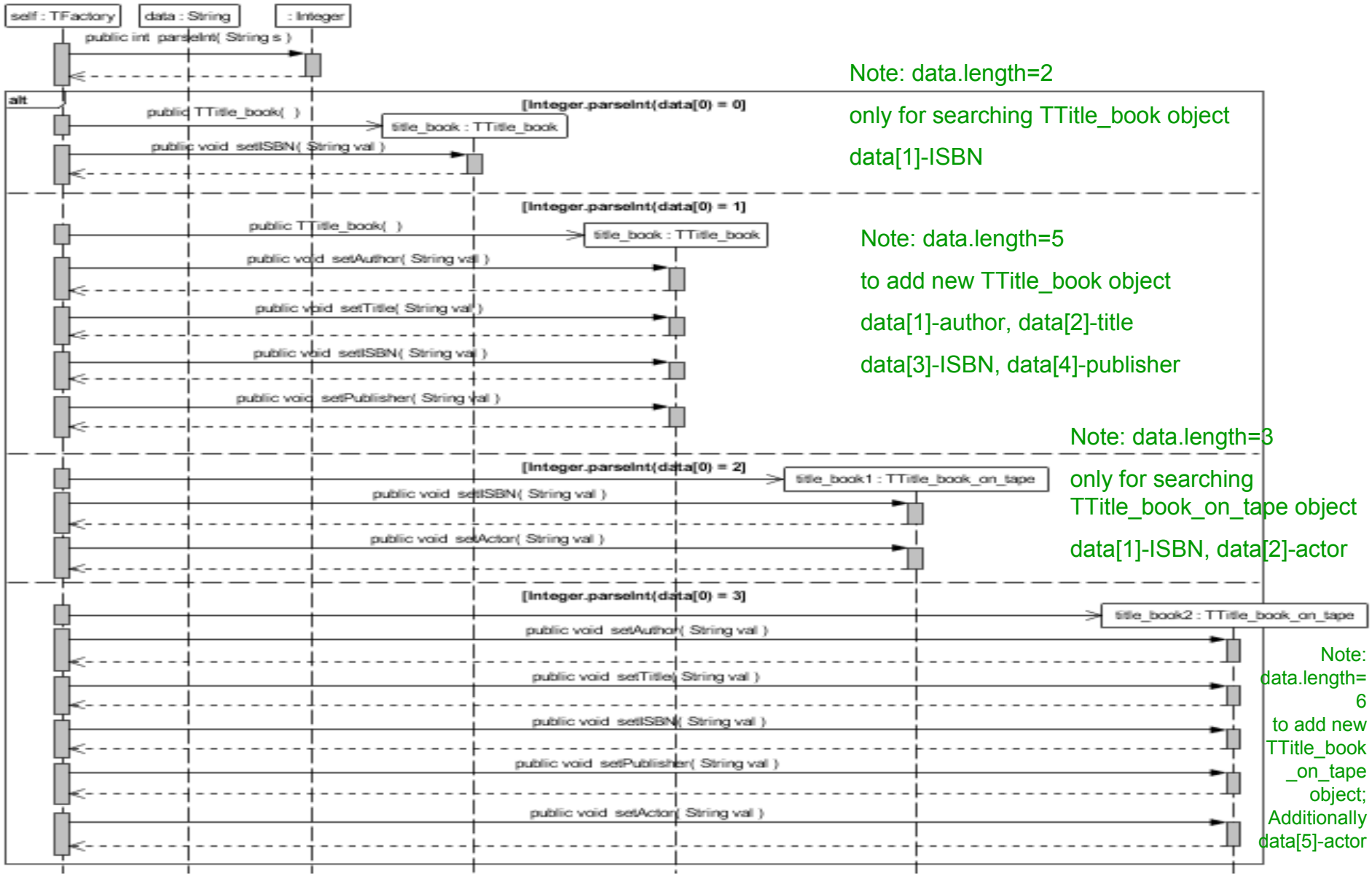
UC_TFacade_add_title_book

TFacade: public synchronized TTitle_book add_title_book(String data[])



Project co-financed from the EU European Social Fund

(1) -TFactory_create_title_book; T**Factory**: public TTitle_book create_title_book(String data[]) (see next 2 slides)



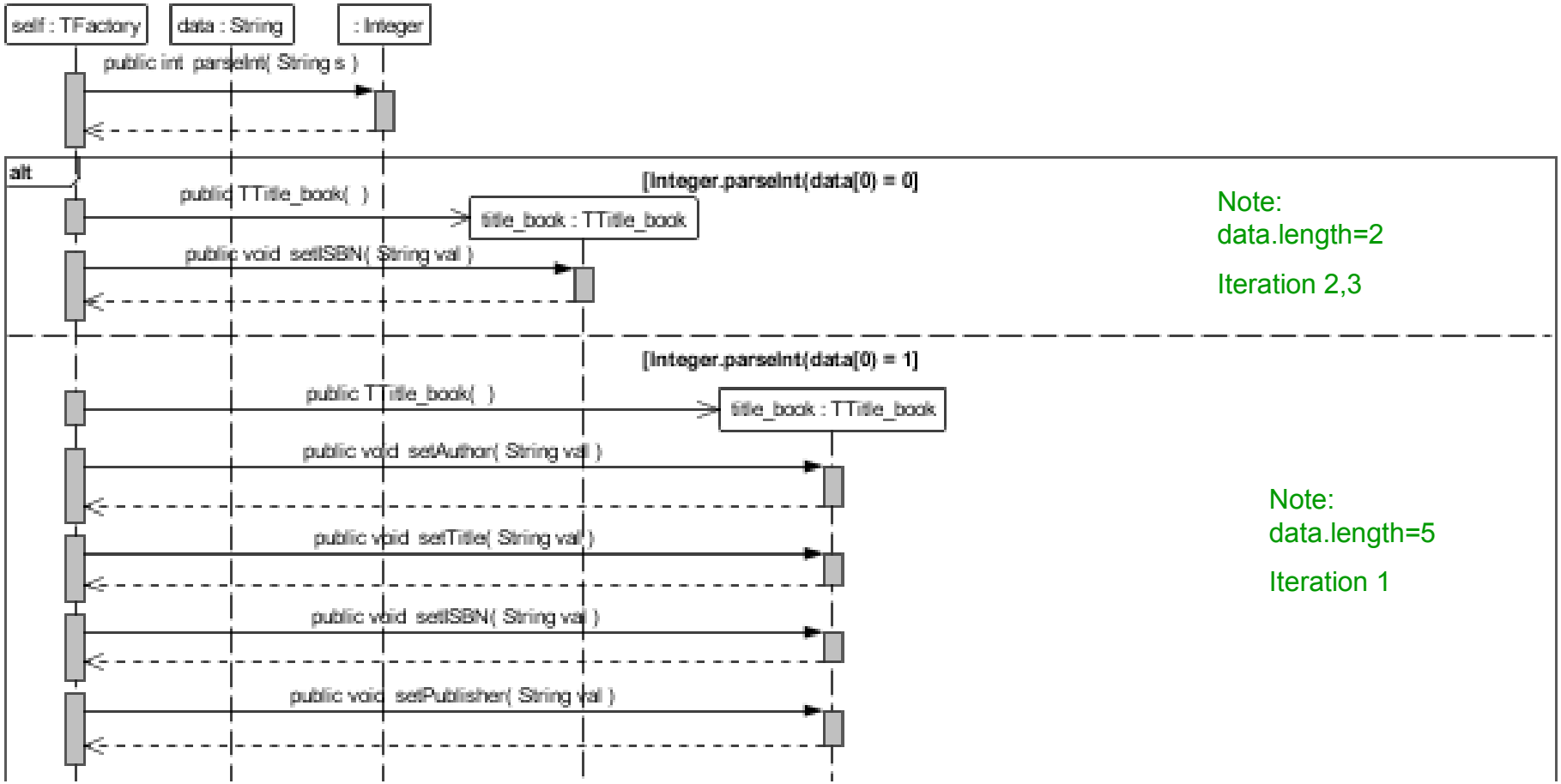
Note: data.length=2
only for searching TTitle_book object
data[1]-ISBN

Note: data.length=5
to add new TTitle_book object
data[1]-author, data[2]-title
data[3]-ISBN, data[4]-publisher

Note: data.length=3
only for searching
TTitle_book_on_tape object
data[1]-ISBN, data[2]-actor

Note: data.length= 6 to add new TTitle_book _on_tape object; Additionally data[5]-actor

(2) – The first fragment of the sequence diagram of the T**Factory**: public TTitle_book create_title_book(String data[]) method



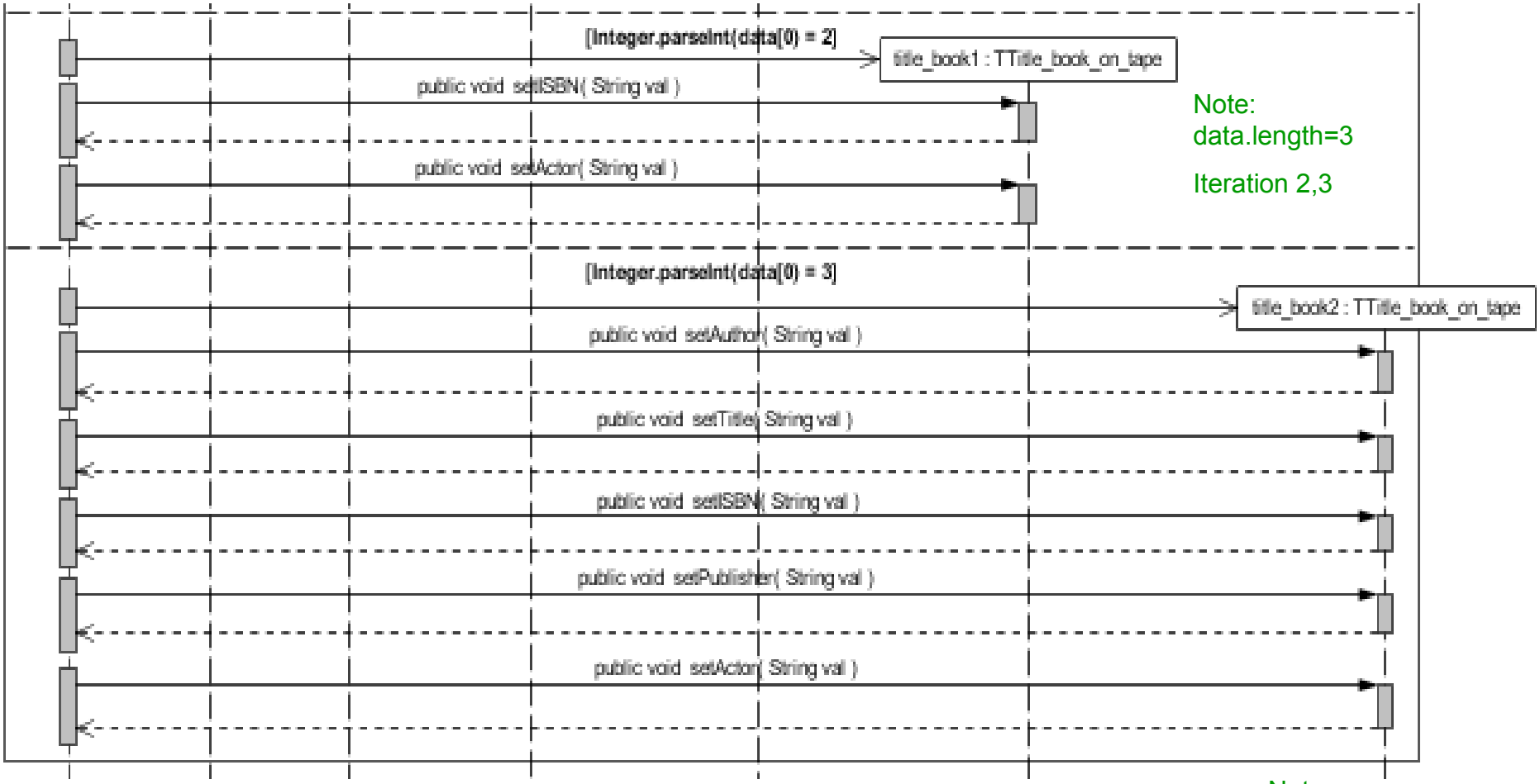
Note:
data.length=2

Iteration 2,3

Note:
data.length=5

Iteration 1

(3) – The second fragment of the sequence diagram of the T**Factory**: public TTitle_book create_title_book(String data[]) method



Note:
data.length=3

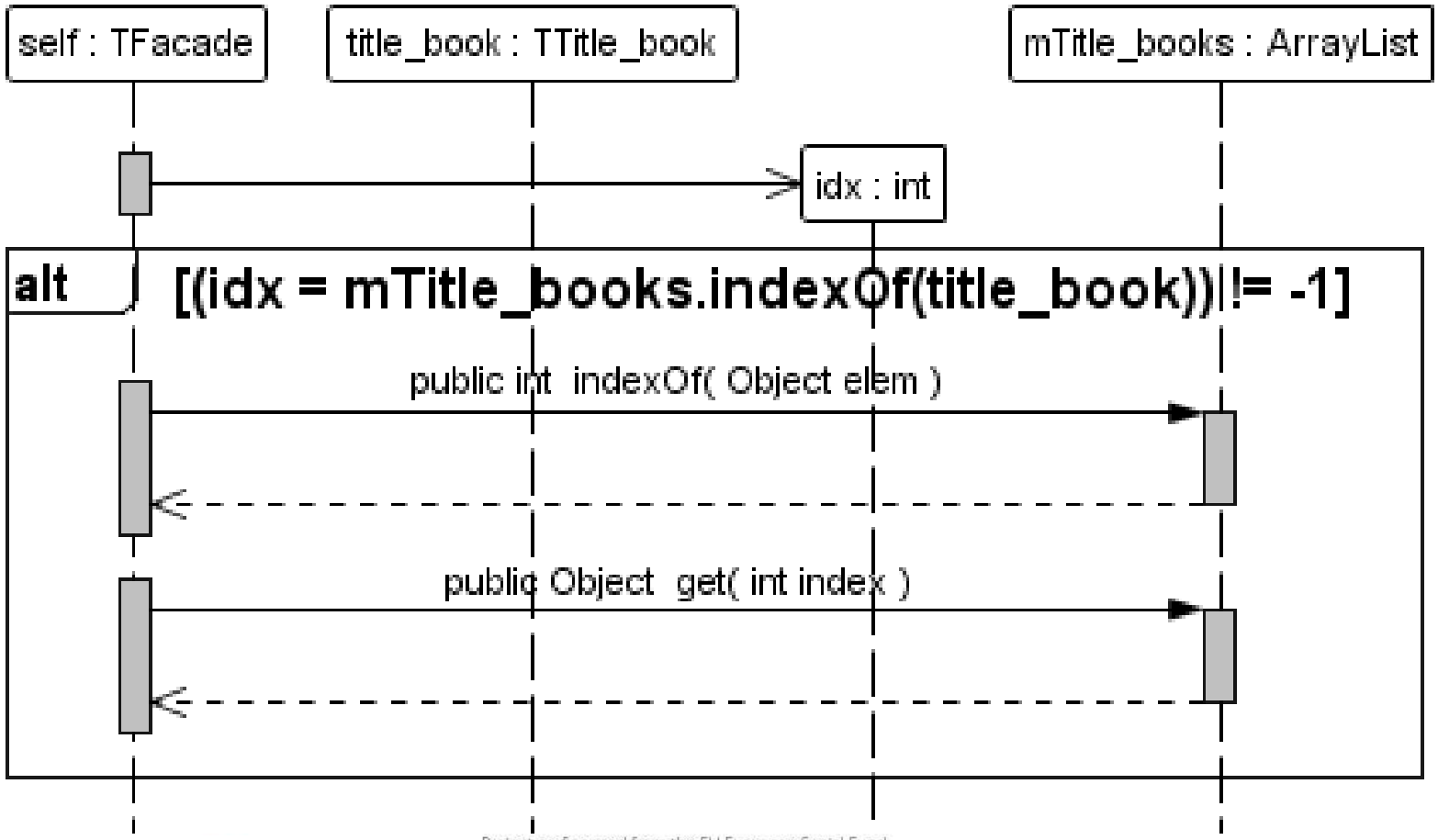
Iteration 2,3

Note:
data.length=6

Iteration 1

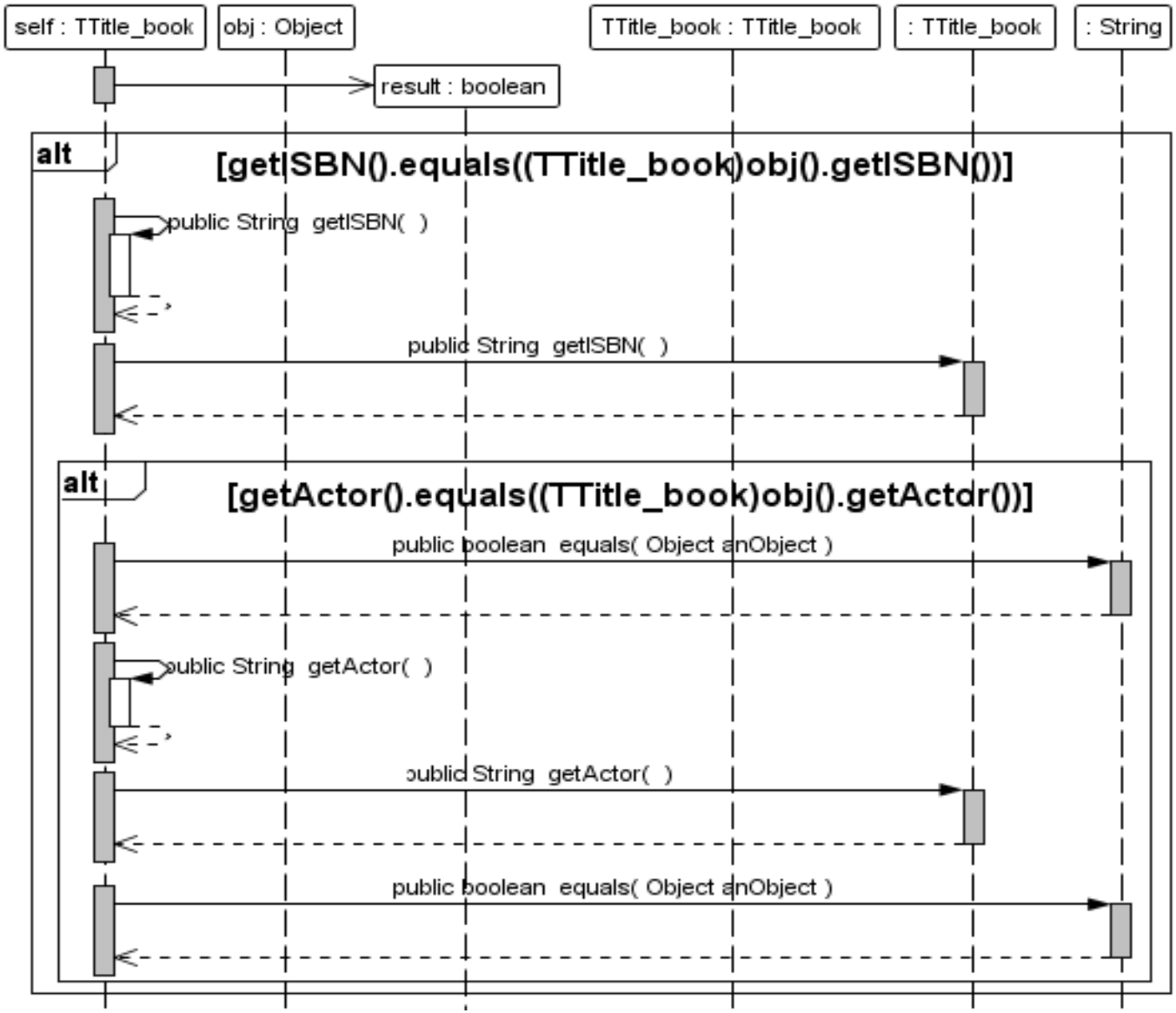
UC_TFacade_search_title_book

TFacade: public synchronized TTitle_book search_title_book(TTitle_book title_book)



Project co-financed from the EU European Social Fund

TTitle_book_equals - TTitle_book: public boolean equals(Object obj)



The first iteration

```
package library1;

import java.io.Serializable;
import java.util.ArrayList;

public class TFacade implements Serializable {
/* Code of TFacade methods*/

public static void main(String t[])    {
      TFacade ap = new TFacade();
      String t1[] = {"1", "Author1","Title1", "ISBN1", "Publisher1"};
      String t2[] = {"1", "Author2","Title2", "ISBN2", "Publisher2"};
      String t3[] = {"1", "Author3","Title3", "ISBN3", "Publisher3"};
      String t4[] = {"3", "Author1","Title1", "ISBN1", "Publisher1", "Actor1"};
      String t5[] = {"3", "Author2","Title2", "ISBN2", "Publisher2", "Actor2"};
      String t6[] = {"3", "Author4","Title4", "ISBN4", "Publisher4", "Actor4"};
     ap.add_title_book(t1);
      ap.add_title_book(t2);
      ap.add_title_book(t2);
      ap.add_title_book(t3);
      ap.add_title_book(t4);
      ap.add_title_book(t5);
      ap.add_title_book(t5);
      ap.add_title_book(t6);
      String lan = ap.getmTitle_books().toString();
      System.out.println(lan);
  }
}
```

Note: data.length=5; to add new TTitle_book object

Note: data.length=6; to add new TTitle_book_on_tape object

```
[
Title: 1 Author: 1 ISBN: 1 Publisher: 1,
Title: 2 Author: 2 ISBN: 2 Publisher: 2,
Title: 3 Author: 3 ISBN: 3 Publisher: 3,
Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1,
Title: 2 Author: 2 ISBN: 2 Publisher: 2 Actor: 2,
Title: 4 Author: 4 ISBN: 4 Publisher: 4 Actor: 4]
```

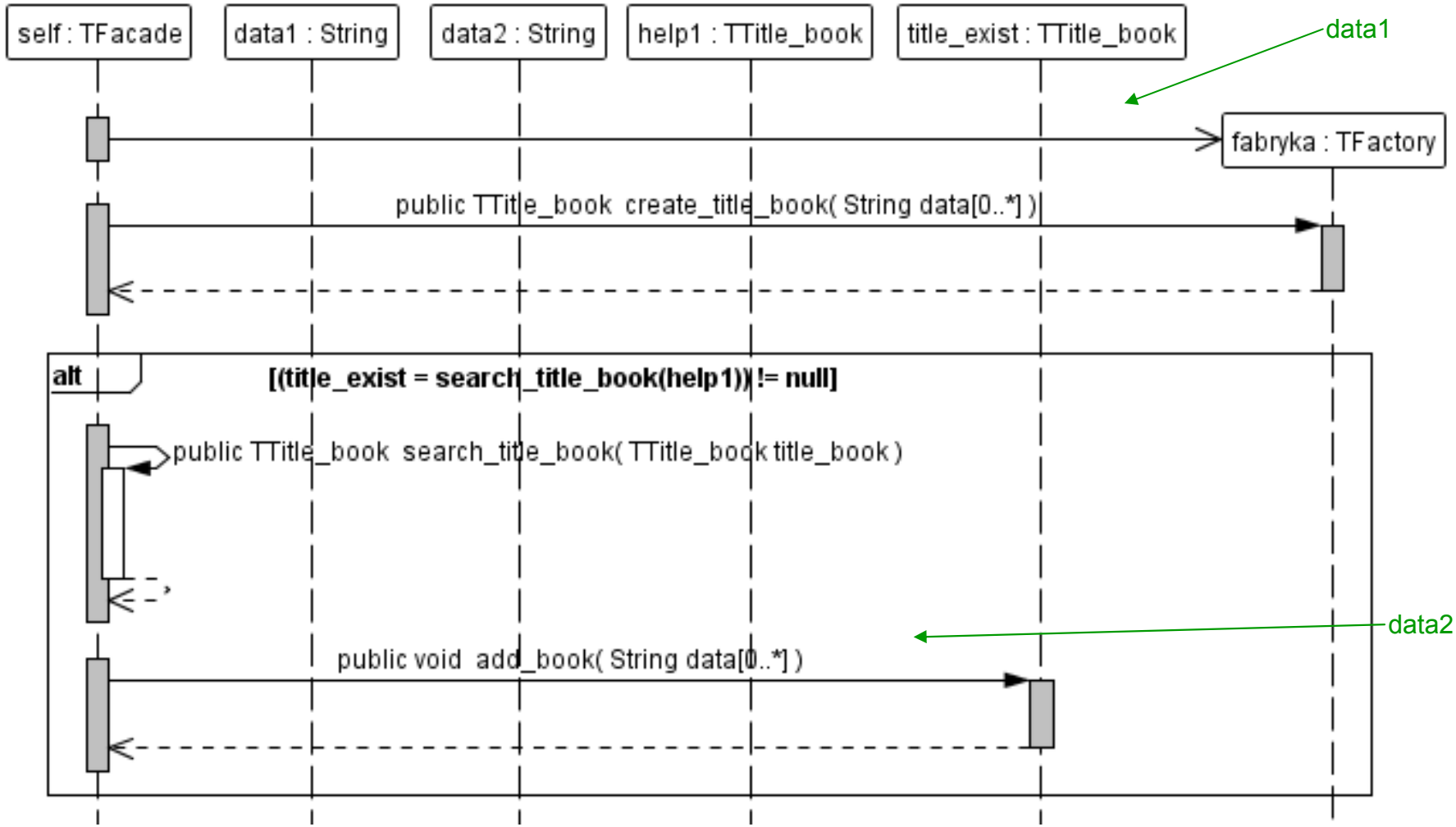
Second Iteration

Create the code of methods based on the class diagram and sequence diagrams. Next you must build and run program with the content of the main method of TFacade class defined on the further slide, as the acceptance test.

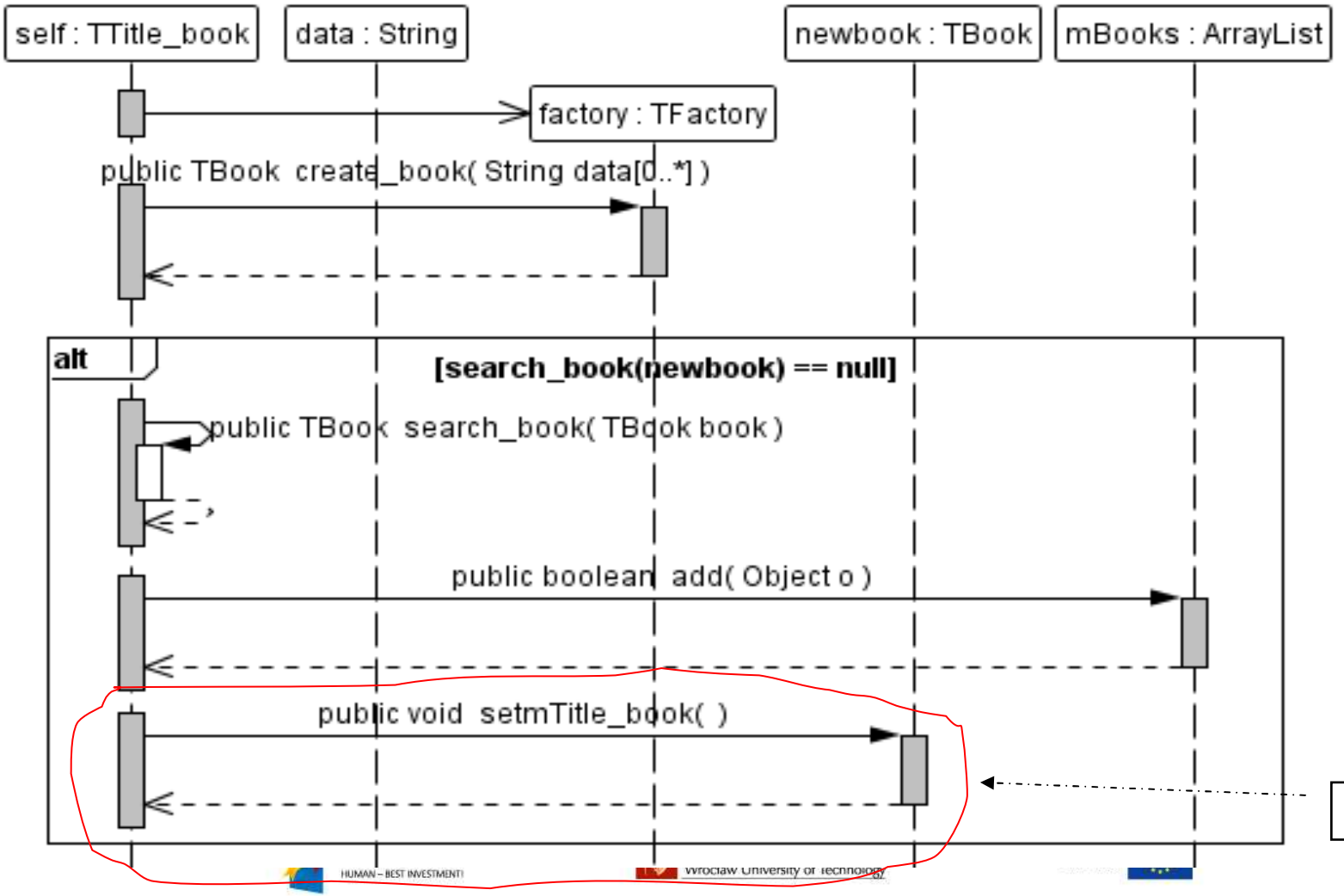
Applying Facade, Factory and Flyweight design patterns

UC_TFacade_add_book

TFacade: public synchronized TTitle_book add_book(String data1[], String data2[])

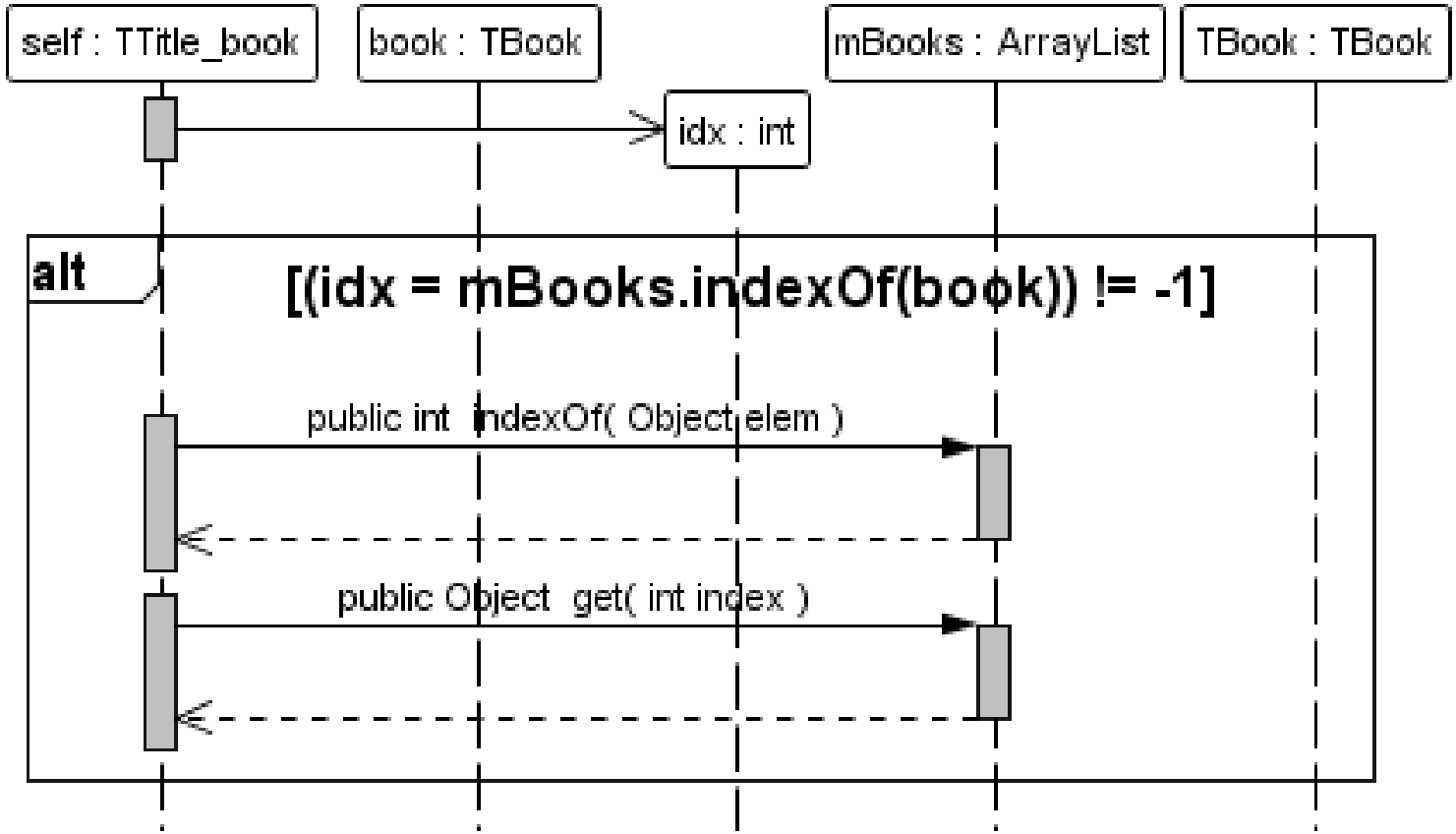


TTitle_book_add_book
Title_book: public void add_book(String data[])



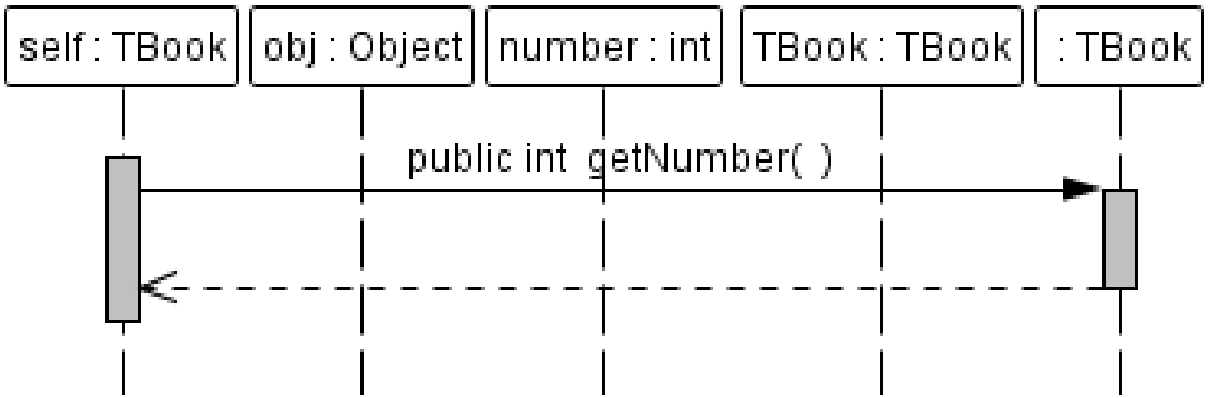
UC_TTitle_book_search_book

TTitle_book: public TBook search_book(TBook book)

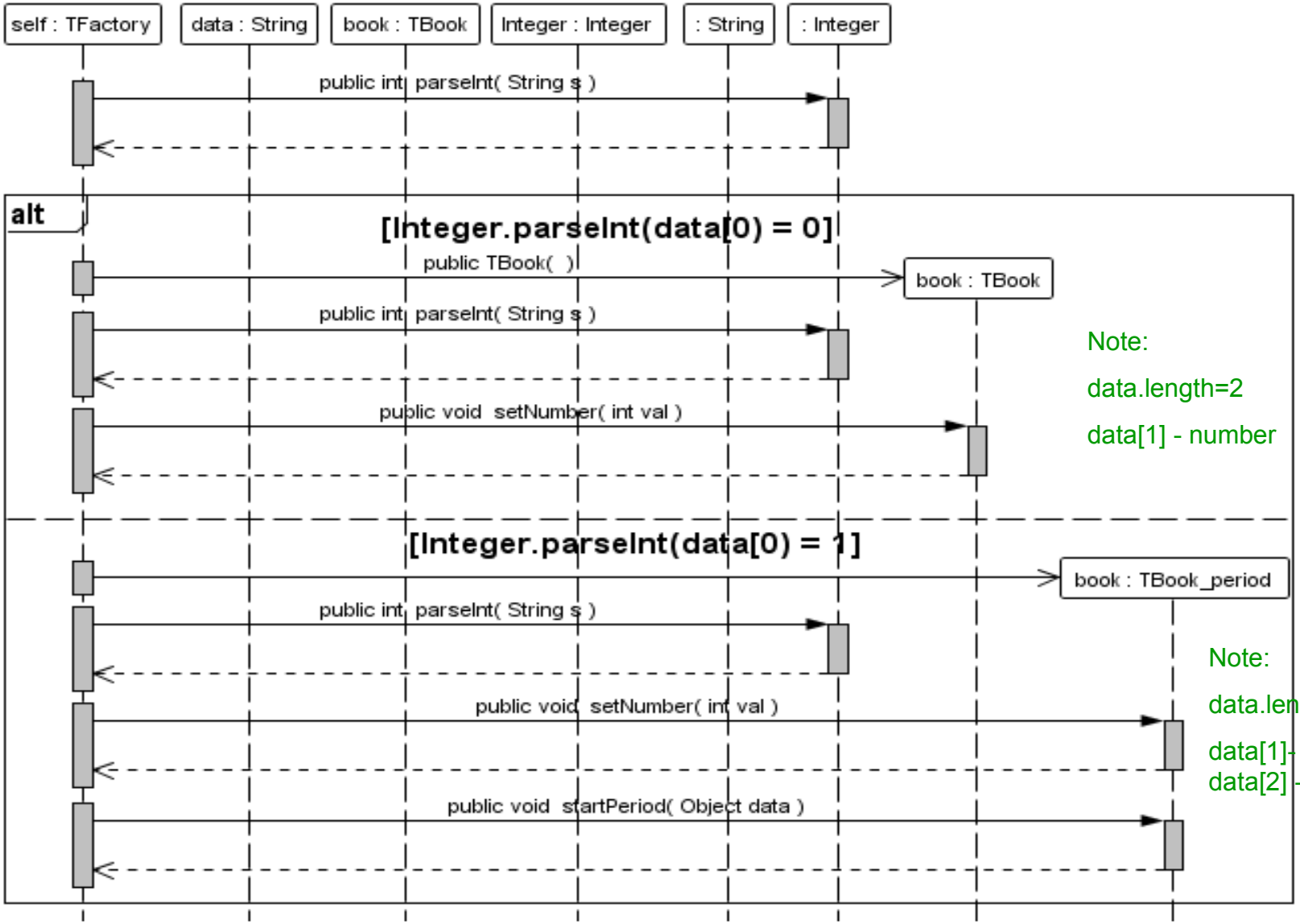


TBook_equals

TBook: public boolean equals(Object obj)

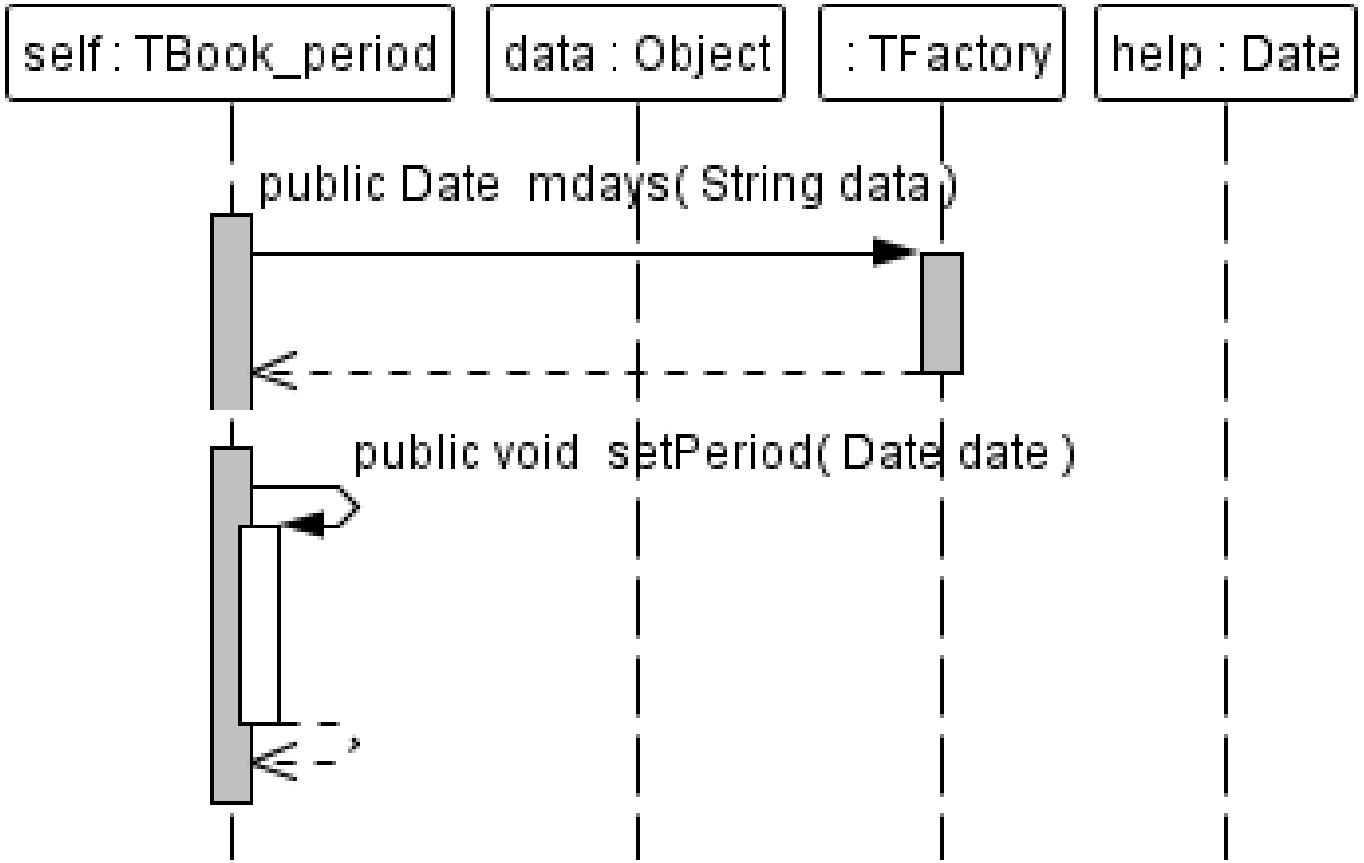


TFactory_create_book TFactory: public TBook create_book(String data[])



TBook_period_startPeriod

TBook_period: public void startPeriod(Object data)



The second iteration (1)

```
package library1;

import java.io.Serializable;
import java.util.ArrayList;

public class TFacade implements Serializable {
/* Code of TFacade methods*/

public static void main(String t[])    {
      TFacade ap = new TFacade();
       String t1[] = {"1", "Author1","Title1", "ISBN1", "Publisher1"};
       String t2[] = {"1", "Author2","Title2", "ISBN2", "Publisher2"};
       String t3[] = {"1", "Author3","Title3", "ISBN3", "Publisher3"};
       String t4[] = {"3", "Author1","Title1", "ISBN1", "Publisher1", "Actor1"};
       String t5[] = {"3", "Author2","Title2", "ISBN2", "Publisher2", "Actor2"};
       String t6[] = {"3", "Author4","Title4", "ISBN4", "Publisher4", "Actor4"};
       ap.add_title_book(t1);
       ap.add_title_book(t2);
       ap.add_title_book(t2);
       ap.add_title_book(t3);
       ap.add_title_book(t4);
       ap.add_title_book(t5);
       ap.add_title_book(t5);
       ap.add_title_book(t6);
       String lan = ap.getmTitle_books().toString();
       System.out.println(lan);
```

Note: data.length=5; to add new TTitle_book object

Note: data.length=6; to add new TTitle_book_on_tape object

The second iteration (2)

```
String d1[] = {"0", "ISBN1"};
String d2[] = {"0", "ISBN2"};
String d3[] = {"0", "ISBN5"};
String d4[] = {"2", "ISBN1", "Actor1"};
String d5[] = {"2", "ISBN4", "Actor4"};
String tr1[] = {"0", "1"};
String tr2[] = {"0", "2"};
String tr3[] = {"1", "3", "3"};
String tr4[] = {"1", "2", "-1"};
 TTitle_book pom = ap.add_book(d1, tr1);
if (pom != null) {        System.out.print(pom.getmBooks().toString());     }
pom = ap.add_book(d2, tr1);
if (pom != null) {        System.out.print(pom.getmBooks().toString());     }
pom = ap.add_book(d2, tr1);
if (pom != null) {        System.out.print(pom.getmBooks().toString());     }
pom = ap.add_book(d2, tr2);
if (pom != null) {        System.out.print(pom.getmBooks().toString());     }
pom = ap.add_book(d3, tr2);
if (pom != null) {        System.out.print(pom.getmBooks().toString());     }
pom = ap.add_book(d4, tr3);
if (pom != null) {        System.out.print(pom.getmBooks().toString());     }
pom = ap.add_book(d4, tr3);
if (pom != null) {        System.out.print(pom.getmBooks().toString());     }
pom = ap.add_book(d4, tr4);
if (pom != null) {        System.out.print(pom.getmBooks().toString());     }
pom = ap.add_book(d5, tr2);
if (pom != null) {        System.out.print(pom.getmBooks().toString());     }
System.out.println();
  }
}
```

Note: data.length=2; only for searching TTitle_book object

Note: data.length=3; only for searching TTitle_book_on_tape object

Note: data.length=2; to add new TBook object; tr1[1], tr2[1] - number

Note: data.length=3; to add new TBook_period object

tr3[1], tr4[1] - number, tr3[2], tr4[2] - period

The second iteration (3) - Output window

[
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: 1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 2][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:51:45 CET 2011][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:51:45 CET 2011][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:51:45 CET 2011,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Sat Feb 12 18:51:45 CET 2011][
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2]

Third Iteration

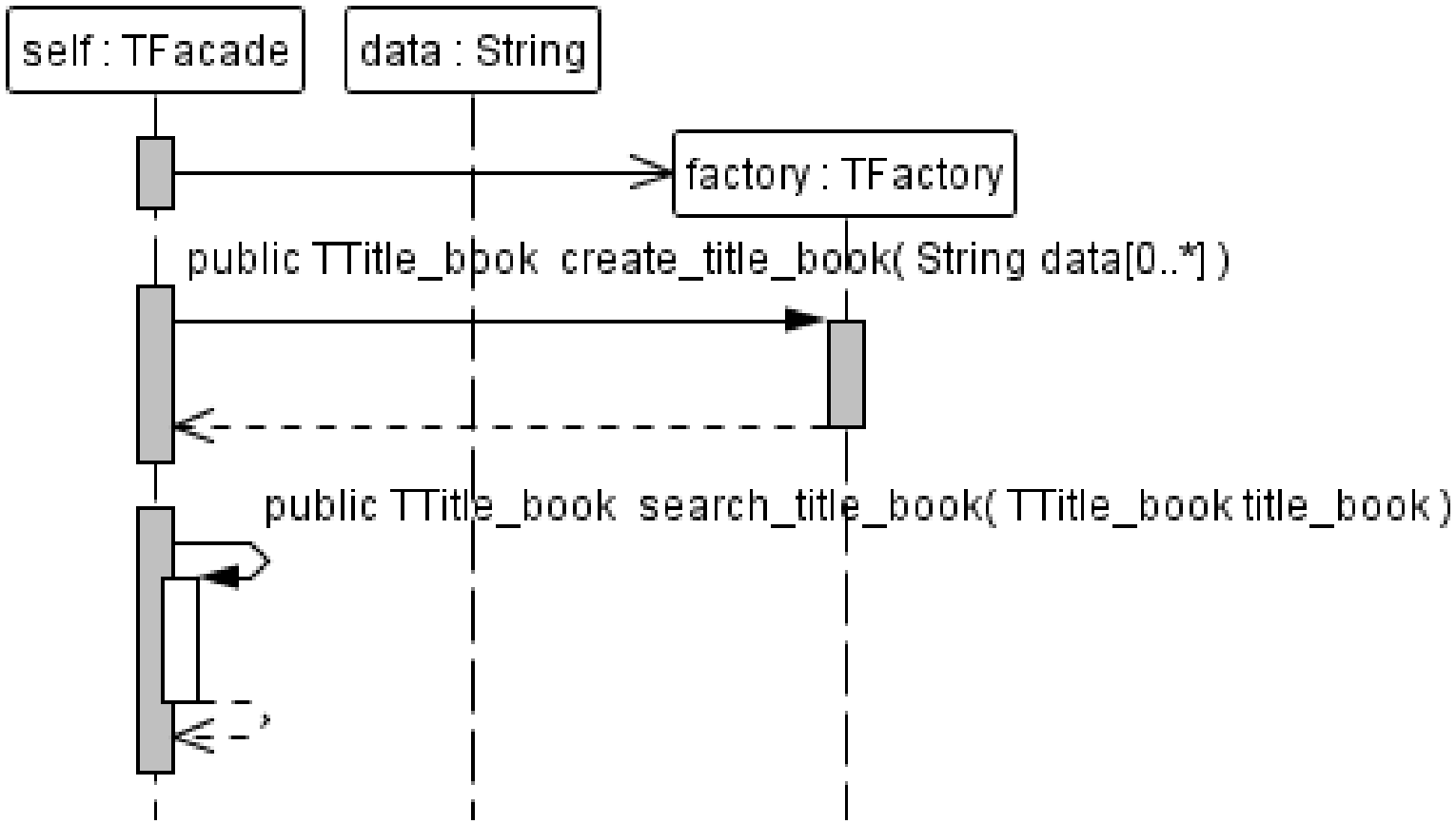
Create the code of methods based on the class diagram and sequence diagrams.

Next you must build and run program with the content of the main method of TFacade class defined on the further slide, as the acceptance test.

Applying of Facade, Factory, Flyweight and Strategy design patterns

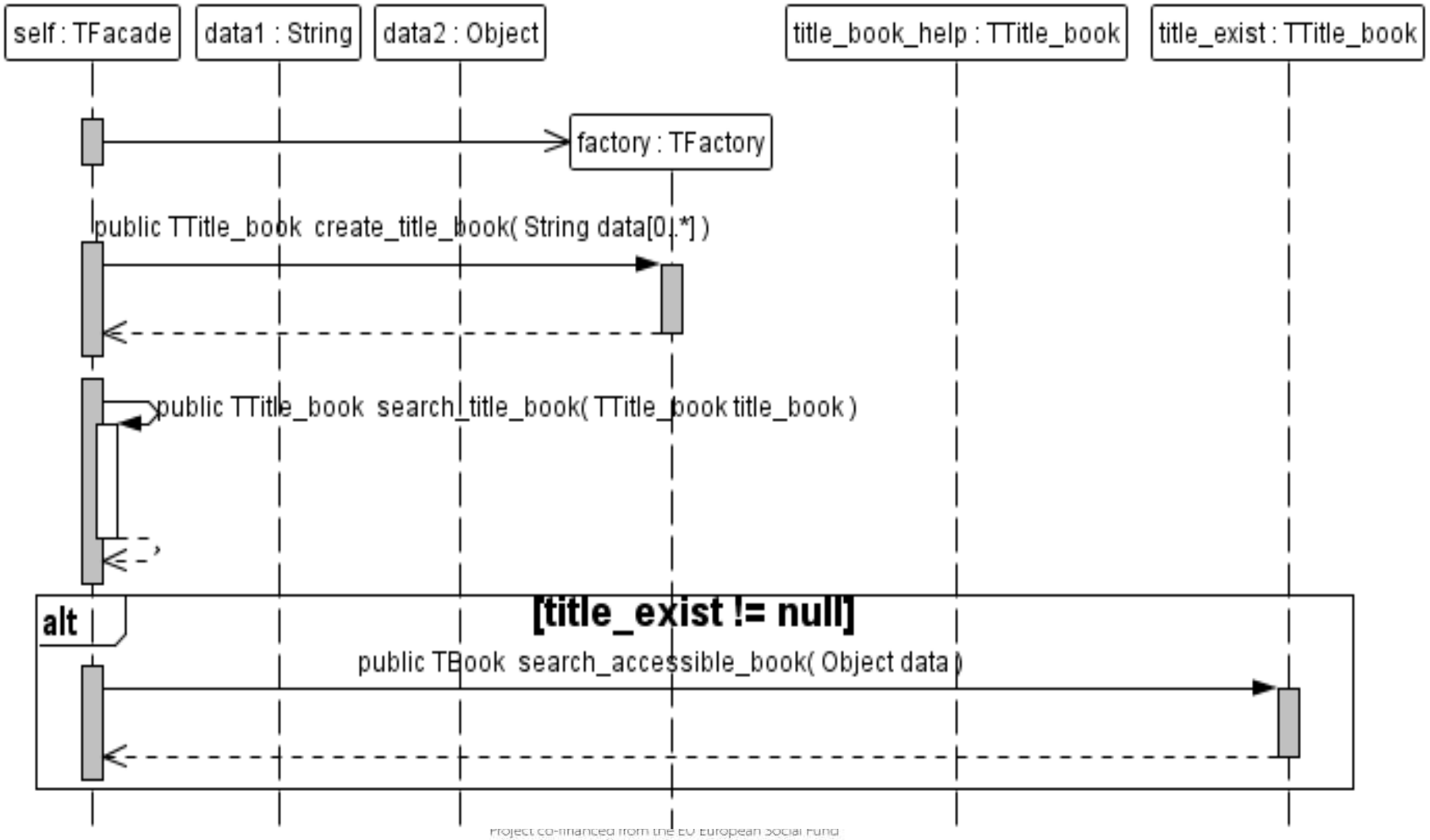
UC_TFacade_Search_title_book

TFacade: public synchronized TTitle_book Search_title_book(String data[])



UC_TFacade_Search_accessible_book

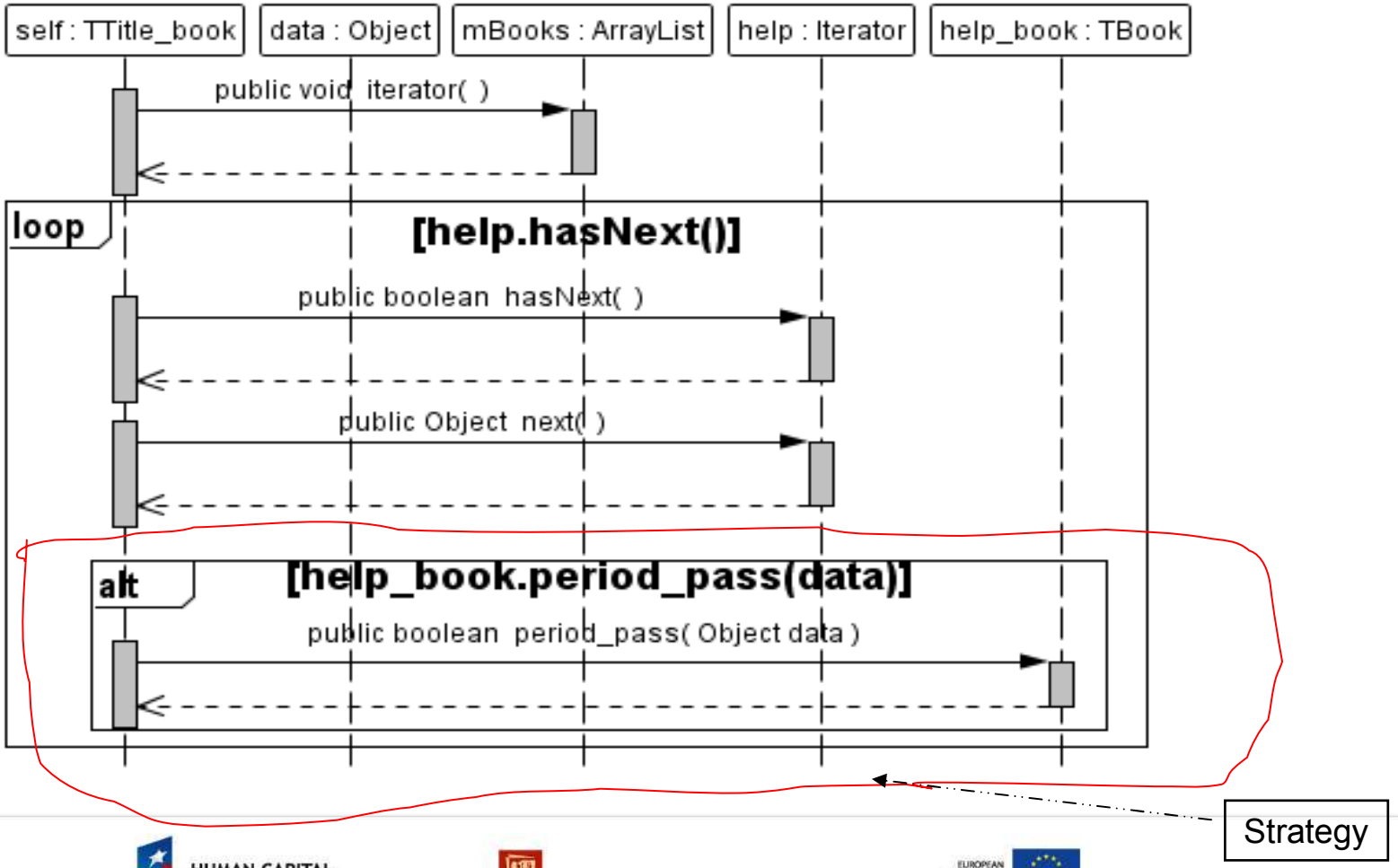
TFacade: public synchronized TBook Search_accessible_book(String data1[])



Project co-financed from the EU European Social Fund

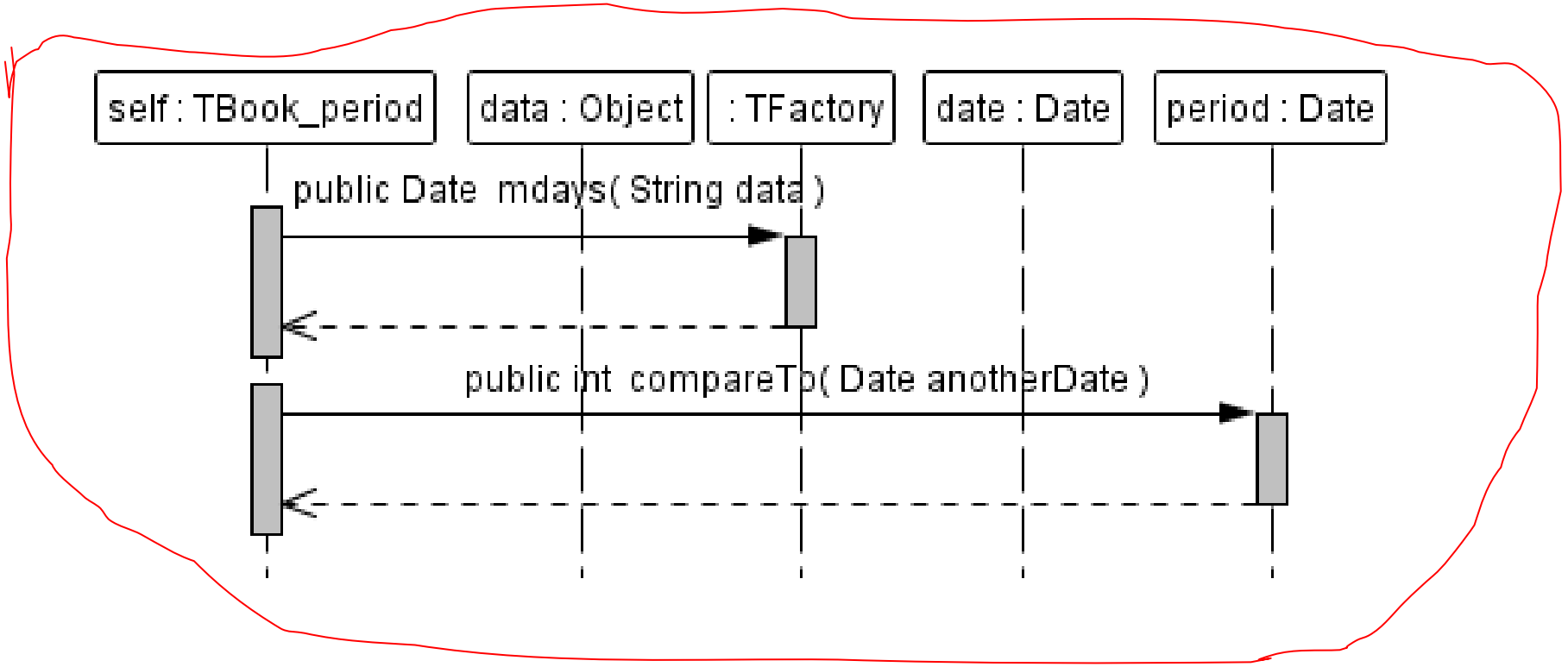
TTitle_book_search_accessible_book

TTitle_book: public synchronized TBook search_accessible_book(Object data)



TBook_period_period_pass

TBook_period: public boolean period_pass(Object data)



Strategy

The third iteration (1)

```
package library1;

import java.io.Serializable;
import java.util.ArrayList;

public class TFacade implements Serializable {
/* Code of TFacade methods*/

public static void main(String t[])    {
      TFacade ap = new TFacade();
       String t1[] = {"1", "Author1","Title1", "ISBN1", "Publisher1"};
      String t2[] = {"1", "Author2","Title2", "ISBN2", "Publisher2"};
      String t3[] = {"1", "Author3","Title3", "ISBN3", "Publisher3"};
      String t4[] = {"3", "Author1","Title1", "ISBN1", "Publisher1", "Actor1"};
      String t5[] = {"3", "Author2","Title2", "ISBN2", "Publisher2", "Actor2"};
      String t6[] = {"3", "Author4","Title4", "ISBN4", "Publisher4", "Actor4"};
       ap.add_title_book(t1);
      ap.add_title_book(t2);
      ap.add_title_book(t2);
      ap.add_title_book(t3);
      ap.add_title_book(t4);
      ap.add_title_book(t5);
      ap.add_title_book(t5);
      ap.add_title_book(t6);
      String lan = ap.getmTitle_books().toString();
      System.out.println(lan);
```

The third iteration (2)

```
String d1[] = {"0", "ISBN1"};
String d2[] = {"0", "ISBN2"};
String d3[] = {"0", "ISBN5"};
String d4[] = {"2", "ISBN1", "Actor1"};
String d5[] = {"2", "ISBN4", "Actor4"};
String tr1[] = {"0", "1"};
String tr2[] = {"0", "2"};
String tr3[] = {"1", "3", "3"};
String tr4[] = {"1", "2", "-1"};
 TTitle_book pom = ap.add_book(d1, tr1);
if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d2, tr1);
if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d2, tr1);
if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d2, tr2);
if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d3, tr2);
if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d4, tr3);
if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d4, tr3);
if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d4, tr4);
if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d5, tr2);
if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
System.out.println();
```

The third iteration (3)

```
 System.out.print("\nSearching of a title");
 System.out.print(ap.Search_title_book(t5).toString());
 System.out.print("\nSearching of an accessible book of a select title");
 System.out.print(ap. Search_accessible_book(d4, "2").toString());
 System.out.println();
 }
}
```

The third iteration (4) - Output window

[
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: 1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 2][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:47:35 CET 2011][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:47:35 CET 2011][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:47:35 CET 2011,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Sat Feb 12 18:47:35 CET 2011][
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2]

Searching of a title
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Actor: Actor2
Searching of an accessible book of a select title
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Sat Feb 12 18:47:35 CET 2011

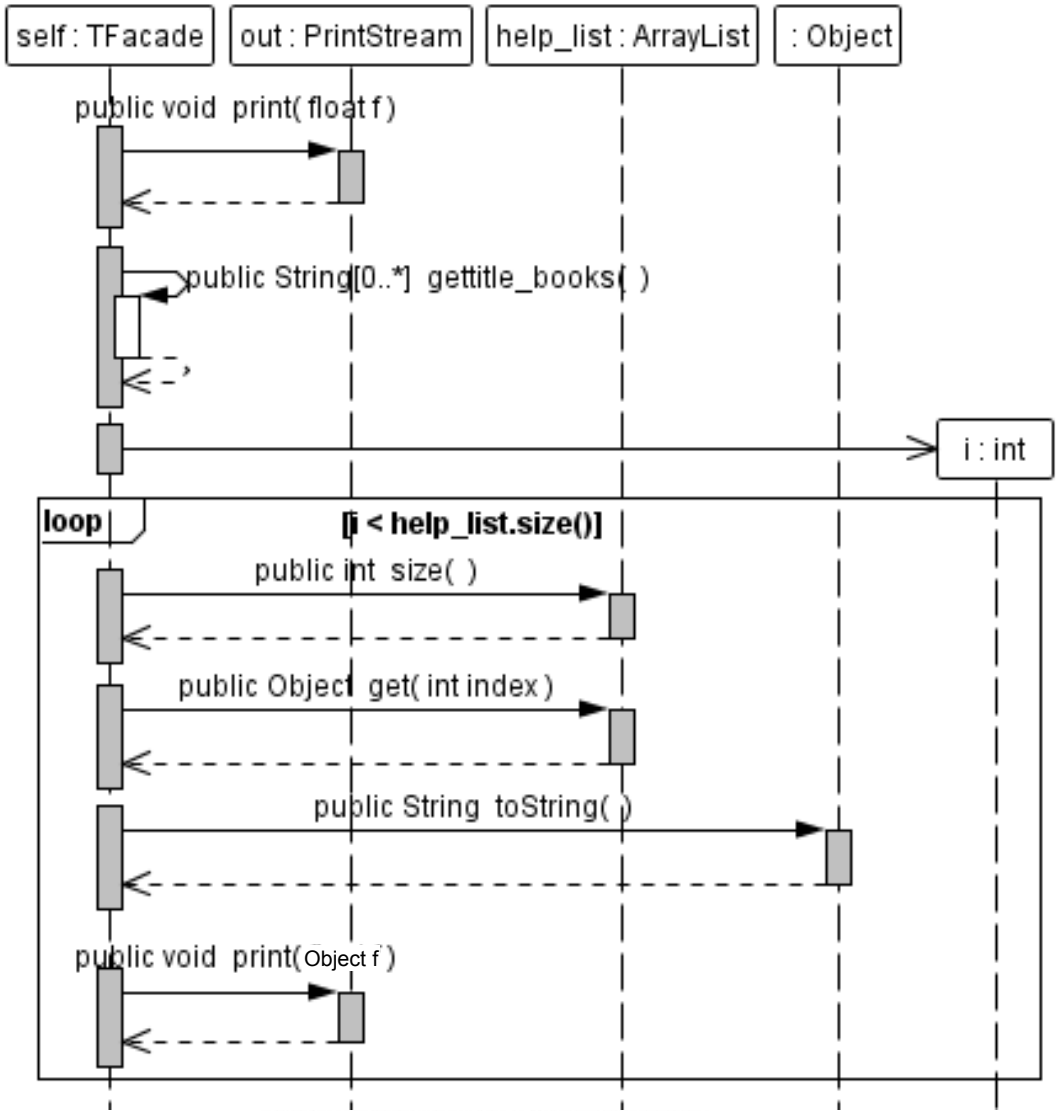
Fourth Iteration

Create the code of methods based on the class diagram and sequence diagrams.

Next you must build and run program with the content of the main method of TFacade class defined on the further slide, as the acceptance test.

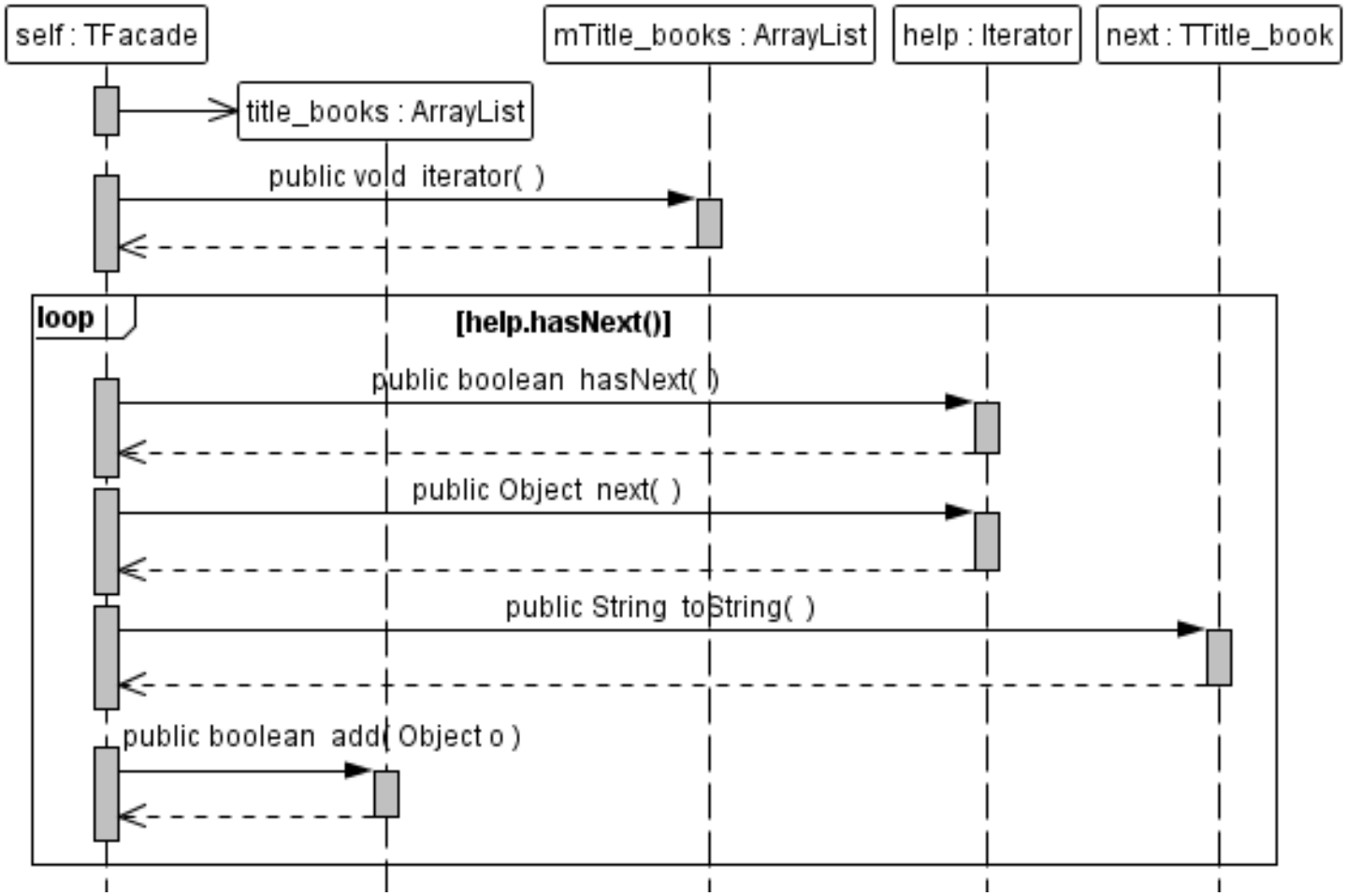
Console presentation

Temp_TFacade_Print_title_books TFacade: public synchronized void Print_title_books()

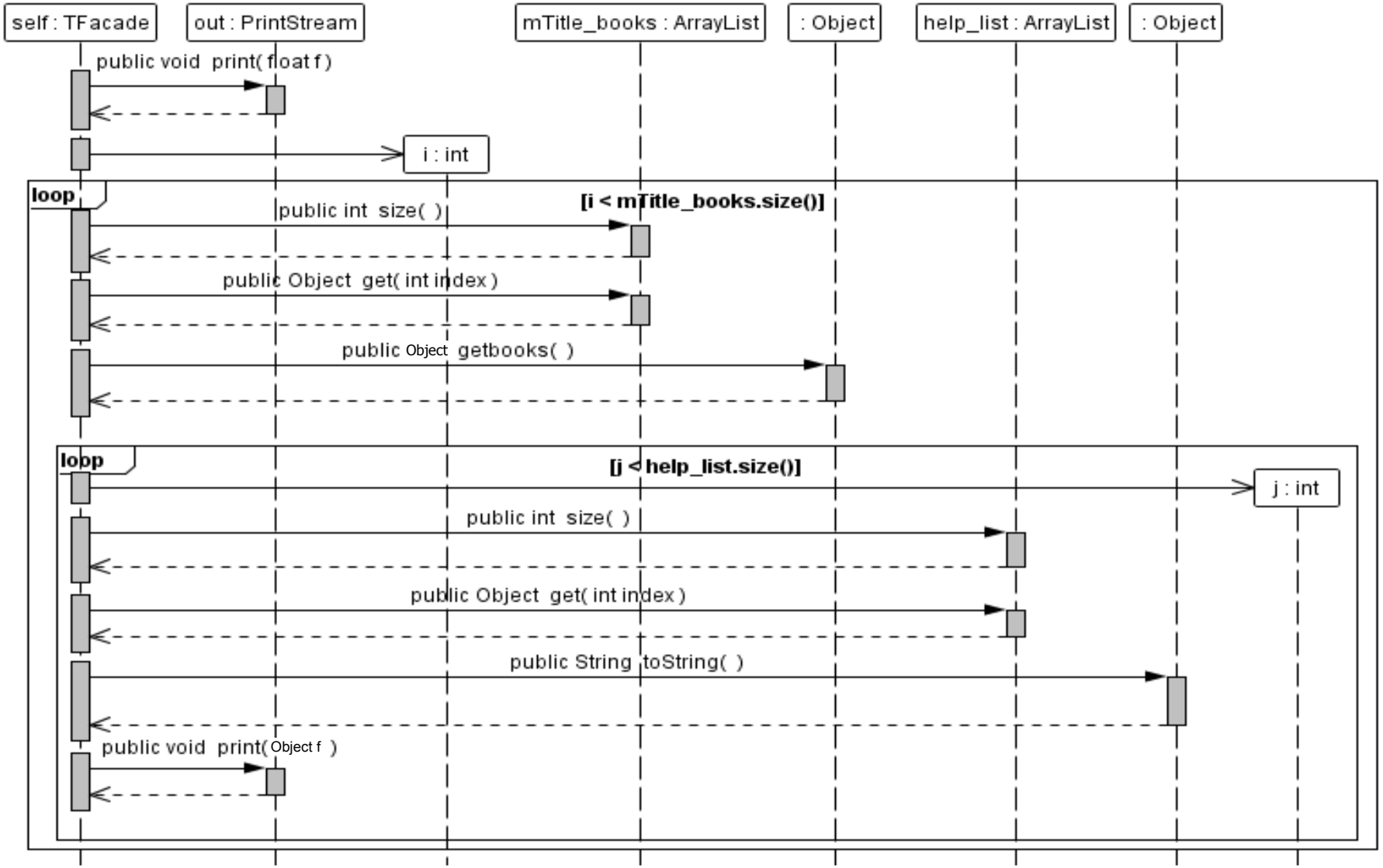


TFacade_gettitle_books

TFacade: public synchronized ArrayList<String> gettitle_books()

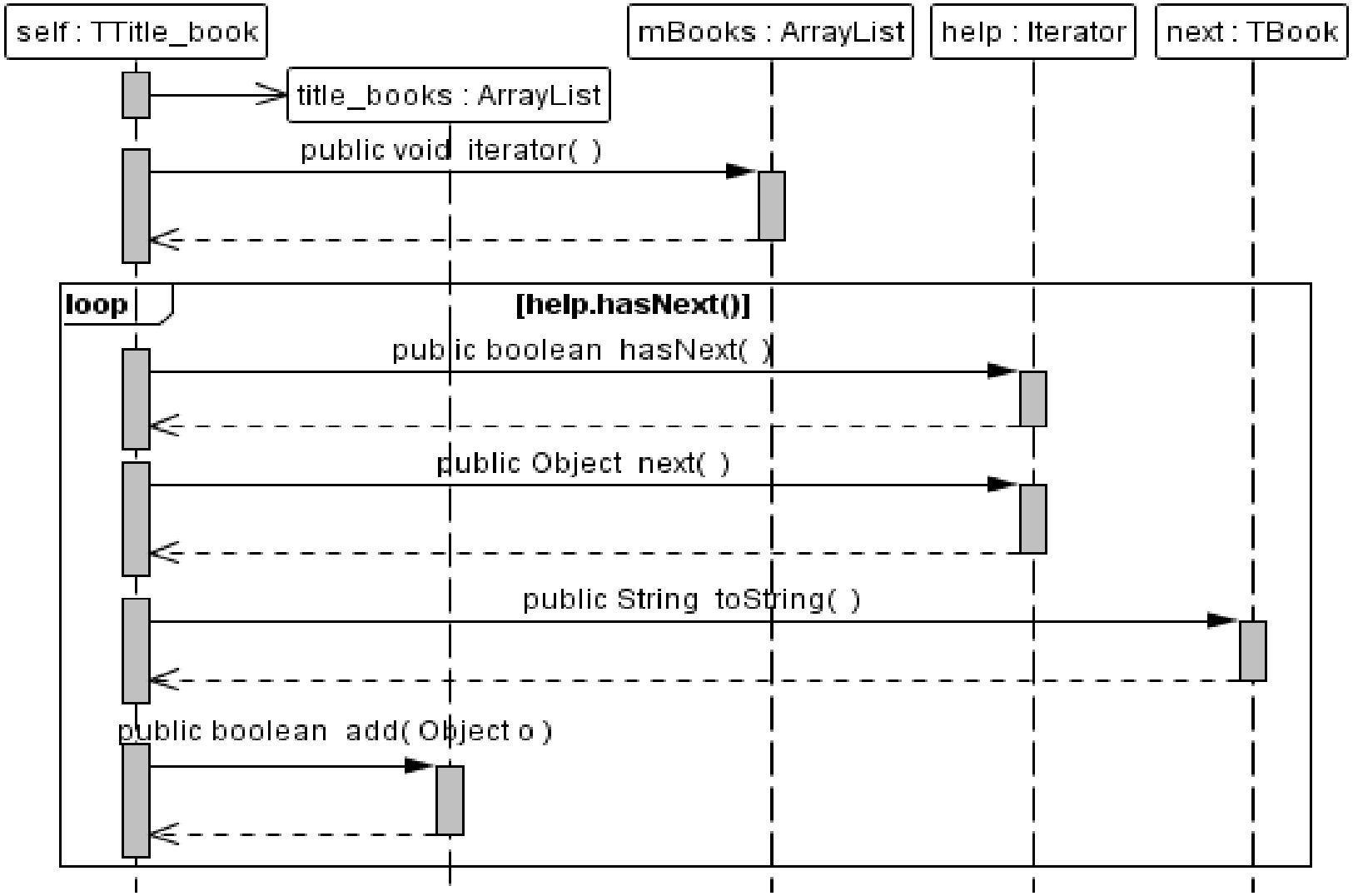


Temp_TFacade_Print_books TFacade: public synchronized void Print_books()



TTitle_book_getbooks

TTitle_book: public ArrayList<String> getbooks()



The fourth iteration (1)

```
package library1;

import java.io.Serializable;
import java.util.ArrayList;

public class TFacade implements Serializable {
/* Code of TFacade methods*/

public static void main(String t[])    {
      TFacade ap = new TFacade();
       String t1[] = {"1", "Author1","Title1", "ISBN1", "Publisher1"};
      String t2[] = {"1", "Author2","Title2", "ISBN2", "Publisher2"};
      String t3[] = {"1", "Author3","Title3", "ISBN3", "Publisher3"};
      String t4[] = {"3", "Author1","Title1", "ISBN1", "Publisher1", "Actor1"};
      String t5[] = {"3", "Author2","Title2", "ISBN2", "Publisher2", "Actor2"};
      String t6[] = {"3", "Author4","Title4", "ISBN4", "Publisher4", "Actor4"};
       ap.add_title_book(t1);
      ap.add_title_book(t2);
      ap.add_title_book(t2);
      ap.add_title_book(t3);
      ap.add_title_book(t4);
      ap.add_title_book(t5);
      ap.add_title_book(t5);
      ap.add_title_book(t6);
      String lan = ap.getmTitle_books().toString();
      System.out.println(lan);
```

The fourth iteration (2)

```
String d1[] = {"0", "ISBN1"};
 String d2[] = {"0", "ISBN2"};
 String d3[] = {"0", "ISBN5"};
 String d4[] = {"2", "ISBN1", "Actor1"};
 String d5[] = {"2", "ISBN4", "Actor4"};
 String tr1[] = {"0", "1"};
 String tr2[] = {"0", "2"};
 String tr3[] = {"1", "3", "3"};
 String tr4[] = {"1", "2", "-1"};
  TTitle_book pom = ap.add_book(d1, tr1);
 if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d2, tr1);
 if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d2, tr1);
 if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d2, tr2);
 if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d3, tr2);
 if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d4, tr3);
 if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d4, tr3);
 if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d4, tr4);
 if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
pom = ap.add_book(d5, tr2);
 if (pom != null) {        System.out.print(pom.getmBooks().toString());      }
 System.out.println();
```

The fourth iteration (3)

```
 ap.Print_title_books();
 ap.Print_books();
 System.out.print("\nSearching of a title");
 System.out.print(ap.Search_title_book(t5).toString());
 System.out.print("\nSearching of an accessible book of a select title");
 System.out.print(ap. Search_accessible_book(d4, "2").toString());
 System.out.println();
 }
}
```

The third iteration (4) - Output window

```
[
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: 1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 2][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:51:45 CET 2011][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:51:45 CET 2011][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:51:45 CET 2011,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Sat Feb 12 18:51:45 CET 2011][
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2]

Titles of book
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
Books
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: 2
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Wed Feb 16 18:51:45 CET 2011
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Sat Feb 12 18:51:45 CET 2011
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2
Searching of a title
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Actor: Actor2
Searching of an accessible book of a select title
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Sat Feb 12 18:51:45 CET 2011
```