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. Prepare the initial design model based on use case diagram and class diagram with classes, which are identified during analysis of commonality and variability.
5. Generate the initial code from this initial design model.
6. Design model of the business service sub-tier based on the class diagram and sequence diagram, created by iteratively driven development of use cases. 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

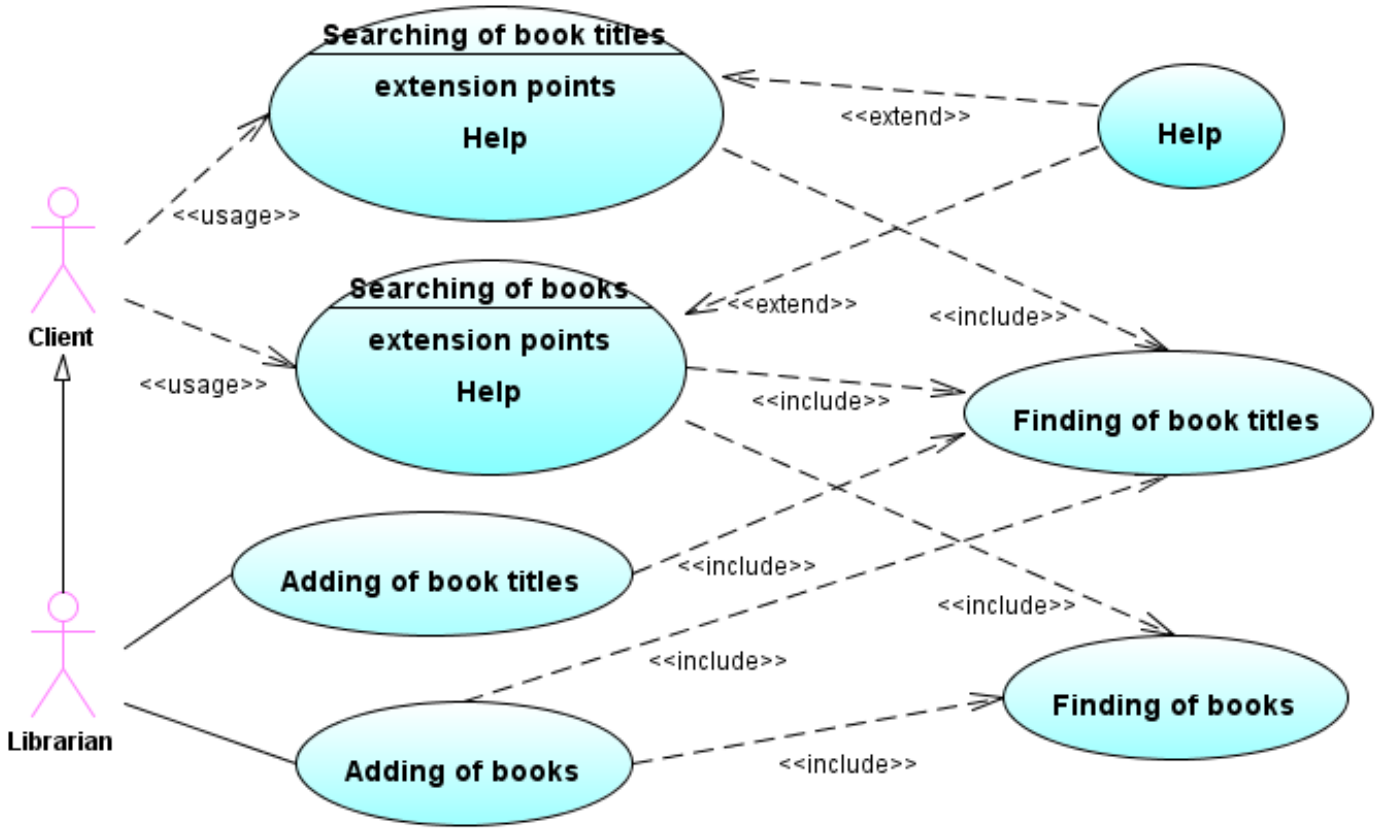
2.1.functional

- 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 Finding 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:

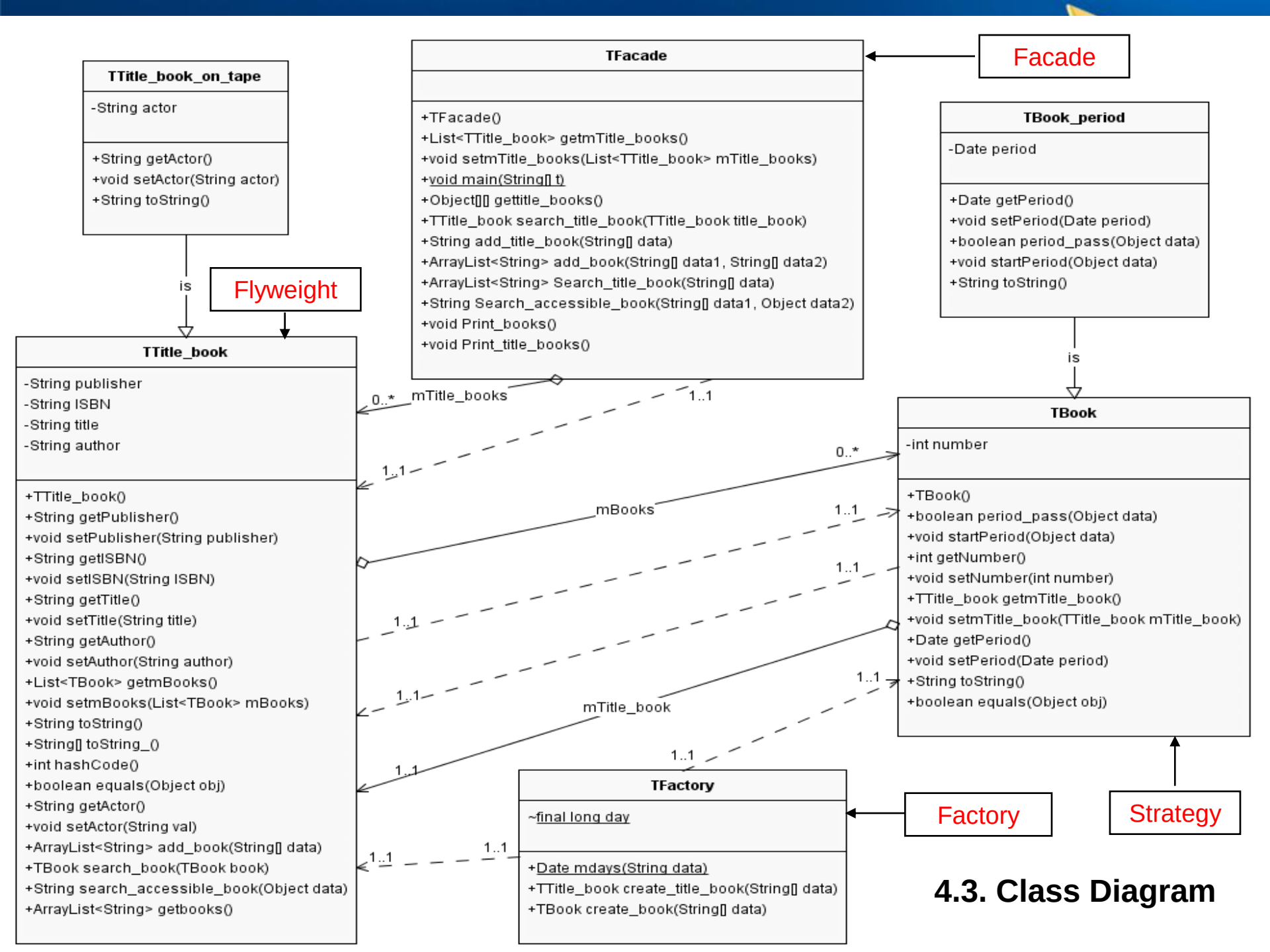
1. It specifies the attributes of the title: ISBN as the mandatory plus actor, if you are looking for is the title of the book as audio book. It creates a pattern title to searching for the real one.
2. It executes **the Finding of book titles use case** providing the pattern title. It makes sure that 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. /4.1. 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.

4.2. An analysis of commonality and variability

Use case	Attributies	Inheritance	Association	Class	
				commonality	variability
Finding of book titles	Publisher, ISBN, Title, author	TTitle_book_on_tape extends TTitle_book		TTitle_book	TTitle_book TTitle_book_on_tape
	Publisher, ISBN Title, author, actor				
Searching of book titles	ISBN	TTitle_book_on_tape extends TTitle_book		TTitle_book	TTitle_book TTitle_book_on_tape
	ISBN, actor				
Finding of books	ISBN number of book	TTitle_book_on_tape extends TTitle_book TBook_period extends TBook_period	1	TTitle_book	TTitle_book TTitle_book_on_tape
	ISBN, actor number of book		0. .*	TBook	TBook TBook_period
Searching of books		TTitle_book_on_tape extends TTitle_book	1	TTitle_book	TTitle_book TTitle_book_on_tape
		TBook_period extends TBook_period	0 ..*	TBook	Tbook TBook_period
Adding of book titles		TTitle_book_on_tape extends TTitle_book		TTitle_book	TTitle_book TTitle_book_on_tape
Adding of books		TTitle_book_on_tape extends TTitle_book	1	TTitle_book	TTitle_book TTitle_book_on_tape
		TBook_period extends TBook_period	0 ..*	TBook	TBook TBook_period
All use cases			1	TFacade	TFacade
			0 ..*	TTitle_book	TTitle_book TTitle_book_on_tape

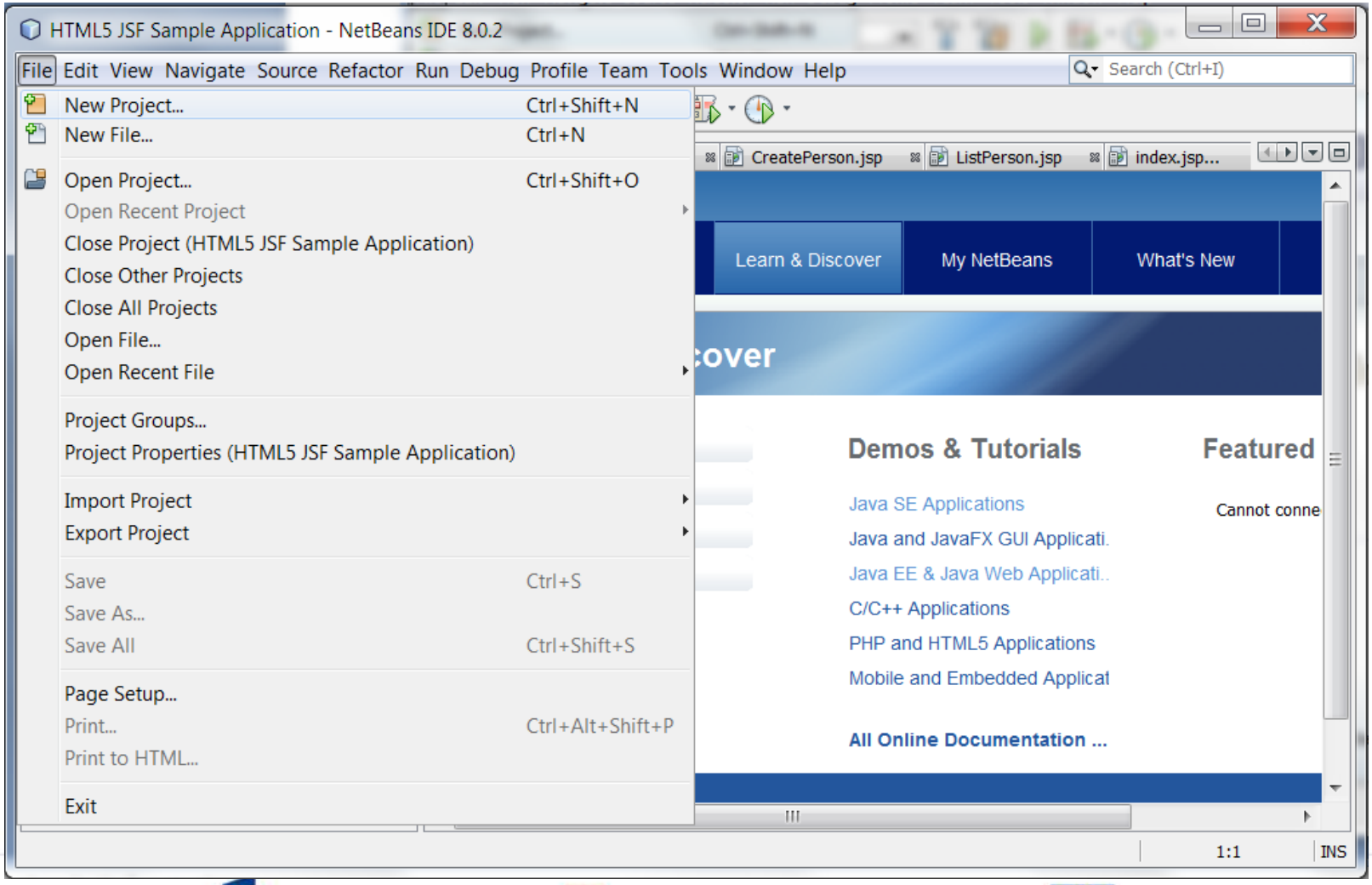


4.3. Class Diagram

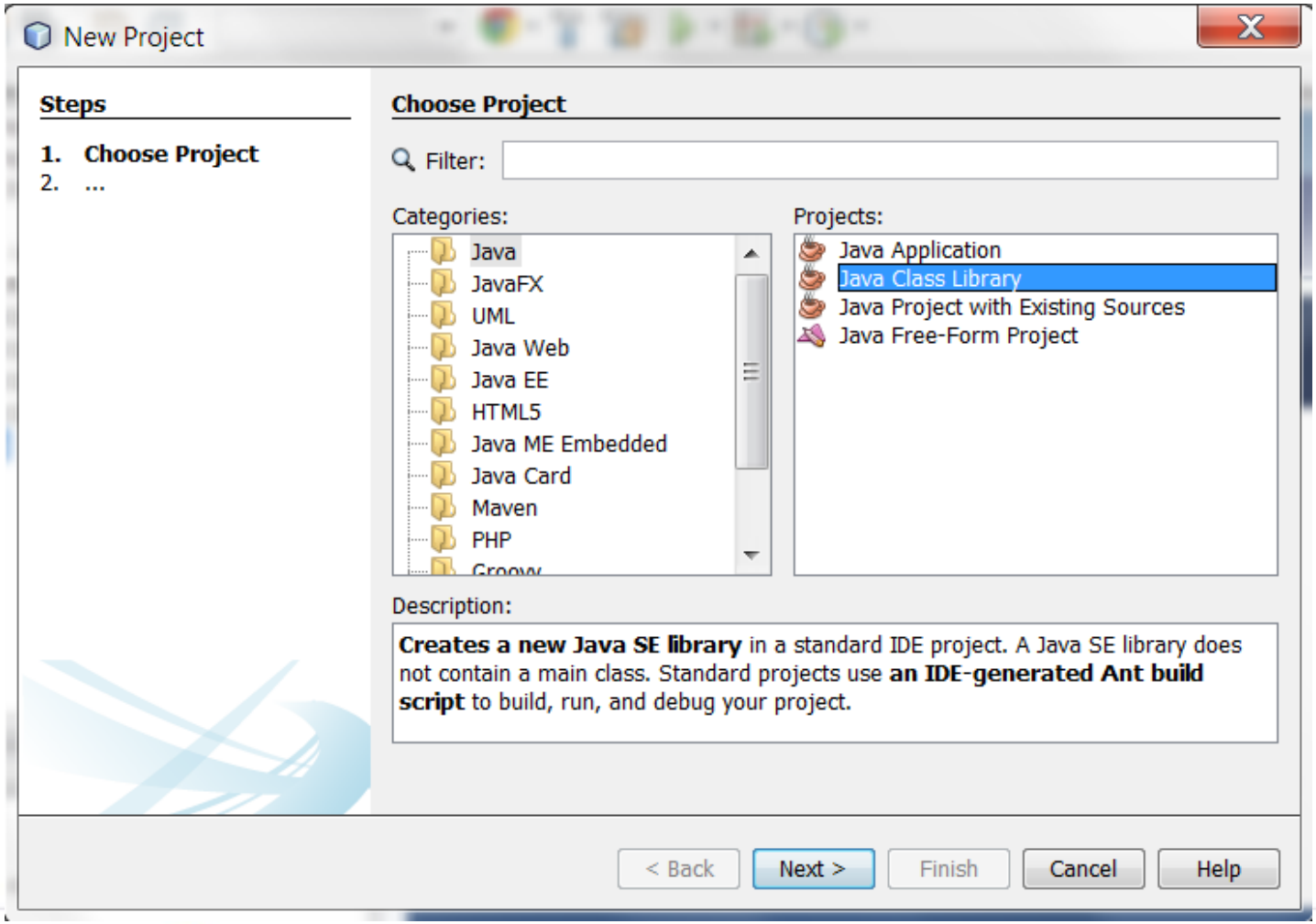
5. Generate code classes based on class diagram developed in the analysis phase (download from the link of the lab1: http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/analizasi/Library1_UML1.rar)

1. It must be created a new Java project such as the **Java Class Library** project named **Library1_JSE** (you must chose in the following order: the File, New Project, Java, Java Class Library, Next, insert the name of the project as the Library1_JSE name, select a folder in the Project Location where is the **Library1_UML1** project, Finish) - 14-17 slides
2. You must open the UML Diagrams project called **Library1_UML1,** provided in an annex to the laboratory - 18-21 slides
3. You should generate code from the **Library1_Model1.cdg diagram** (in the Edit Window right click on the Library1_Model1.cdg diagram of the Library1_UML1 project and select the **Generate code** item from the pop-up menu). In the **Generate code** form, you must select the Target Project as **the Library1_JSE**, and press **OK** – 22 slide
4. In all generated files in Library1_JSE project 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) – 23-26 slides
5. **Finally, you must prepare some additional changes, accordingly to the some tips** – 27- 41 slides
6. You must prepare copy of the **Library1_JSE** project as the **Library1_JSE2** project – 42 slide

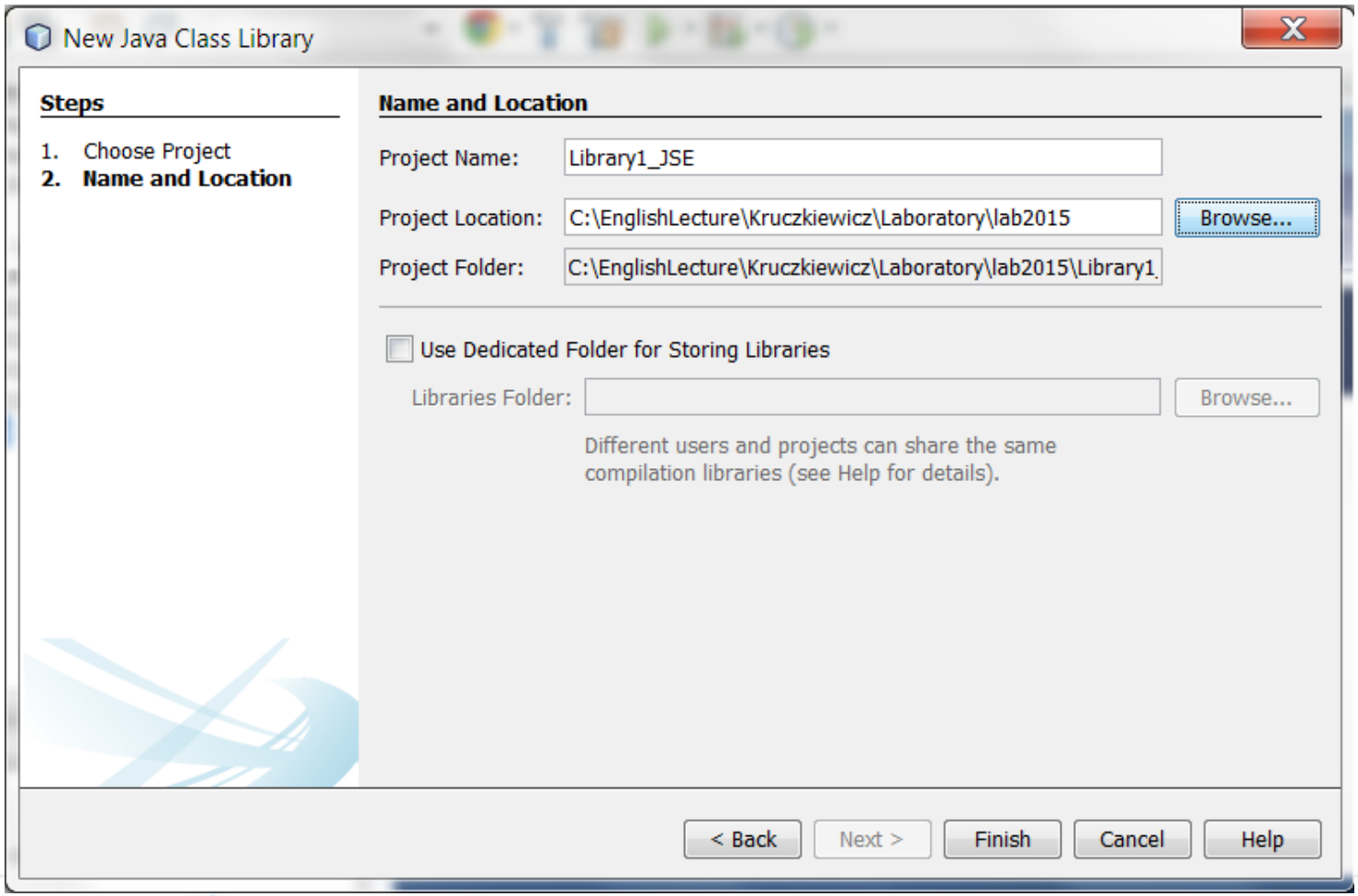
5.1/5.1.1. Create empty Java Class Library Project



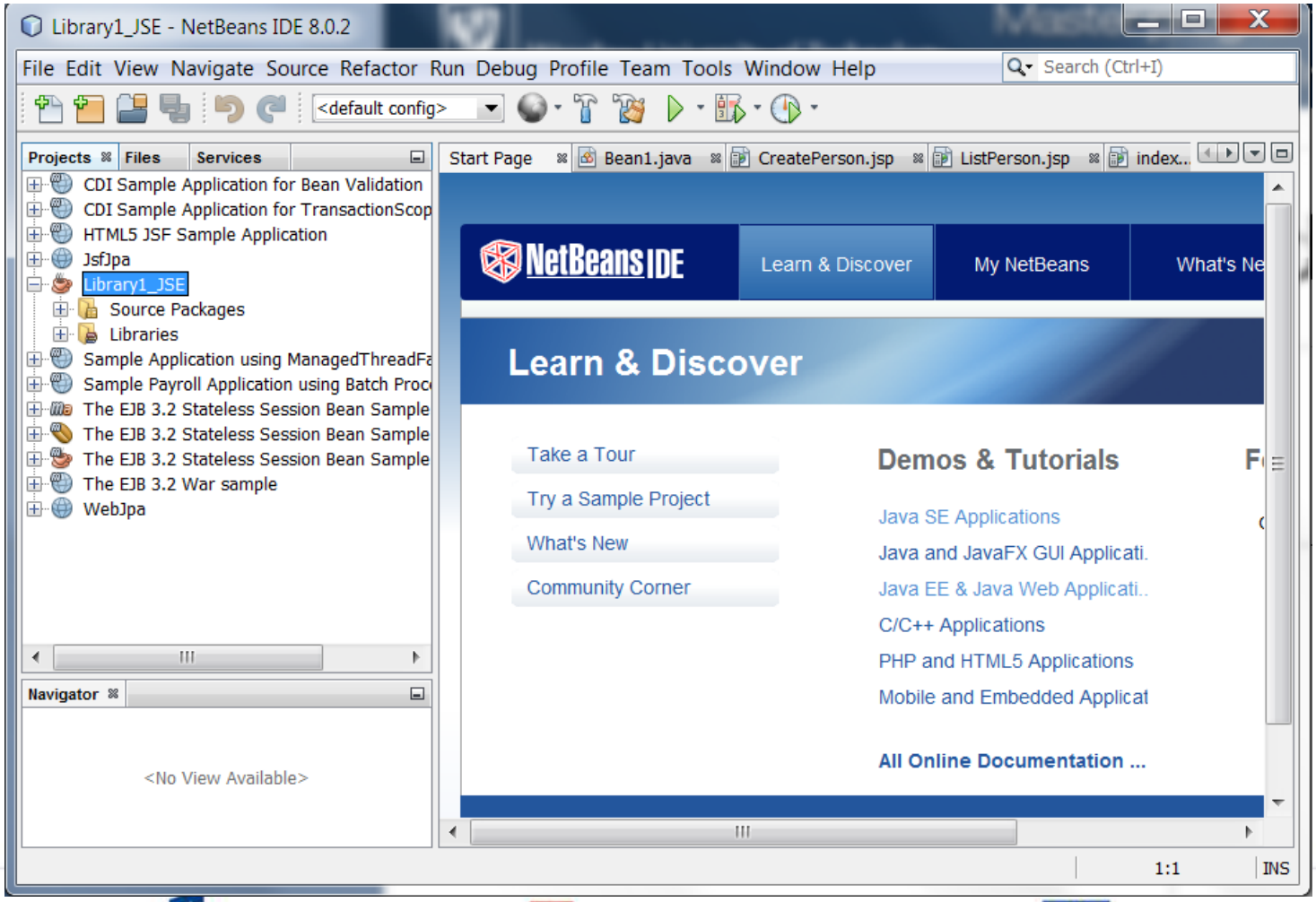
5.1.2. Create empty Java Class Library Project



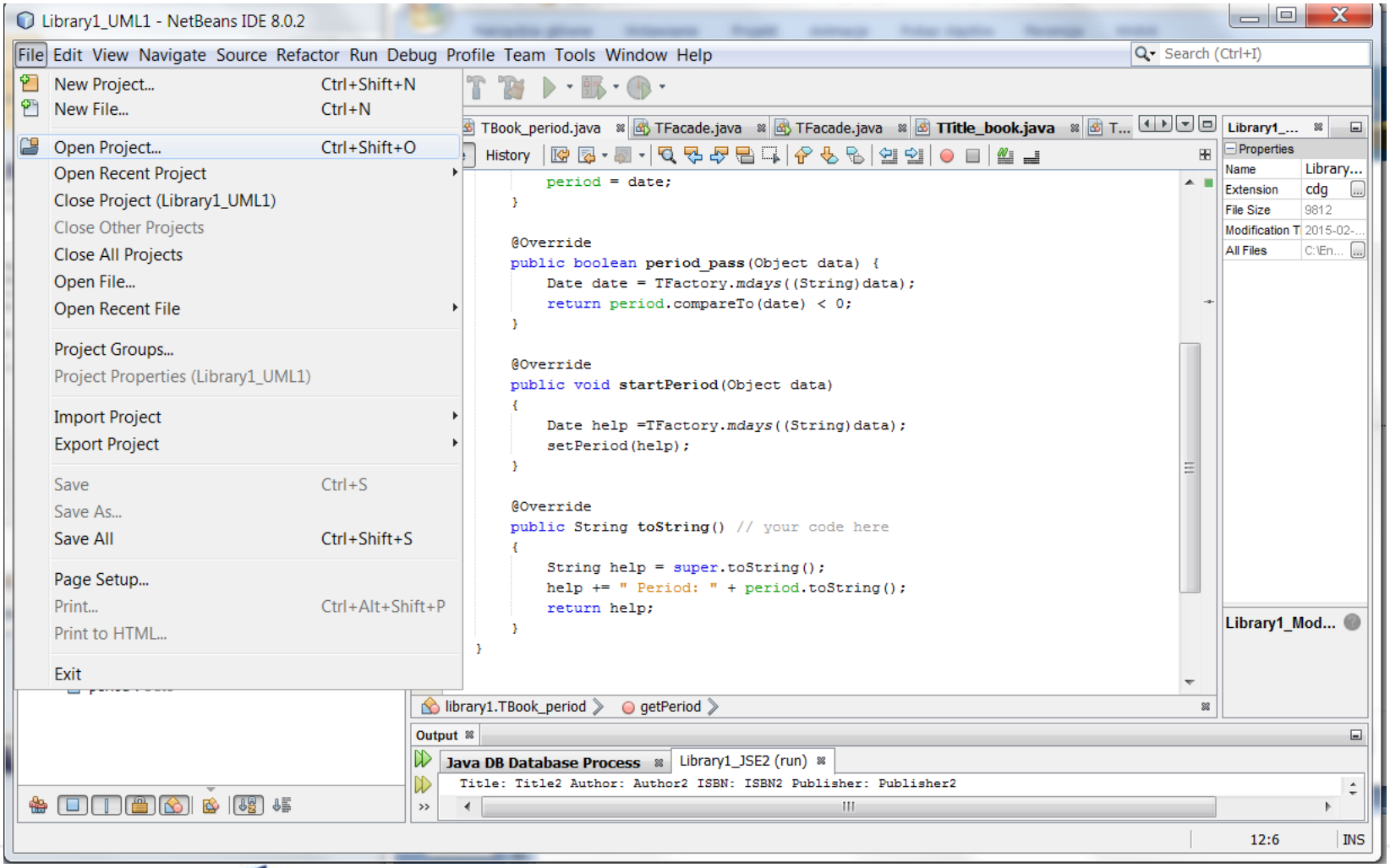
5.1.3. Create empty Java Class Library Project (1.3)



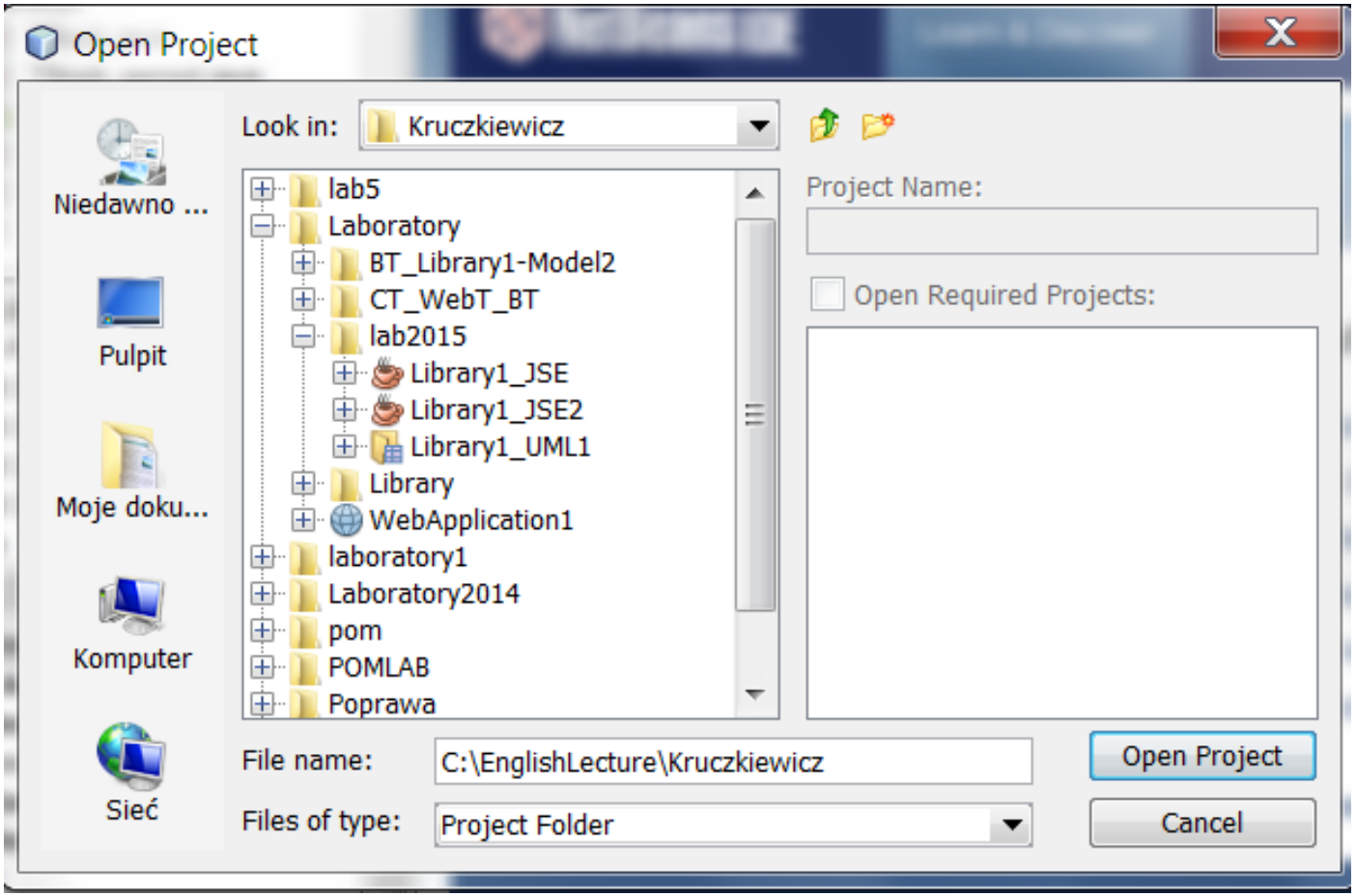
5.1.4. Create empty Java Class Library Project (1.4)



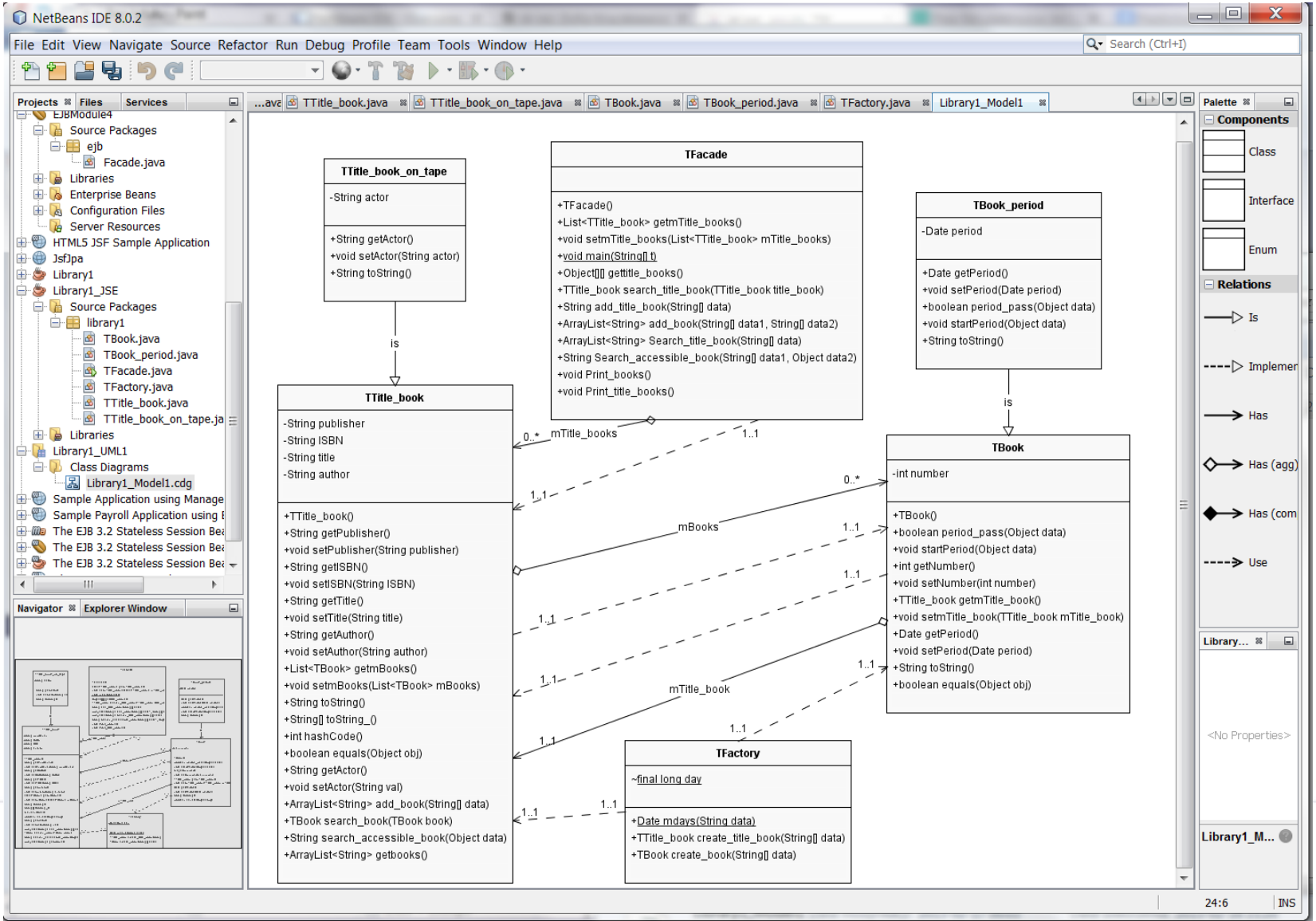
5.2./5.2.1. Open the UML Diagrams Project (2.1)



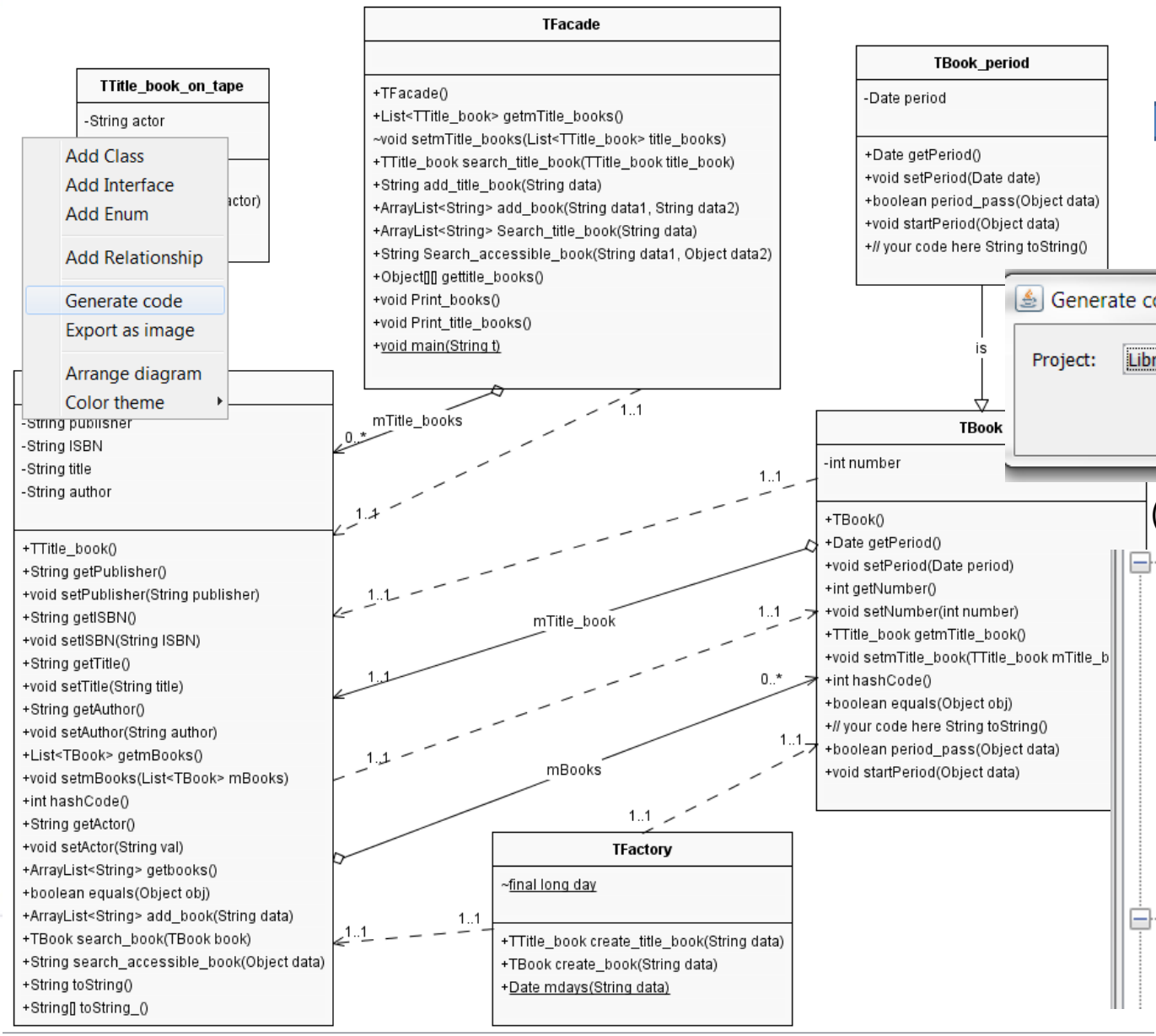
5.2.2. Open the UML Diagrams Project



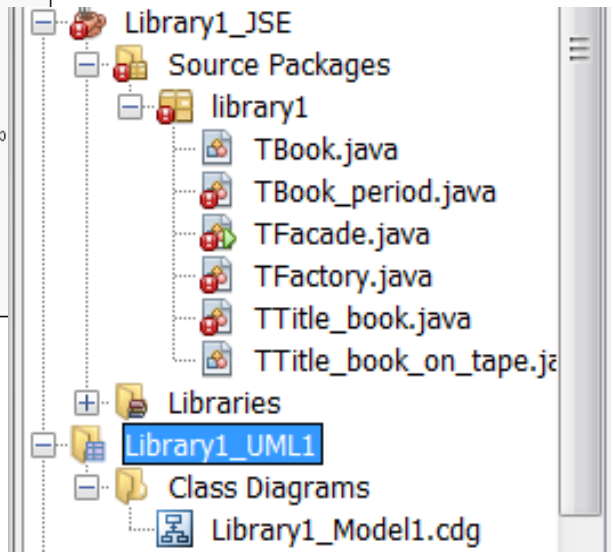
5.2.3. Open the UML Diagrams Project



5.3. Java code generation from the class diagram to the selected Library1_JSE Java Class Library type project

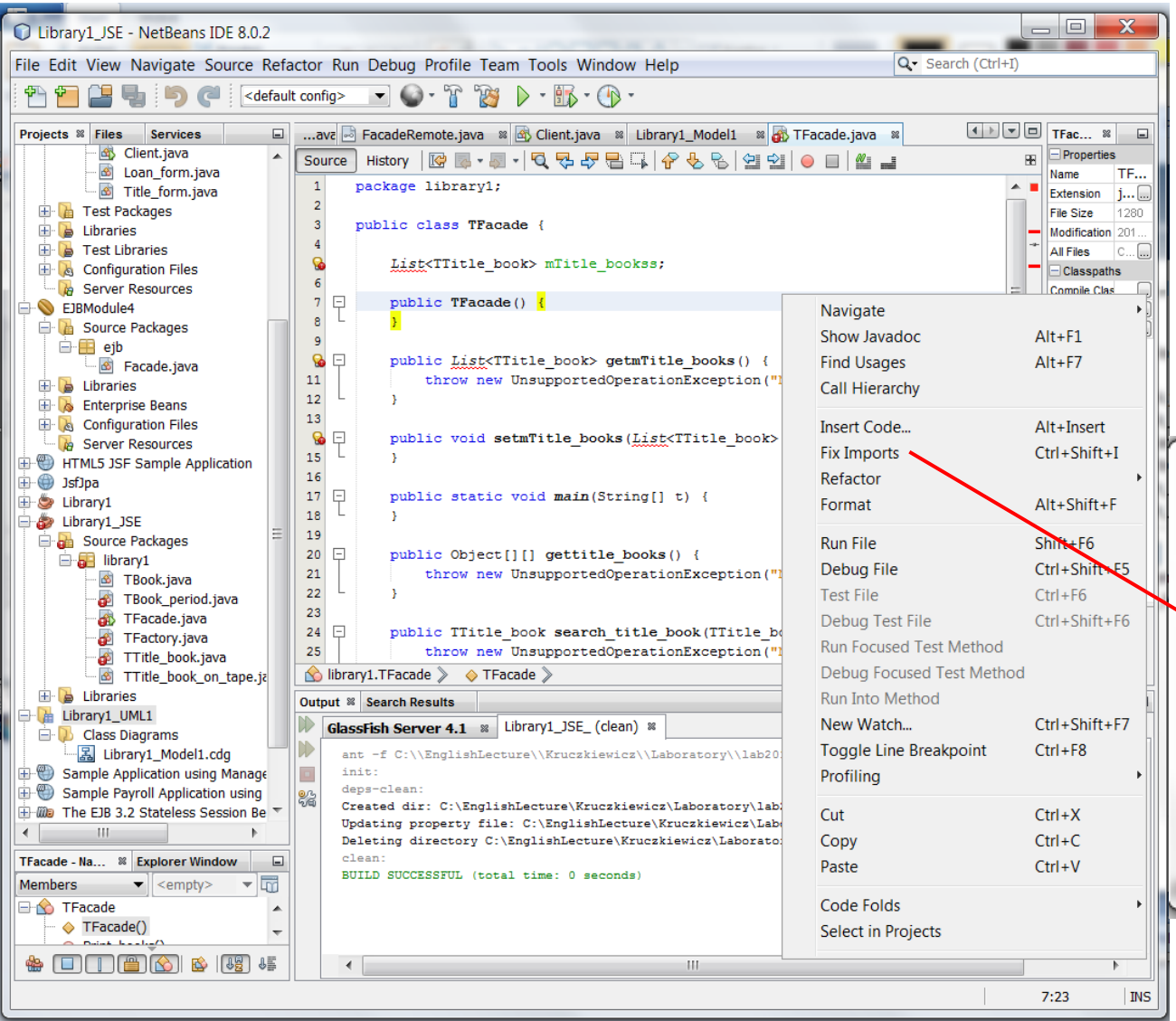


(3.2. Result of generating code

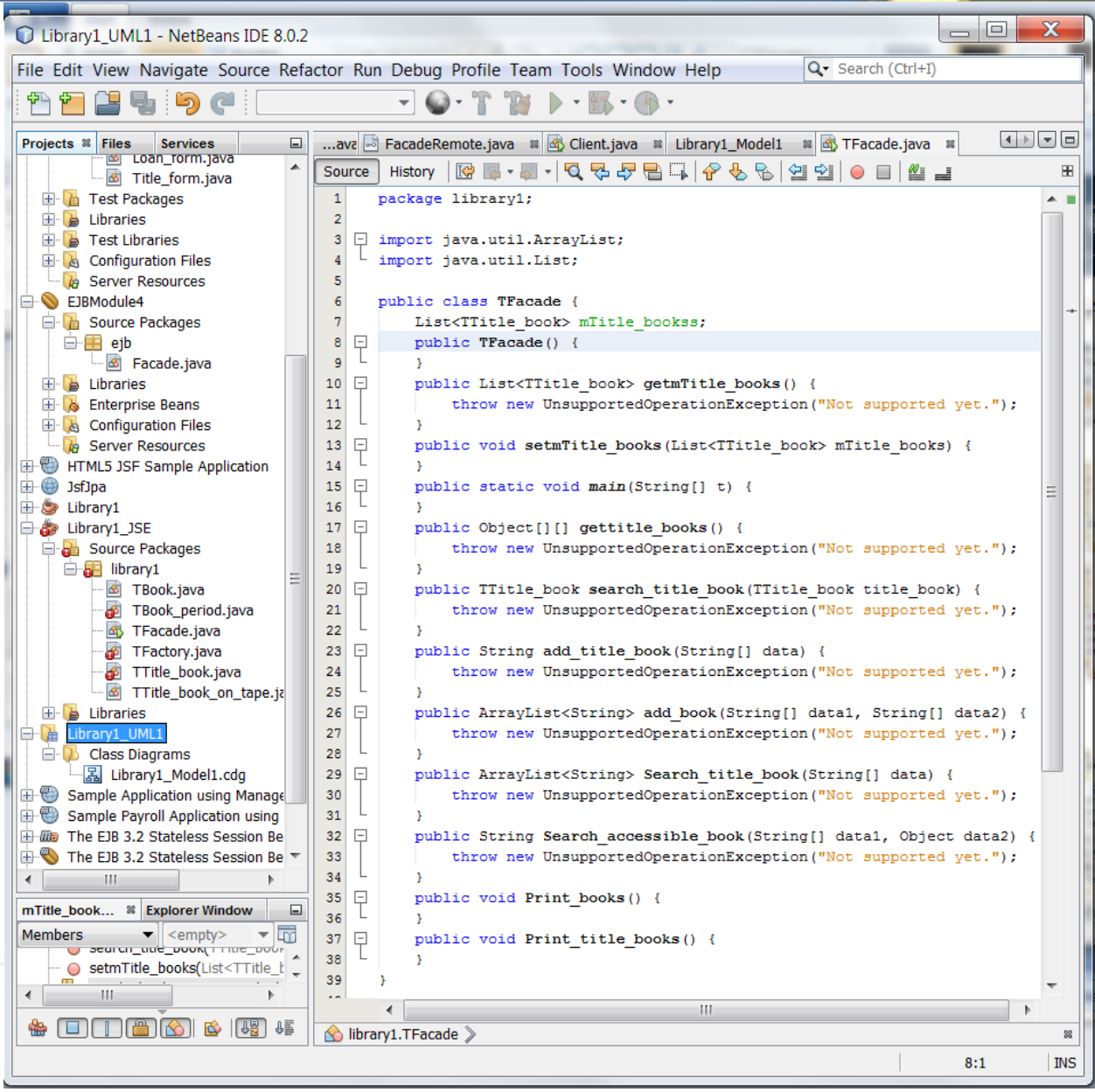


5.4./ 5.4.1. The result of generating code – using Fix Imports.. for removing faults from the TFacade class

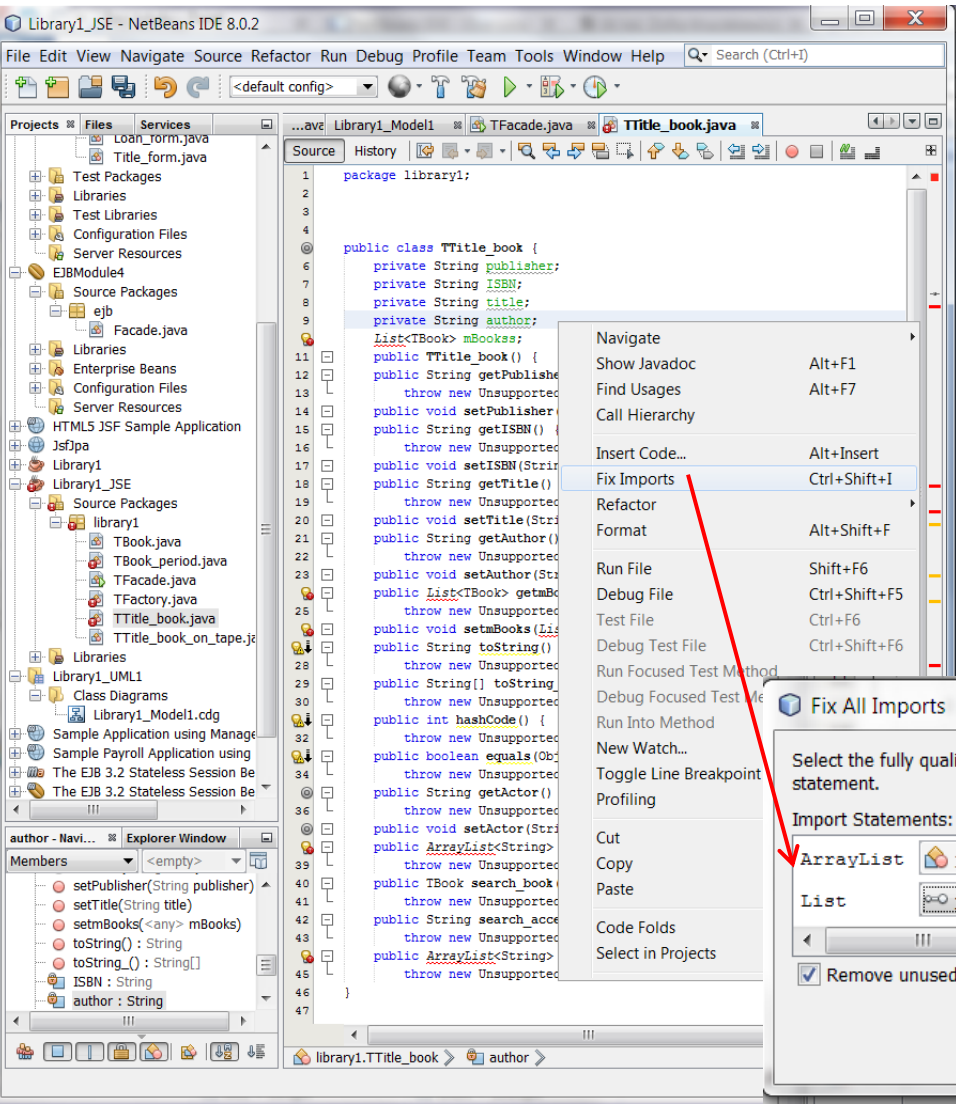
(right click on the code editor, select the Fix Imports… item and select the ArrayList and List imports)



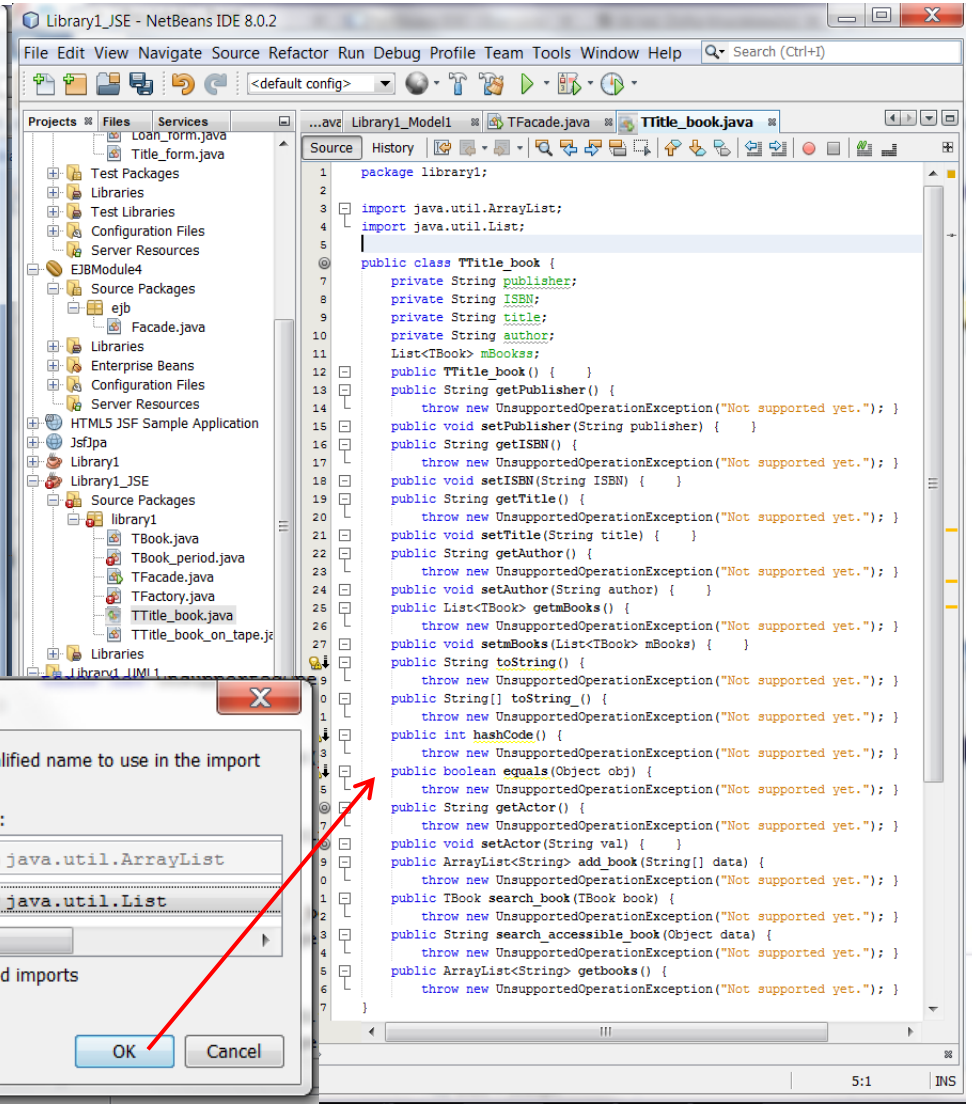
5.4.2. Removing faults – result of using Fix imports for removing faults from the TFacade class



5.5./ 5.5.1. Removing faults - using Fix imports for removing faults from the TTitle_book class

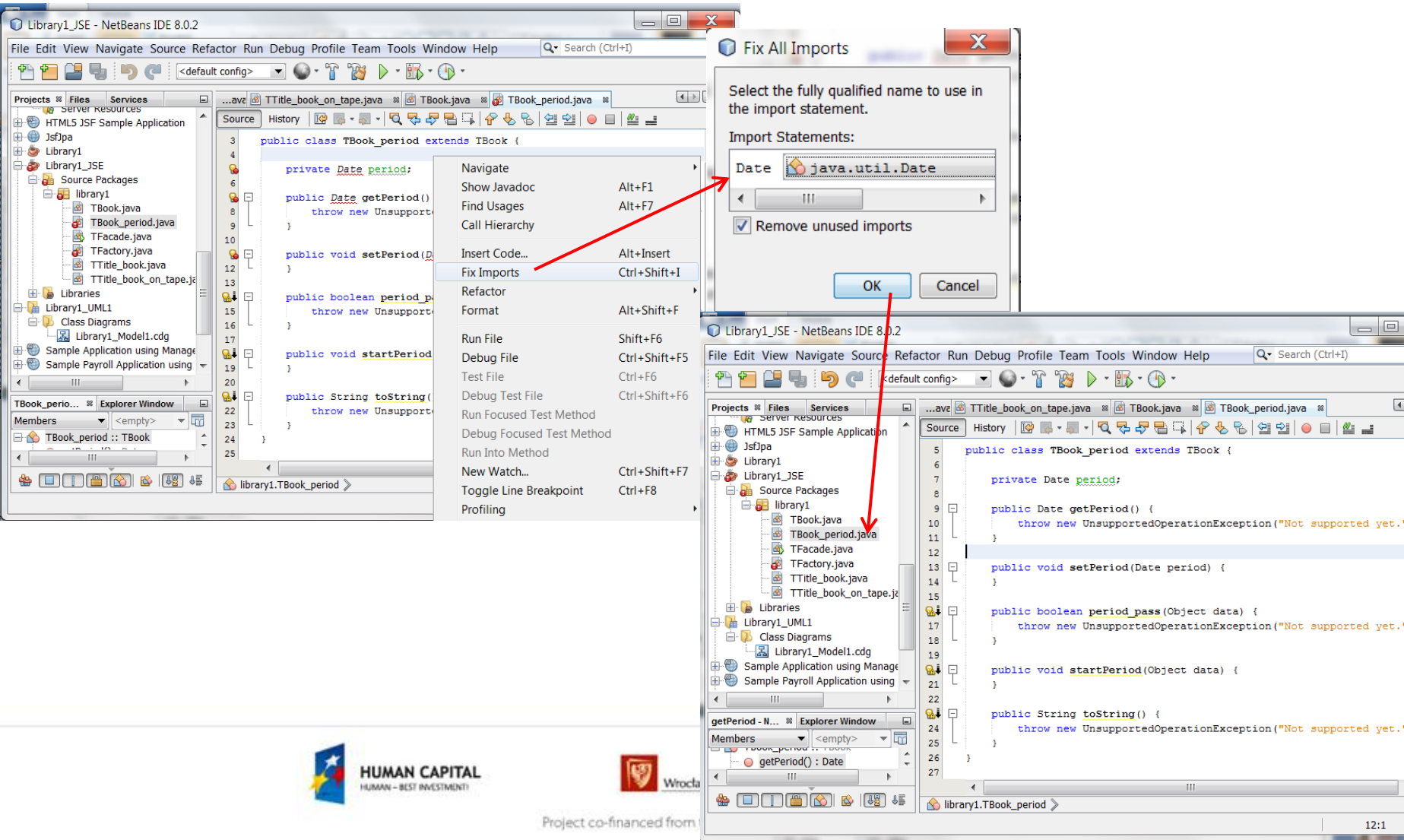


5.5.2. Removing faults - result of using Fix imports for removing faults from the TTitle_book class



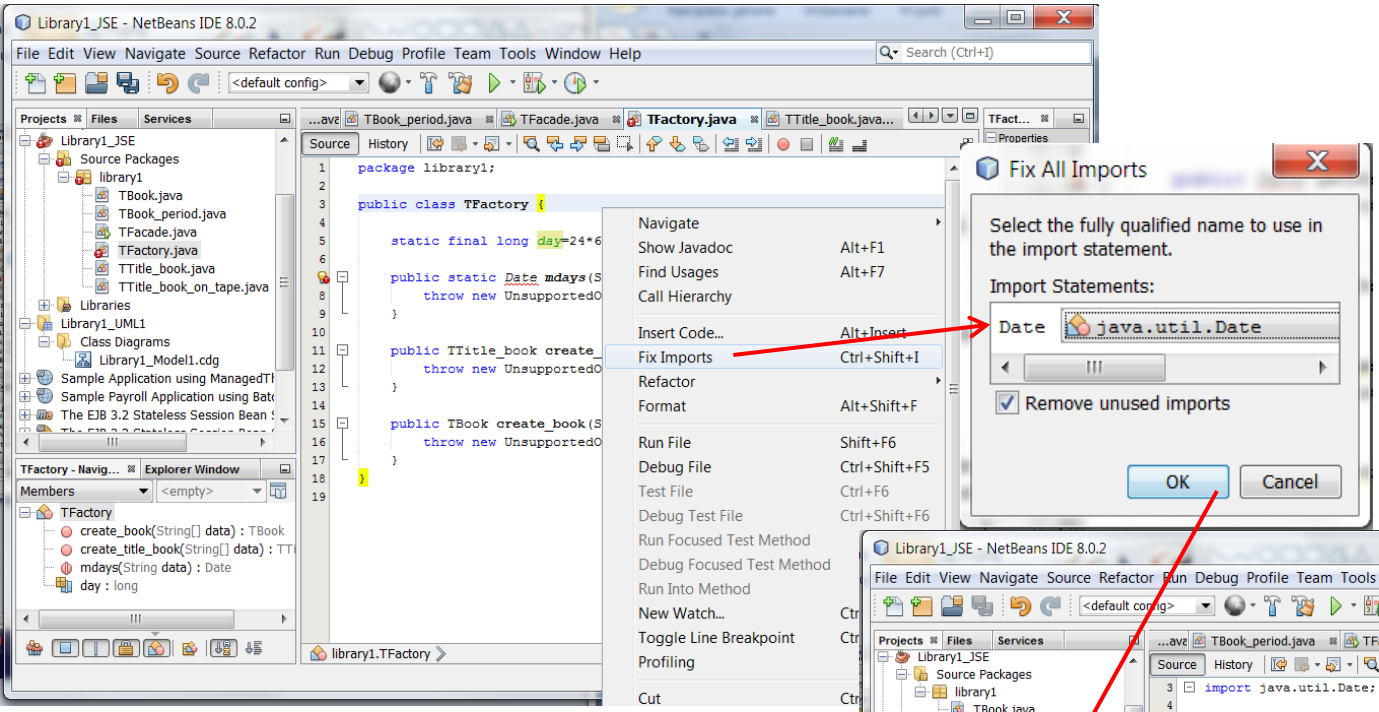
5.6./ 5.6.1. Removing faults - using Fix imports for removing faults from the TBook_period class

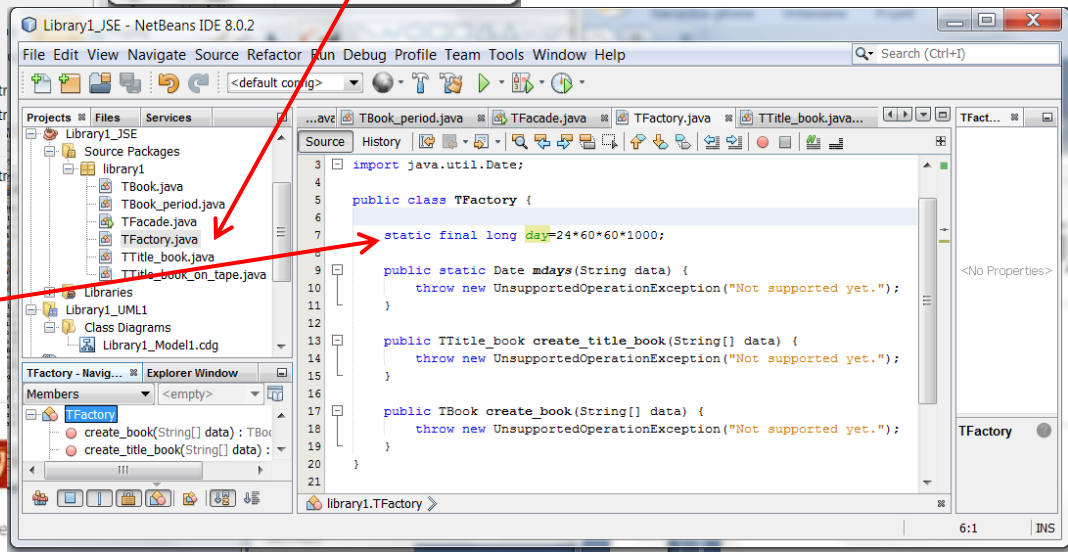
5.6.2. Removing faults - result of using Fix imports for removing faults from the TBook_period class



5.7./ 5.7.1. Removing faults - using Fix imports for removing faults from the TFactory class

5.7.2. Removing faults - result of using Fix imports for removing faults from the TFactory class



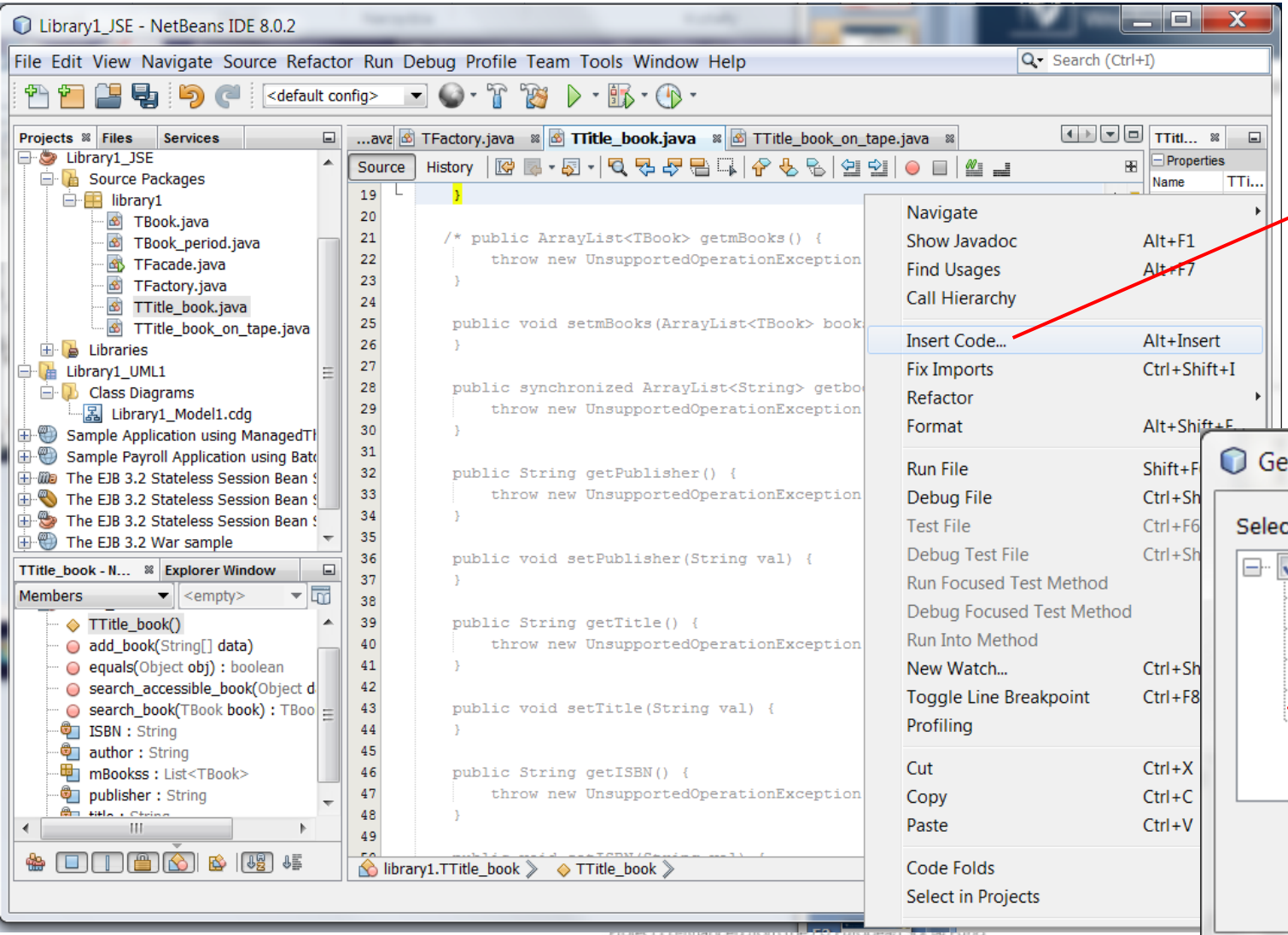


static final long day= 24*60*60*1000

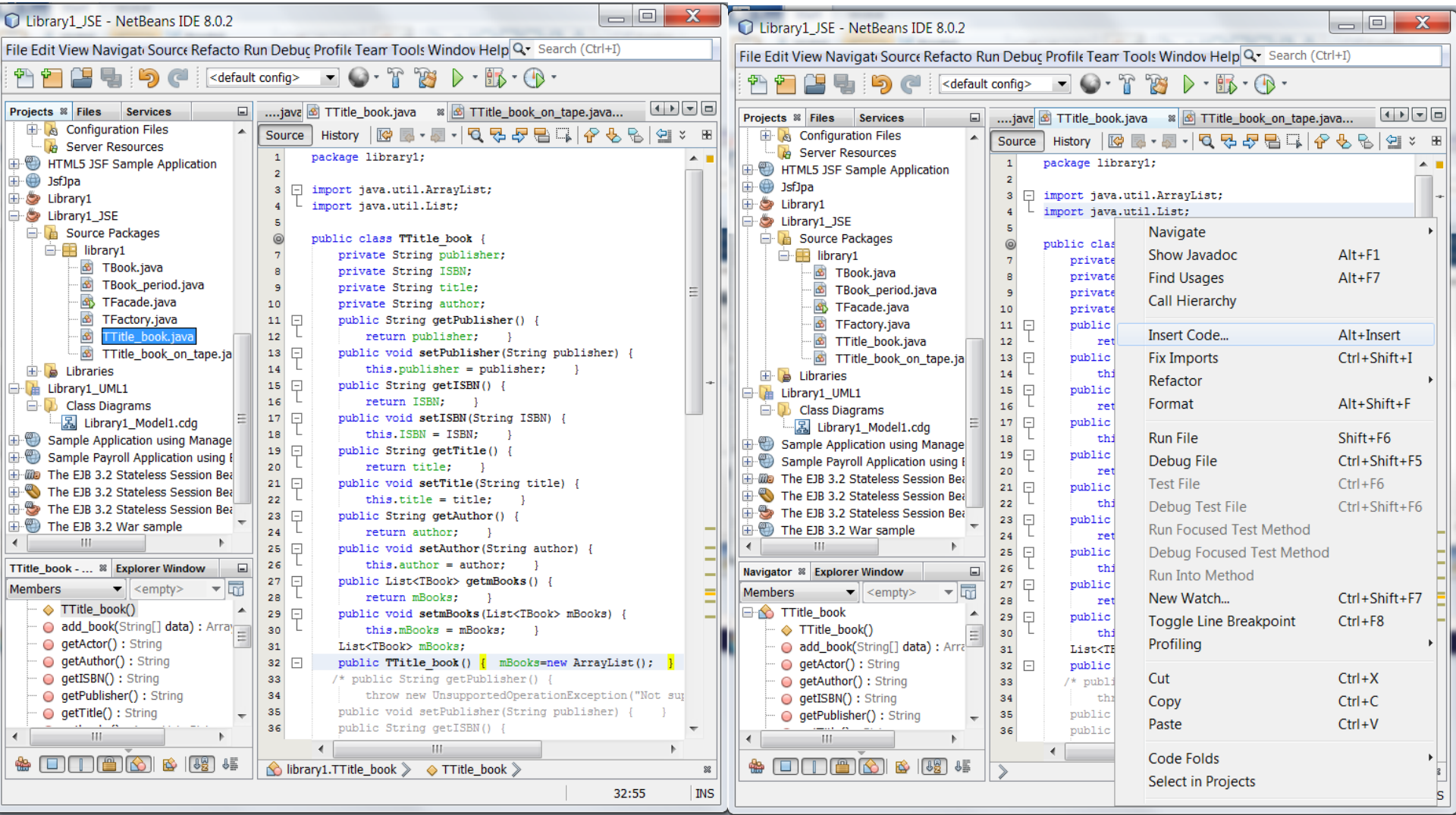
5.8./ 5.8.1. Code of the TTitle_book class – you must comment selected getter and setter methods of the TTitle_book class

```
package library1;

import java.util.ArrayList;
import java.util.List;

public class TTitle_book{
   private String publisher;
   private String ISBN;
   private String title;
   private String author;
   private List<TBook> mBooks;
   public TTitle_book ()                                   {  mBooks=new ArrayList<>();  }
    /*  public List<TBook> getmBooks ()       { throw new UnsupportedOperationException("Not supported yet.");   }
        public void setmBooks (List<TBook> books)          {   }
        public String getPublisher ()                      { throw new UnsupportedOperationException("Not supported yet."); }
        public void setPublisher (String val)              {   }
        public String getTitle ()                          { throw new UnsupportedOperationException("Not supported yet.");   }
        public void setTitle (String val)                  {   }
        public String getISBN ()                           { throw new UnsupportedOperationException("Not supported yet.");  }
        public void setISBN (String val)                   {   }
        public String getAuthor ()                         { throw new UnsupportedOperationException("Not supported yet.");  }
        public void setAuthor (String val)                 {   }
        public int hashCode ()                             { throw new UnsupportedOperationException("Not supported yet.");  }
        public boolean equals (Object obj)                 { throw new UnsupportedOperationException("Not supported yet."); }  */

   public String getActor ()                               { throw new UnsupportedOperationException("Not supported yet.");   }
   public void setActor (String val)                       {   }
  public ArrayList<String> getbooks()          {   }
  public String toString ()                                { throw new UnsupportedOperationException("Not supported yet.");   }
   public String[]  toString_ ()                           { throw new UnsupportedOperationException("Not supported yet.");   }
   public ArrayList<String> add_book (String[] data)                    {   }
   public TBook search_book (TBook book)                   { throw new UnsupportedOperationException("Not supported yet."); }
   public String search_accessible_book(Object data)       {   }
}
```

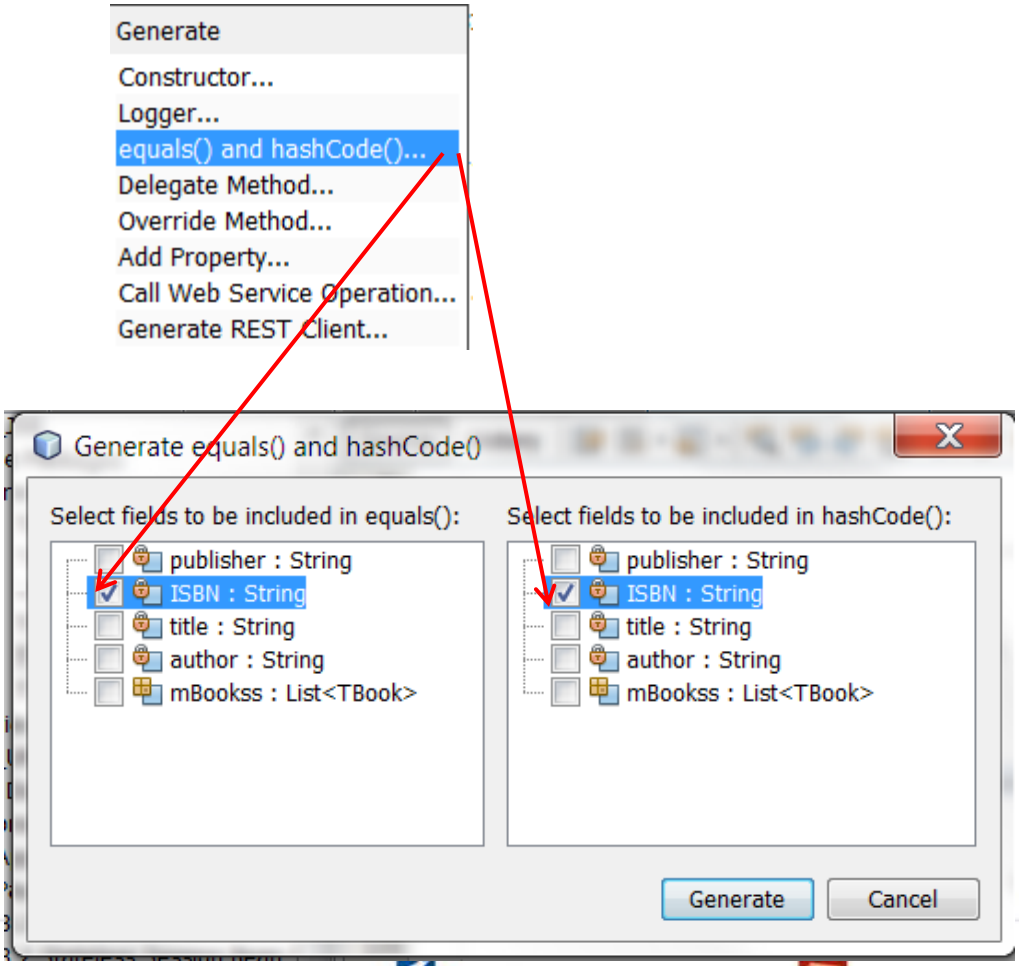
5.8.2. Select **Insert code** option from pop-up menu (generating Setter and Getter methods replaced with created from Insert code option) –right click the code editor, select the InsertCode…/Getter and Setter… items and select all atributies of TTitle_book in getter and setter dialog.



5.8.3. Insert code option from pop-up menu (generating equals and hashCode methods replaced with created from Insert code option) - right click the code editor, select the InsertCode… item



5.8.4. Insert code option from pop-up menu (generating equals and hashCode methods replaced with created from Insert code option) – select equals() and hashCode()… item and select the ISBN atribute of TTitle_book in their dialogs.

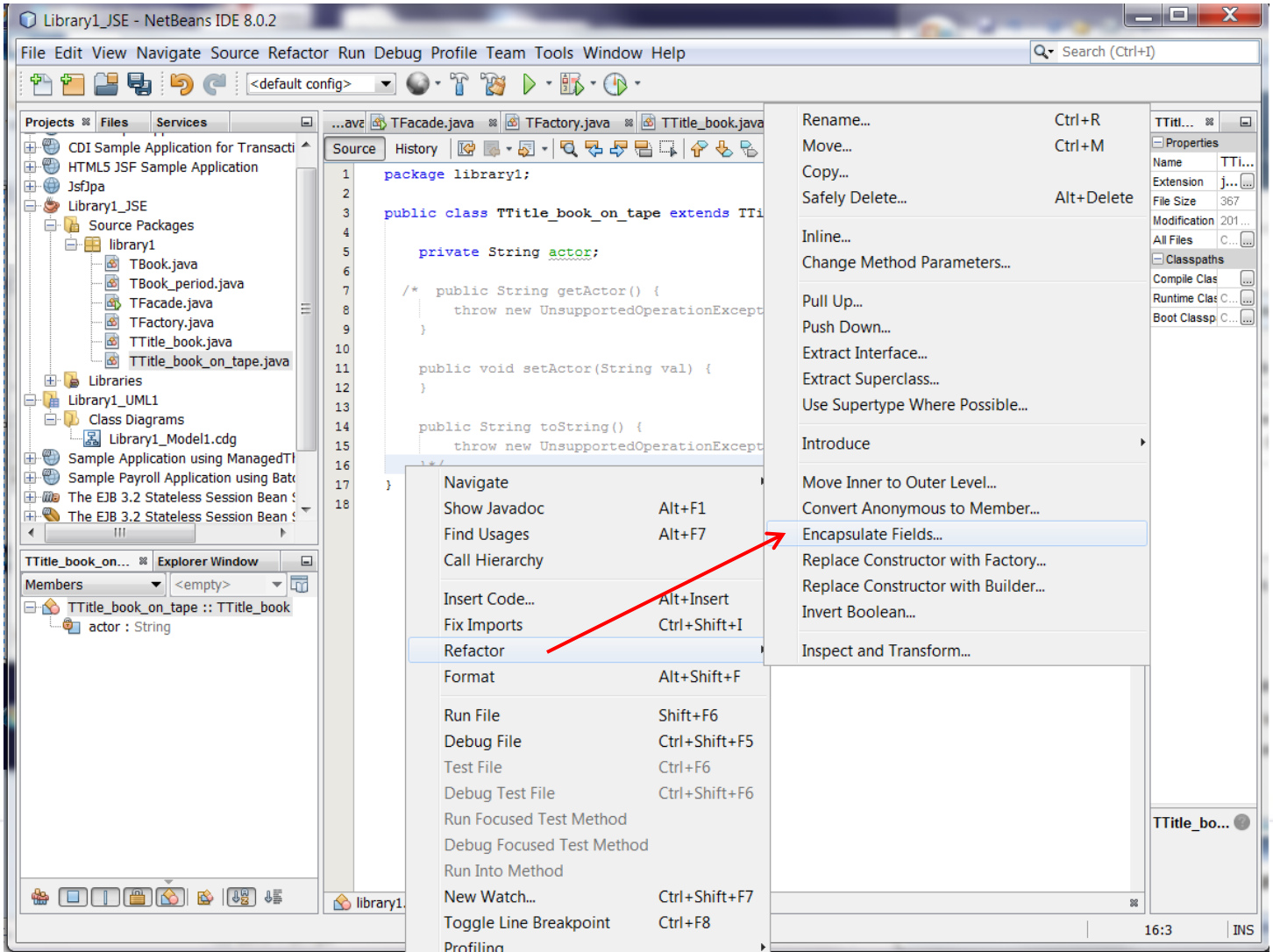



```
@Override
  public int hashCode() {
      int hash = 7;
      hash = 83 * hash + Objects.hashCode(this.ISBN);
      return hash;
  }

  @Override
  public boolean equals(Object obj) {
      if (obj == null) {
          return false;
      }
      if (getClass() != obj.getClass()) {
          return false;
      }
      final TTitle_book other = (TTitle_book) obj;
      if (!Objects.equals(this.ISBN, other.ISBN)) {
          return false;
      }
      return true;
  }
```

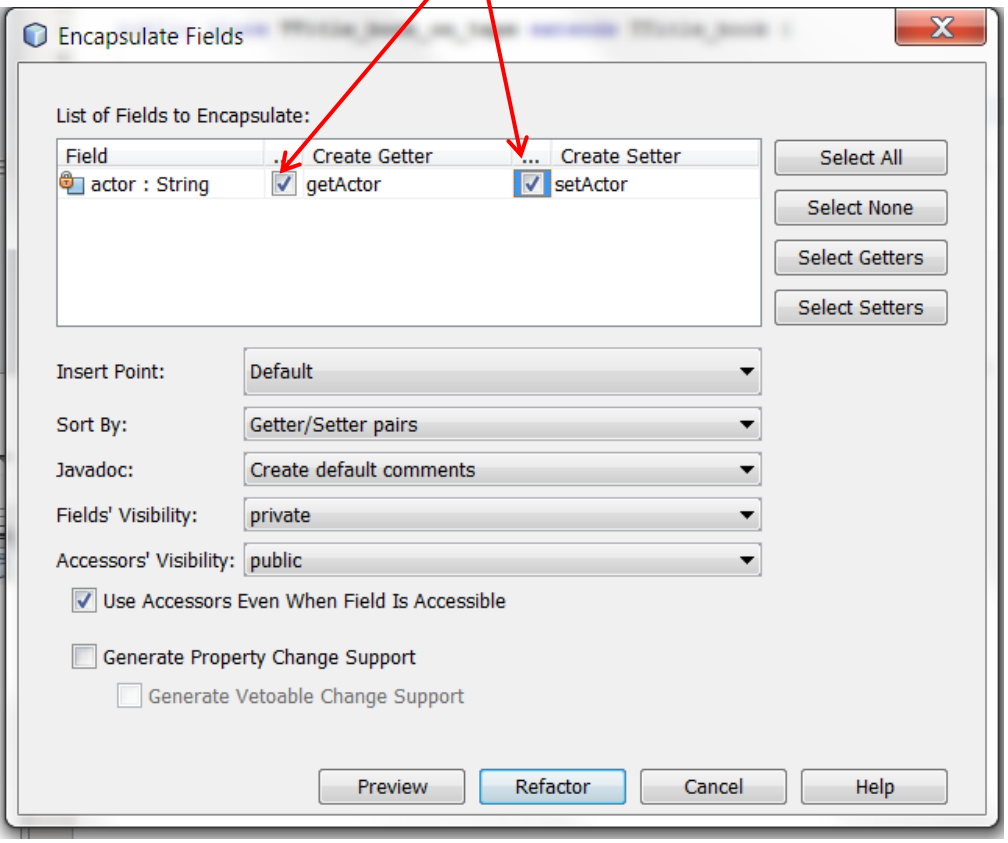
5.9. /5.9.1. Code of TTitle_book_on_tape – you must create the real getter and setter methods

```
package library1;


public class TTitle_book_on_tape extends TTitle_book {

    private String actor;

   /* public String getActor ()          { throw new UnsupportedOperationException("Not supported yet.");    }
    public void setActor (String val)     {    } */

    public String toString ()             { throw new UnsupportedOperationException("Not supported yet.");    }
}
```

5.9.2. You may create the code of getter and setter methods by using Refactor/Encapsulate Fields… options – right click the code editor, select the Refactor/Encapsulate Fields… items



5.9.3. You may create the code of getter and setter methods by using Refactor/Encapsulate Fields… options – select the pair of methods



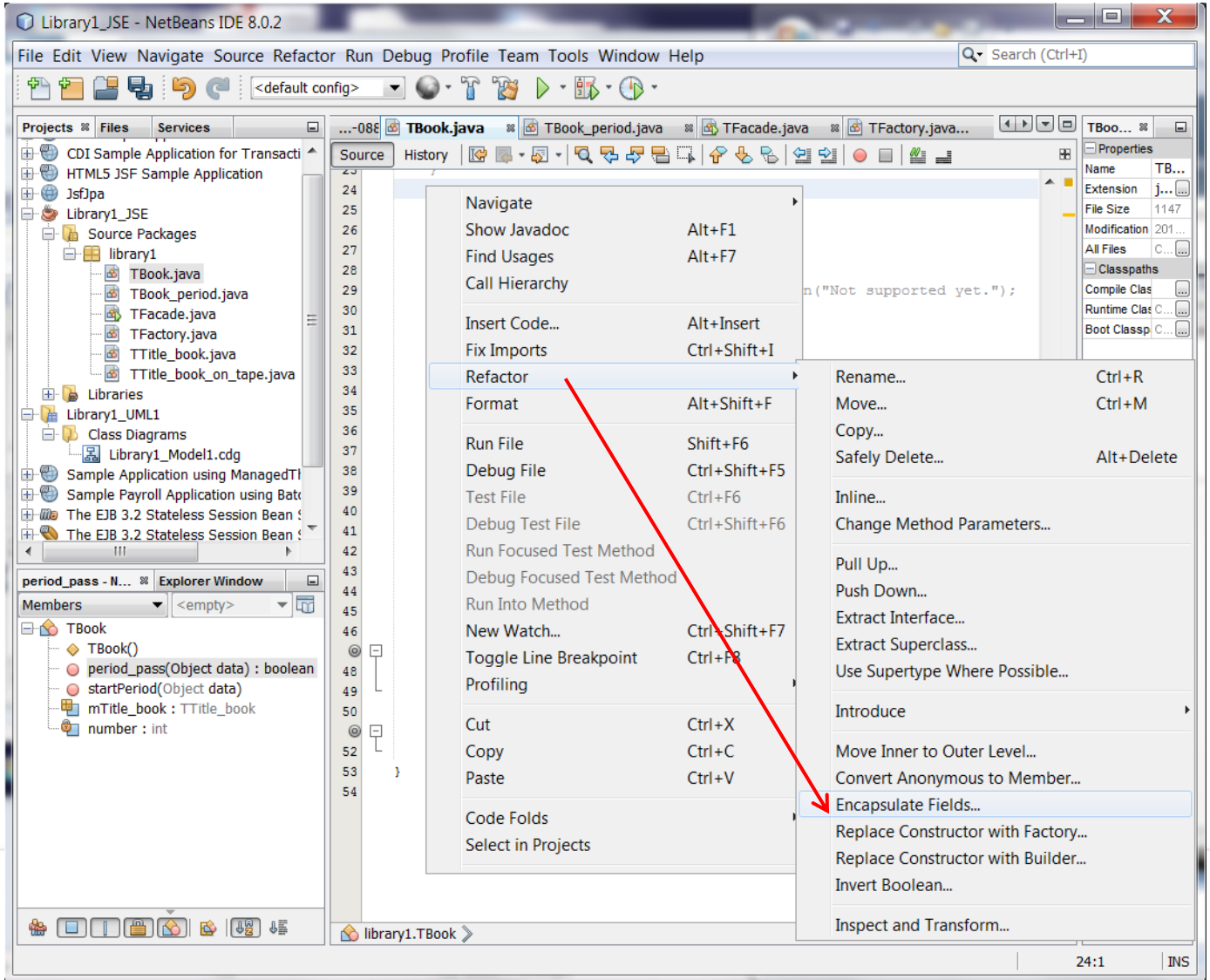
5.9.4. Code of TTitle_book_on_tape - after changing code of setter and getter methods

```
package library1;

public class TTitle_book_on_tape extends TTitle_book
{
  private String actor;
   public String getActor ()
        { return actor;  }
   public void setActor (String val)
        { this.actor = actor;    }


  public String toString ()
    { throw new UnsupportedOperationException("Not
       supported yet."); }
}
```

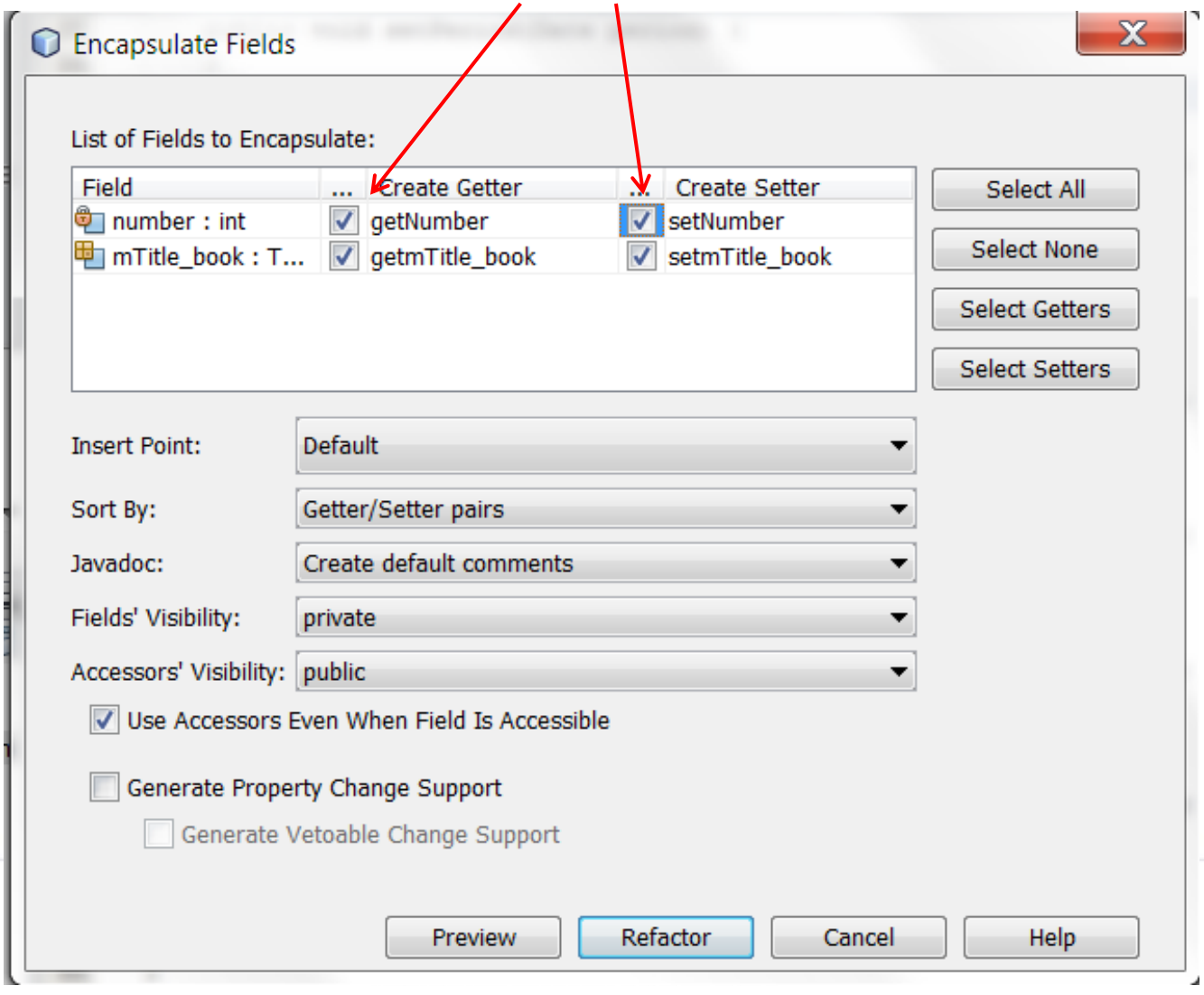
5.10. / 5.10.1. Code of TBook - before changing code of setter and getter methods

```
package library1;

import java.util.Date;

public class TBook  {

   private int number;
   TTitle_book mTitle_book;

   public TBook ()                                  {   }

  /* public TTitle_book getmTitle_book ()           { throw new UnsupportedOperationException("Not supported yet."); }
   public void setmTitle_book (TTitle_book title_book)  {   }
   public int getNumber ()                          { throw new UnsupportedOperationException("Not supported yet."); }
   public void setNumber (int val)                  {   } */

   public String toString ()                        { throw new UnsupportedOperationException("Not supported yet.");}
   public int hashCode ()                           { throw new UnsupportedOperationException("Not supported yet.");  }
   public boolean equals (Object obj)               { throw new UnsupportedOperationException("Not supported yet.");}

   public Date getPeriod ()                         { throw new UnsupportedOperationException("Not supported yet."); }
   public void setPeriod (Date period)              {   }

   public boolean period_pass(Object data)          { throw new UnsupportedOperationException("Not supported yet.");}
   public void startPeriod(Object data)             {   }
}
```

5.10.2. Select Refactor/Encapsulation Fields… options from pop-up menu (generating Setter and Getter methods replaced with created from Refactor/Encapsulation Fields… options)



5.10.3. Refactor>Encapsulation Method> option from pop-up menu (generating Setter and Getter methods replaced with created from Refactor/Encapsulation Fields… options)



5.11. /5.11.1. Code of TBook - after changing code of setter and getter methods

```
package library1;

import java.util.Date;

public class TBook  {

   private int number;
   private TTitle_book mTitle_book;

   public TBook ()                                          {   }

  public TTitle_book getmTitle_book ()                      { return mTitle_book; }
   public void setmTitle_book (TTitle_book mTitle_book)     { this.mTitle_book = mTitle_book;  }

   public int getNumber ()                                  { return number; }
   public void setNumber (int val)                          { this.number = number; }

   public String toString ()                                { throw new UnsupportedOperationException("Not supported yet.");}
   public int hashCode ()                                    { throw new UnsupportedOperationException("Not supported yet.");  }
   public boolean equals (Object obj)                       { throw new UnsupportedOperationException("Not supported yet.");}

   public Date getPeriod ()                                 { throw new UnsupportedOperationException("Not supported yet."); }
   public void setPeriod (Date period)                      {   }

   public boolean period_pass(Object data)                  { throw new UnsupportedOperationException("Not supported yet.");}
   public void startPeriod(Object data)                     {   }
}
```

5.12/ 5.12.1. Code of TBook_period – before changing code of setter and getter methods

```
package library1;

import java.util.Date;

public class TBook_period extends TBook {
    private Date period;
   /*public Date getPeriod ()                    { throw new UnsupportedOperationException("Not supported yet.");    }
   public void setPeriod (Date date)             {   } */

   public String toString ()                     { throw new UnsupportedOperationException("Not supported yet."); }
   public boolean period_pass(Object data)       { throw new UnsupportedOperationException("Not supported yet.");}
   public void startPeriod(Object data)          {   }
}
```

5.12.2. Code of TBook_period - after changing code of setter and getter methods

```
package library1;

import java.util.Date;

   public class TBook_period extends TBook {
    private Date period;
   public Date getPeriod ()                      {  return period;  }
   public void setPeriod (Date date)             {   this.period = period; }
   public String toString ()                     { throw new UnsupportedOperationException("Not supported yet."); }
   public boolean period_pass(Object data)       { throw new UnsupportedOperationException("Not supported yet.");}
   public void startPeriod(Object data)          {   }
}
```

5.13./5.13.1. Code of TFacade – before changing code of setter and getter methods

```
package library1;

import java.util.ArrayList;

public class TFacade {

   List<TTitle_book> mTitle_books;

   public TFacade ()                                                    { mTitle_books = new ArrayList<>();  }
 /*  public  ArrayList<TTitle_book> getmTitle_books ()                  {throw new UnsupportedOperationException("Not supported yet.");   }
   public  void setmTitle_books (ArrayList<TTitle_book> title_books)                    {   }*/

   public  TTitle_book search_title_book (TTitle_book title_book)   {throw new UnsupportedOperationException("Not supported yet.");  }
   public  String add_title_book (String[] data)                    {throw new UnsupportedOperationException("Not supported yet.");  }
   public  ArrayList<String> add_book (String[] data1, String[] data2) {throw new UnsupportedOperationException("Not supported yet.");  }
   public  ArrayList<String> Search_title_book (String[] data)      {throw new UnsupportedOperationException("Not supported yet.");  }
   public  String Search_book (String[] data1, String[] data2)      {throw new UnsupportedOperationException("Not supported yet.");  }
   public  String Search_accessible_book(String data1[], Object data2)
                                                                    {throw new UnsupportedOperationException("Not supported yet.");  }

   public  Object[][] gettitle_books()              {throw new UnsupportedOperationException("Not supported yet.");   }
   public  void Print_books ()                                                  {   }
   public  void Print_title_books ()                                            {   }
   public static void main (String[] t)                                        {   }
}
```

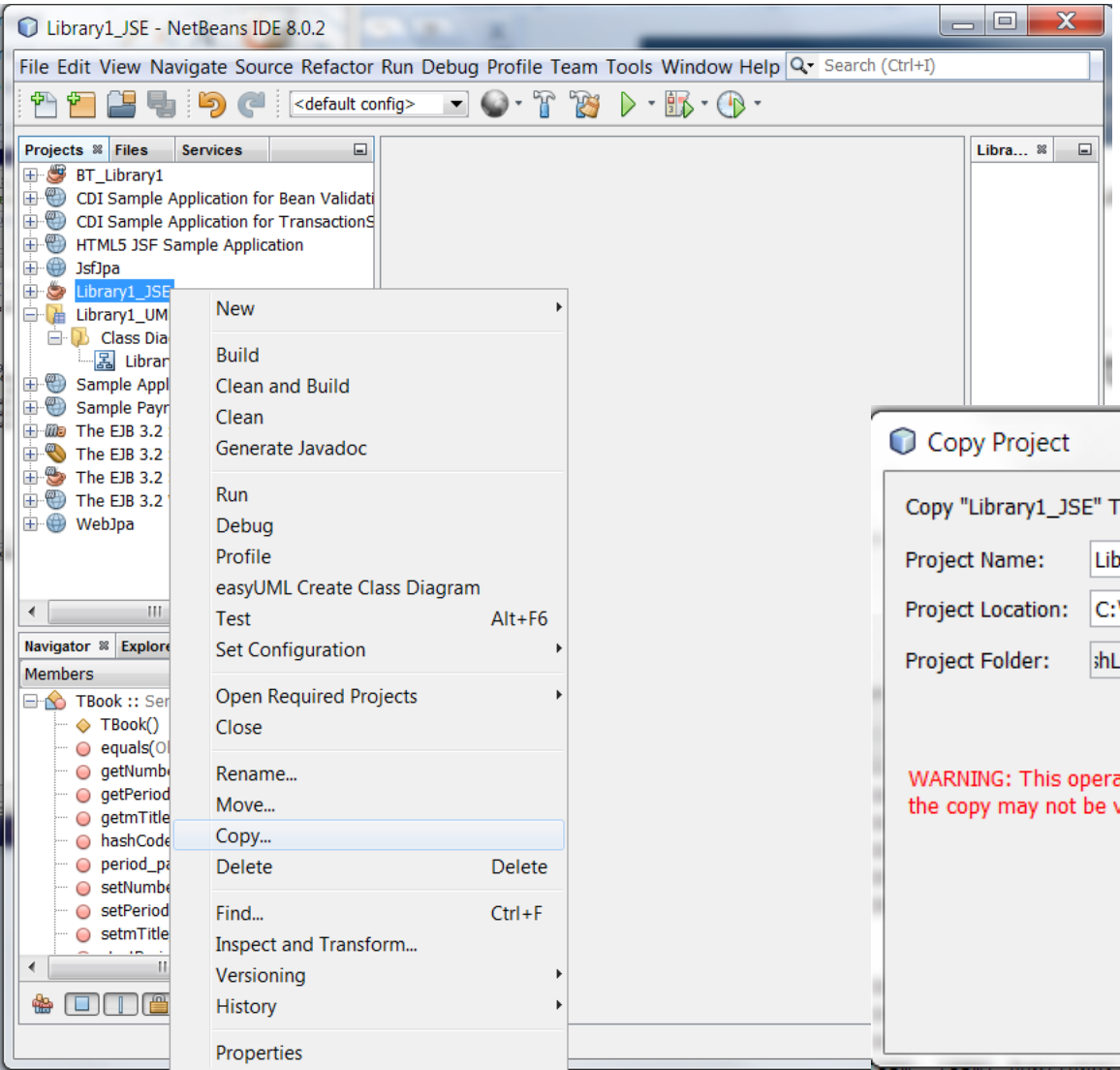
5.13.2. Code of TFacade - after changing code of setter and getter methods

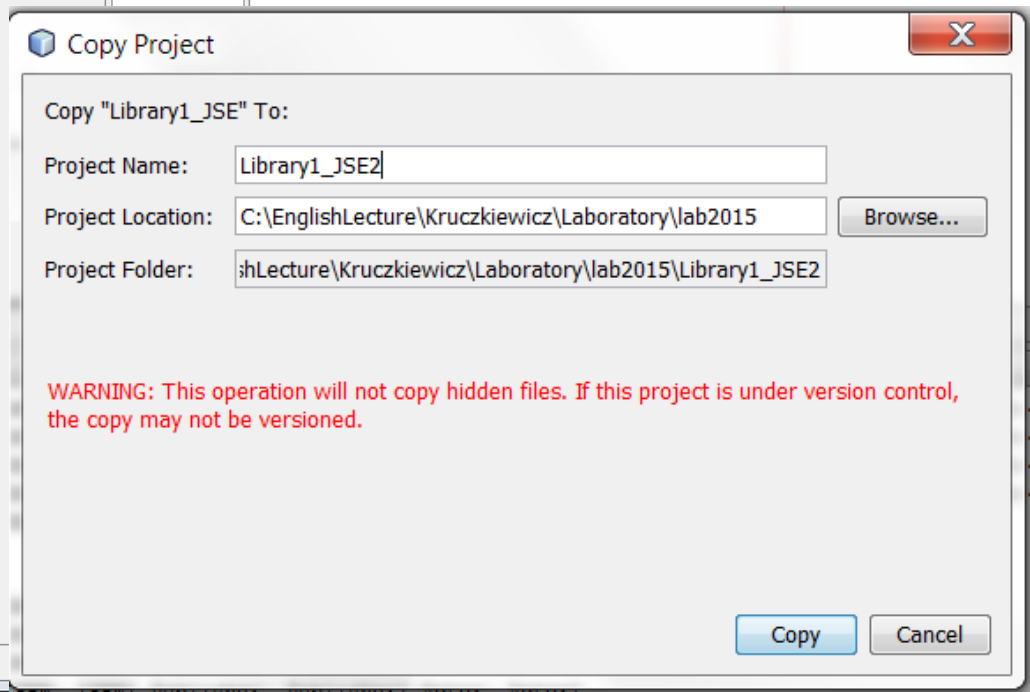
```
package library1;

import java.util.ArrayList;

public class TFacade {

   List<TTitle_book> mTitle_books ;

   public TFacade ()                                                    {      mTitle_books = new ArrayList<>();  }
   public List<TTitle_book> getmTitle_books()                           {      return mTitle_books;   }
   public void setmTitle_bookss(List<TTitle_book> mTitle_books)         {      this.mTitle_books = mTitle_books;   }

   public  TTitle_book search_title_book (TTitle_book title_book)   {throw new UnsupportedOperationException("Not supported yet.");  }
   public  String add_title_book (String[] data)                    {throw new UnsupportedOperationException("Not supported yet.");  }
   public ArrayList<String> add_book (String[] data1, String[] data2) {throw new UnsupportedOperationException("Not supported yet.");  }
   public ArrayList<String>  Search_title_book (String[] data)      {throw new UnsupportedOperationException("Not supported yet.");  }
   public  String Search_book (String[] data1, String[] data2)      {throw new UnsupportedOperationException("Not supported yet.");  }
   public  String Search_accessible_book(String data1[], Object data2)
                                                                    {throw new UnsupportedOperationException("Not supported yet.");  }

   public  Object[][] gettitle_books()                              {throw new UnsupportedOperationException("Not supported yet.");   }
   public  void Print_books ()                                                 {  }
   public  void Print_title_books ()                                           {  }
   public static void main (String[] t)                                       {  }
}
```

5.14. Code of TFactory class

```
package library1;

public class TFactory {

   static final long day=24*60*60*1000;
   static public Date mdays(String data)                { throw new UnsupportedOperationException("Not supported yet.");      }

   public TTitle_book create_title_book (String[] data)  { throw new UnsupportedOperationException("Not supported yet."); }
   public TBook create_book (String[] data)              { throw new UnsupportedOperationException("Not supported yet."); }


}
```

5.15. Preparing the copy of the **Library1_JSE** project as the **Library1_JSE2** project





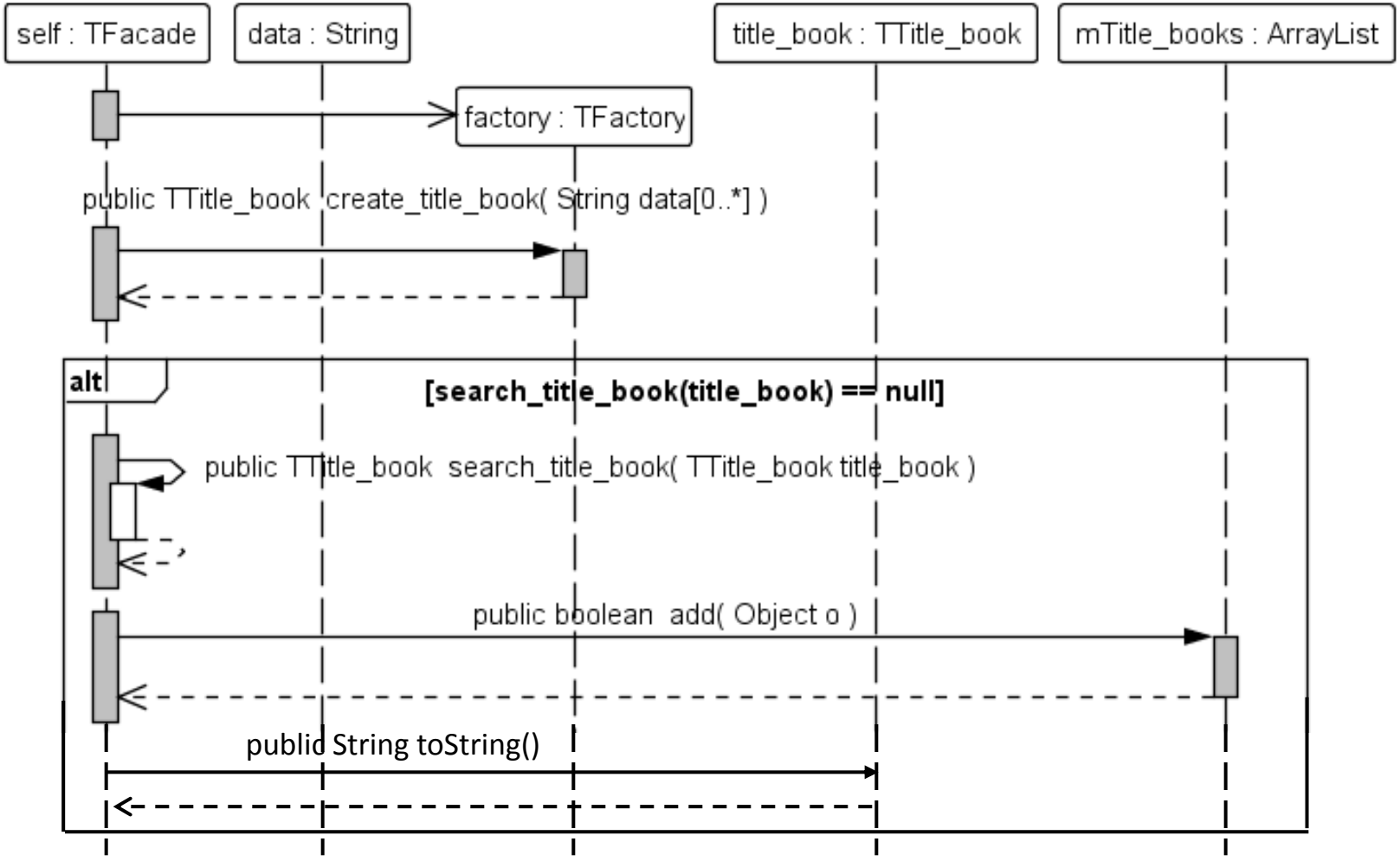
6./6.1. First Iteration - obligatory

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.

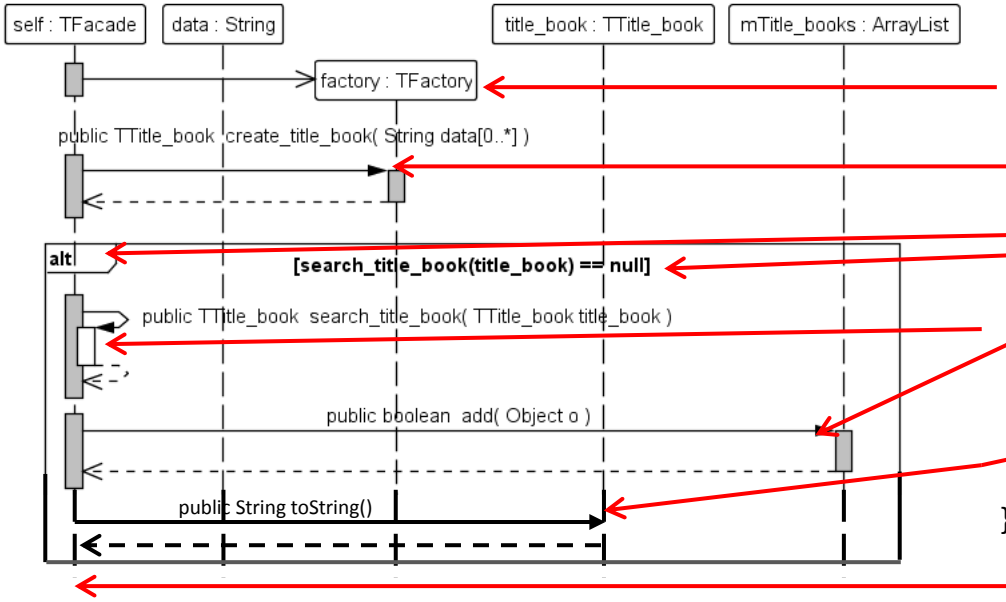
Applying Facade and Factory design patterns

6.1.1. UC_TFacade_add_title_book

TFacade: public String add_title_book(String data[])

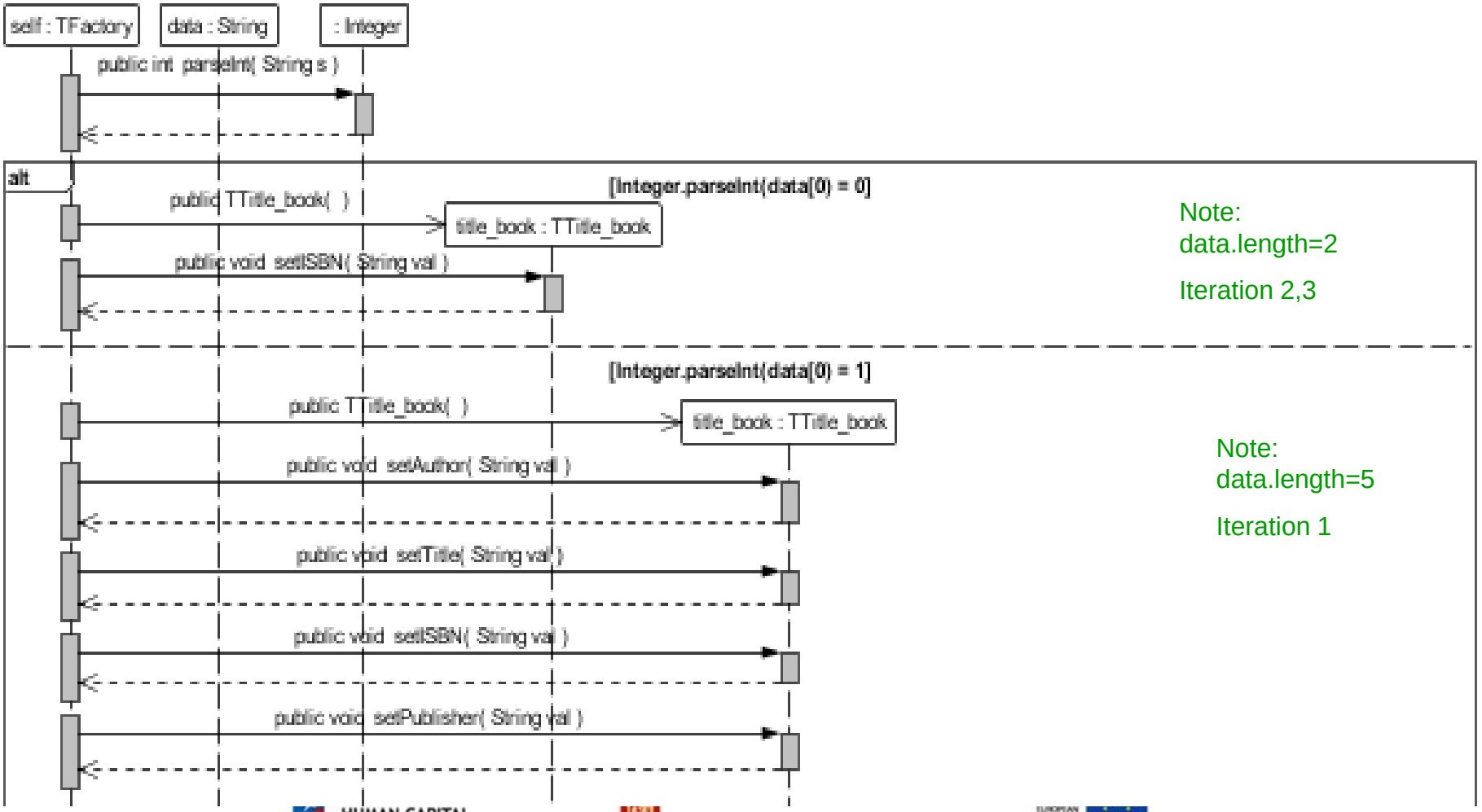


6.1.1. Continuation

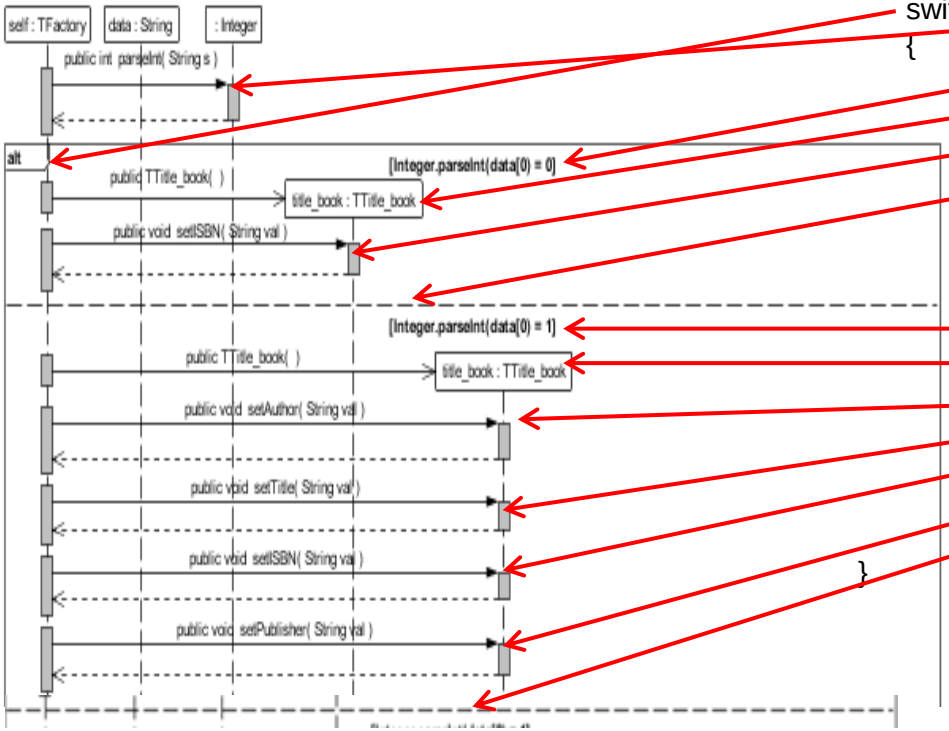



```
public String add_title_book(String data[]) {
    TFactory factory = new TFactory();
    TTitle_book title_book = factory.create_title_book(data);
    if (search_title_book(title_book) == null) {
        mTitle_books.add(title_book);
        return title_book.toString();
    }
    return null;
}
```

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

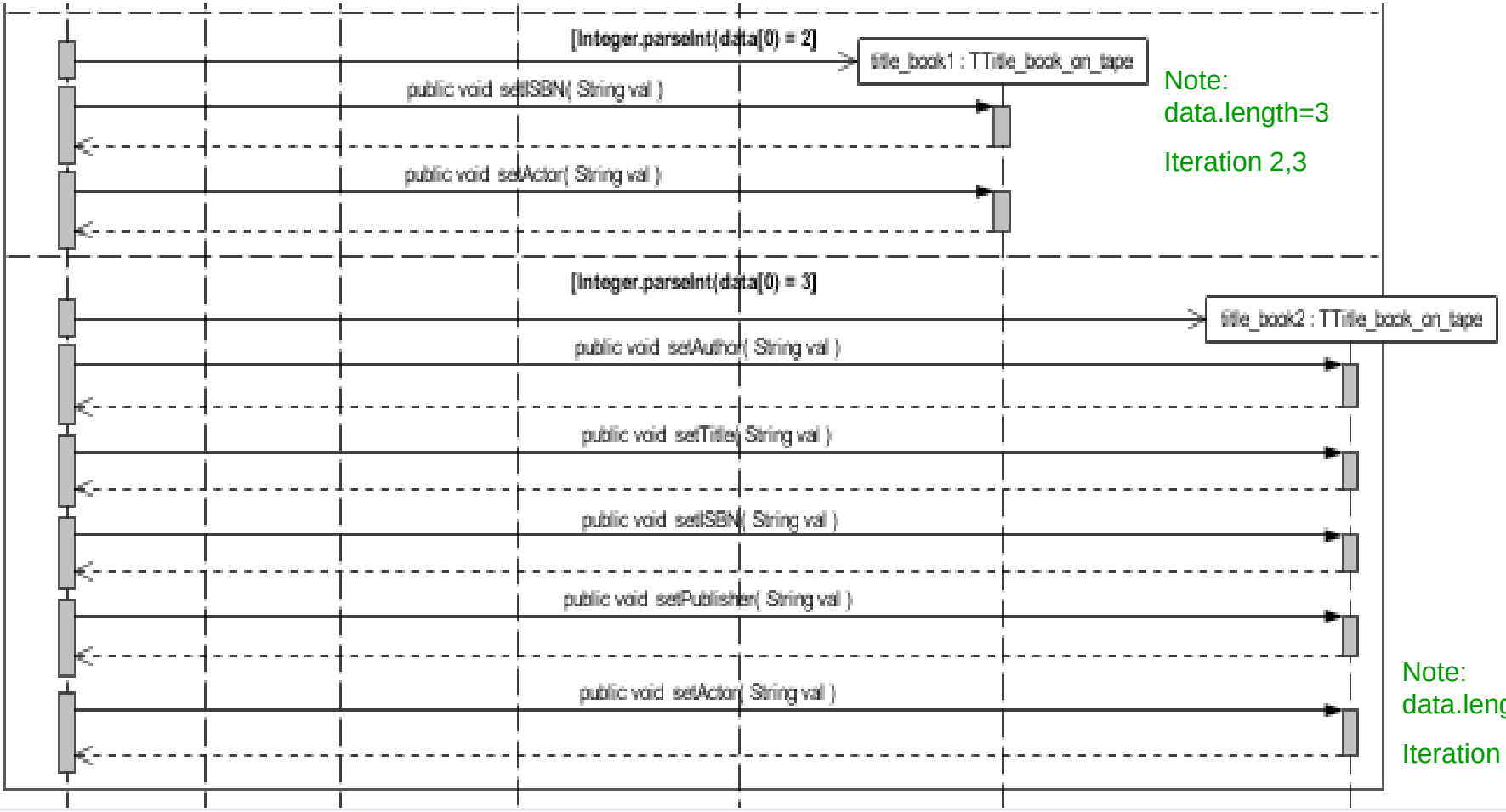


6.1.2. – Continuation




```
public TTitle_book create_title_book(String data[]) {
    TTitle_book title_book=null;
    switch (Integer.parseInt(data[0])) //what_title_book_type
    {
    case 0:
      title_book = new  TTitle_book(); //TTitle_book object for searching
      title_book.setISBN(data[1]);
      break;

    case 1:
      title_book = new  TTitle_book(); //TTitle_book object for persisting
      title_book.setAuthor(data[1]);
      title_book.setTitle(data[2]);
      title_book.setISBN(data[3]);
      title_book.setPublisher(data[4]);
      break;
    }
```

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



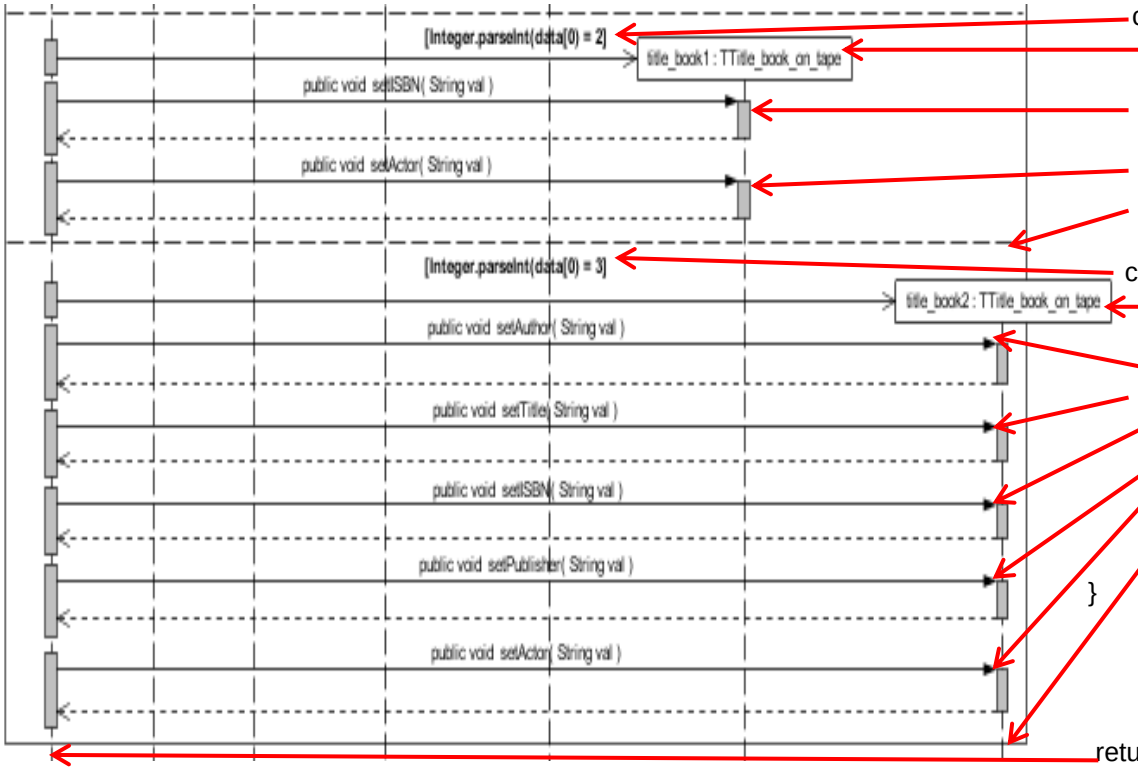
Note:
data.length=3

Iteration 2,3

Note:
data.length=6

Iteration 1

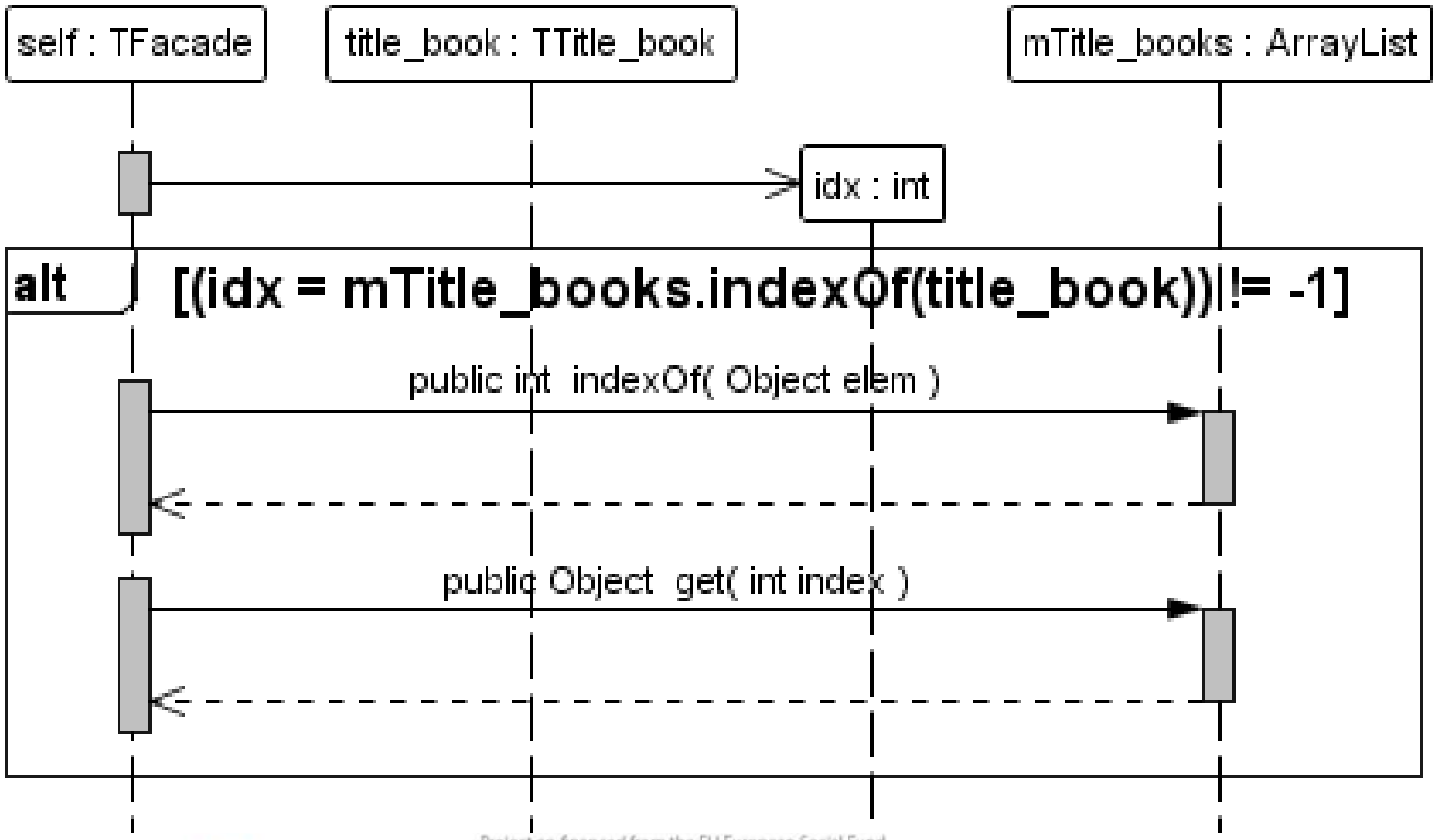
6.1.2. – Continuation



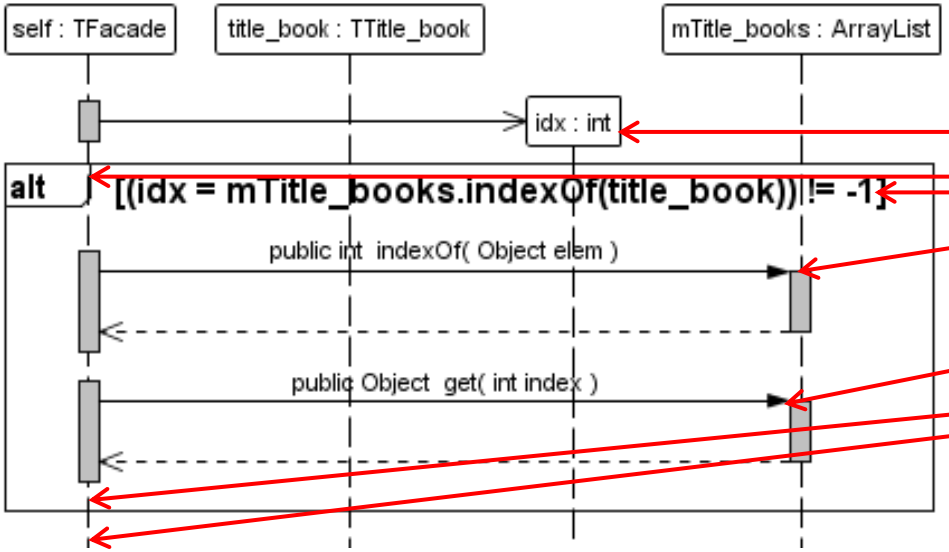

```
case 2:
 TTitle_book_on_tape title_book1= new TTitle_book_on_tape();
          //TTitle_book_on_tape object for searching
title_book1.setISBN(data[1]);
title_book1.setActor(data[2]);
title_book=title_book1;
break;

case 3:
TTitle_book_on_tape title_book2= new TTitle_book_on_tape();
          //TTitle_book_on_tape object for persisting
title_book2.setAuthor(data[1]);
title_book2.setTitle(data[2]);
title_book2.setISBN(data[3]);
title_book2.setPublisher(data[4]);
title_book2.setActor(data[5]);
title_book=title_book2;
break;
}

return title_book;
```

6.1.3. UC_TFacade_search_title_book

TFacade: public TTitle_book search_title_book(TTitle_book title_book)

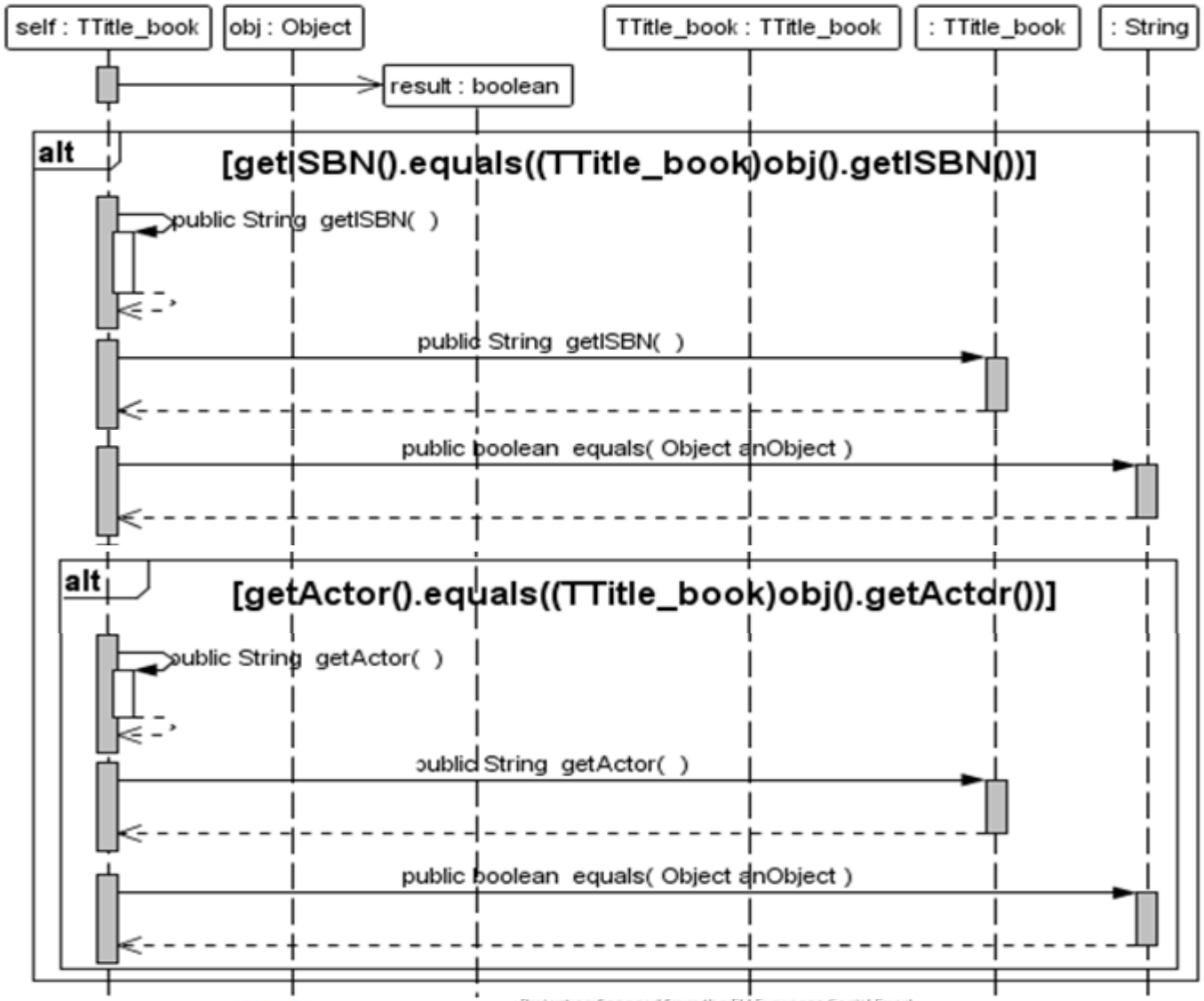


6.1.3. Continuation

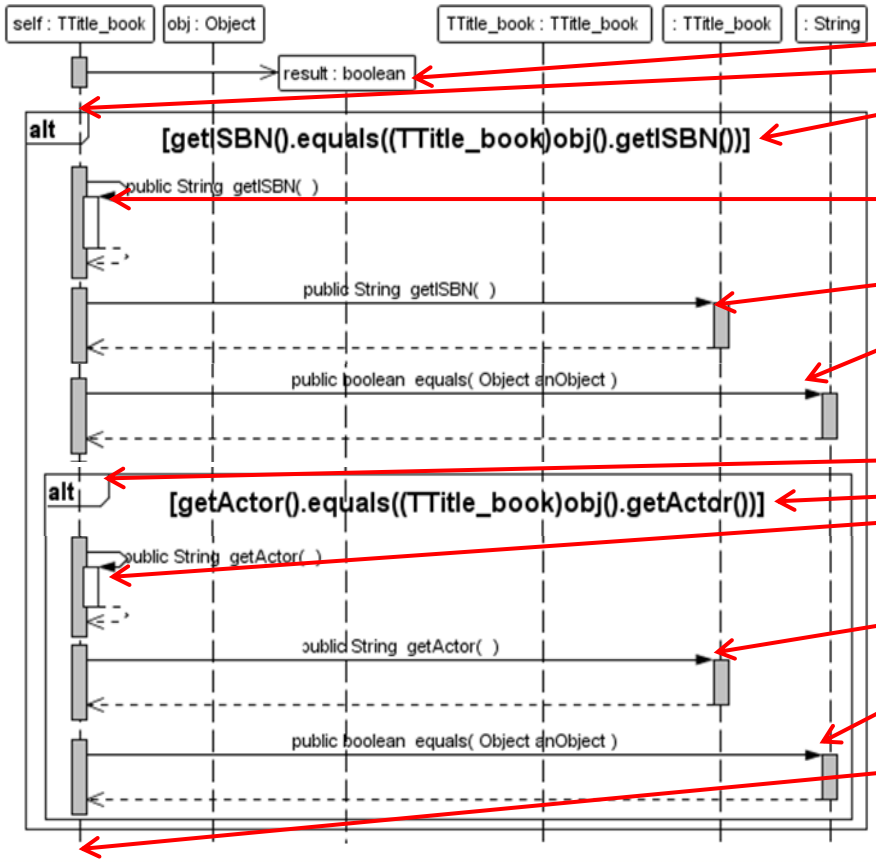



```
public TTitle_book search_title_book(TTitle_book title_book) {
    int idx;
    if ((idx = mTitle_books.indexOf(title_book)) != -1) {
        title_book = mTitle_books.get(idx);
        return title_book;  }
    return null;
    }
```

6.1.4. TTitle_book_equals - TTitle_book: public boolean equals(Object obj); **getActor** method is a virtual one



6.1.4. Continuation




```
@Override
  public boolean equals(Object obj) {
    boolean result = false;
    if (getISBN().equals(((TTitle_book) obj).getISBN()))
    {
      if (getActor().equals(((TTitle_book) obj).getActor())) {
        result = true;
      }
    }
    return result;
  }
```

6.1.5. Update of the getActor() virtual method and definition of the toString() method of the TTitle_book class

```
public String getActor() {
      // throw new UnsupportedOperationException("Not supported yet.");
       return "";
  }
@Override
public String toString() {
      // throw new UnsupportedOperationException("Not supported yet.");
       String help = "\nTitle: " + getTitle();
       help += " Author: " + getAuthor();
       help += " ISBN: " + getISBN();
       help += " Publisher: " + getPublisher();
       return help;
  }
```

6.1.5. Definition of the toString() method of the TTitle_book_on_tape class

```
@Override
   public String toString() {
       //throw new UnsupportedOperationException("Not supported yet.");
       String help = super.toString();
       help += " Actor: " + getActor();
       return help;
   }
```

6.1.5. 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]
```

6.2. Second Iteration-obligatory

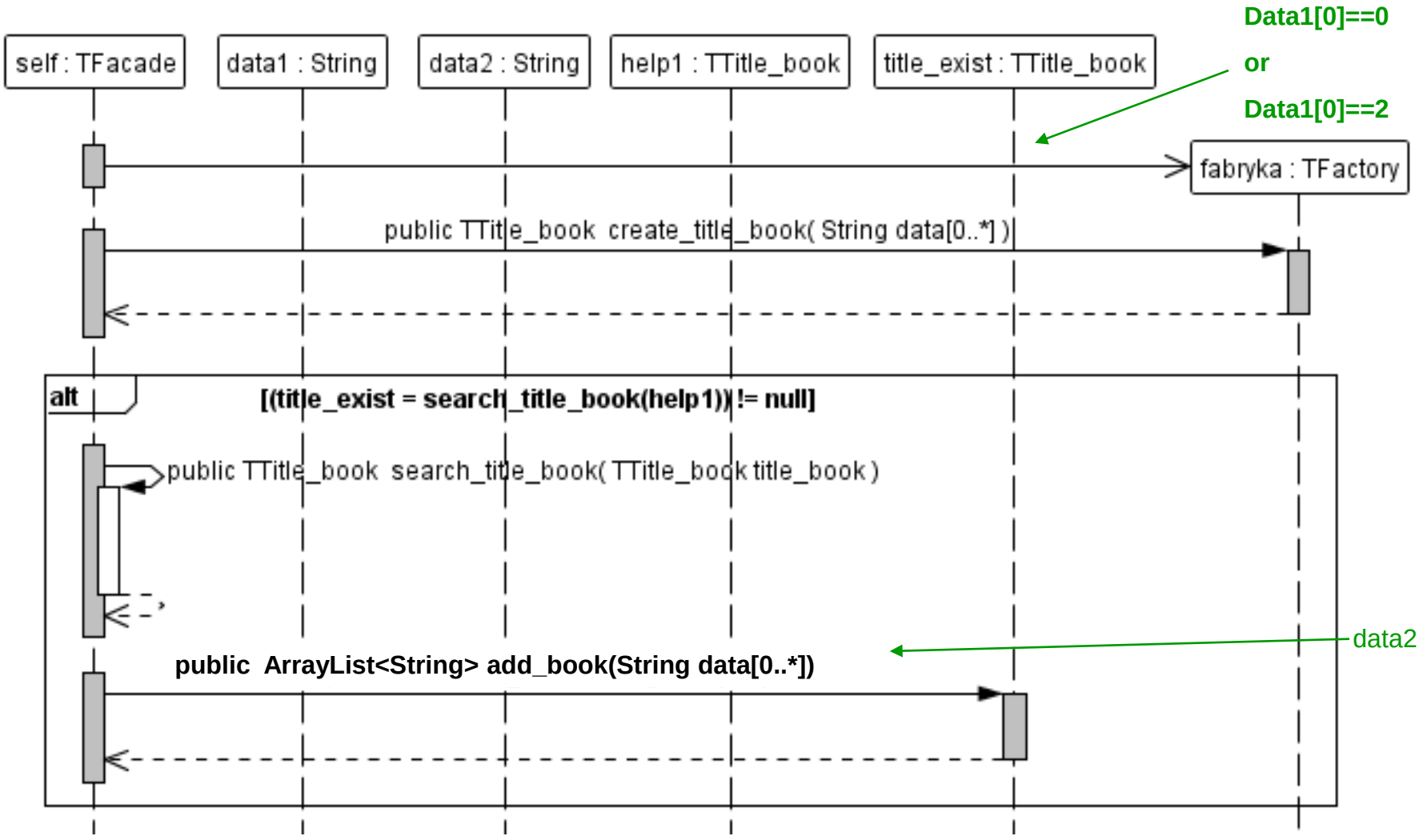
Create the code of methods based on the class diagram and sequence diagrams.

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

6.2.1. UC_TFacade_add_book

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

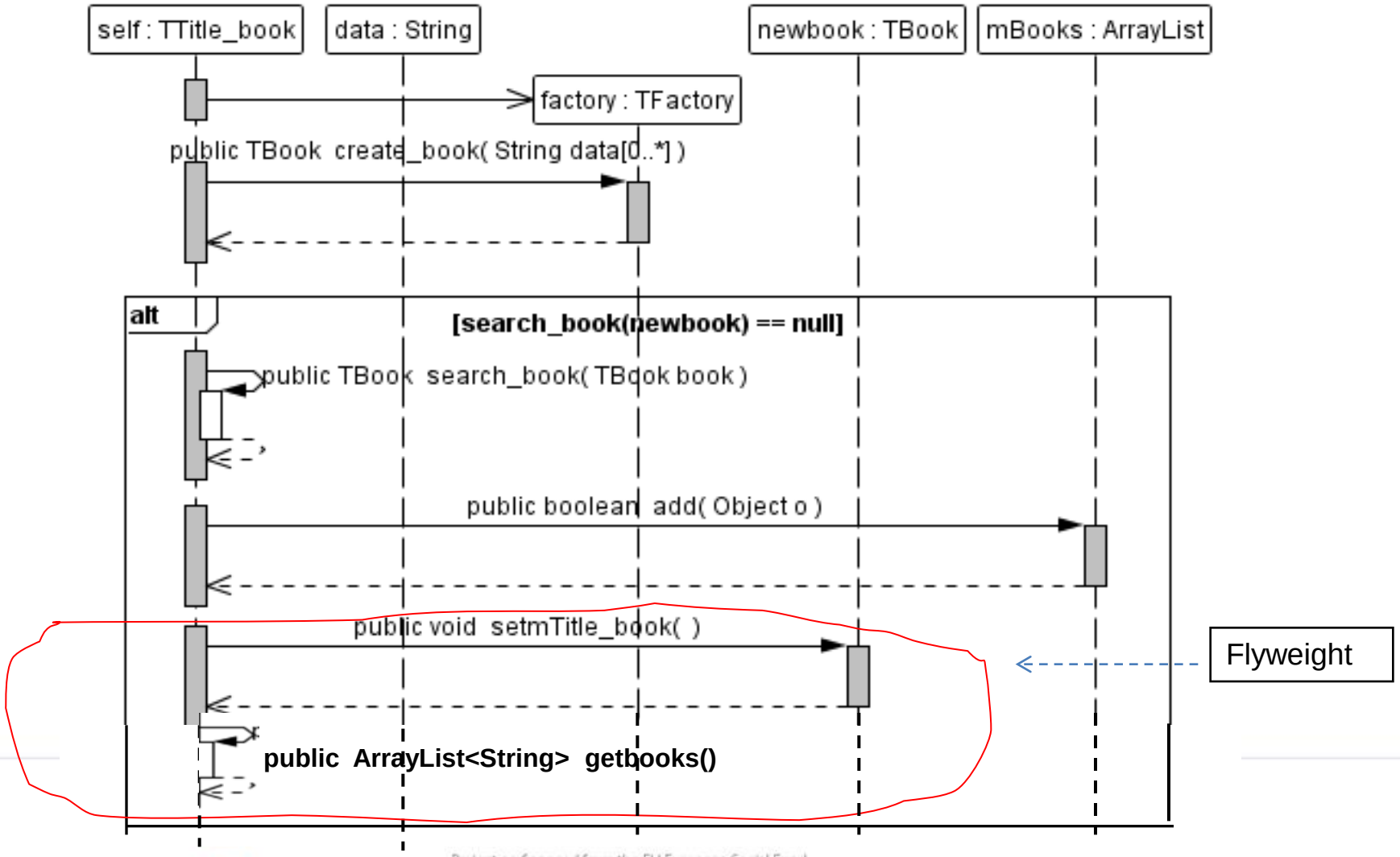


TTitle_book.add_book

TTitle_book: public void add_book(String data [])

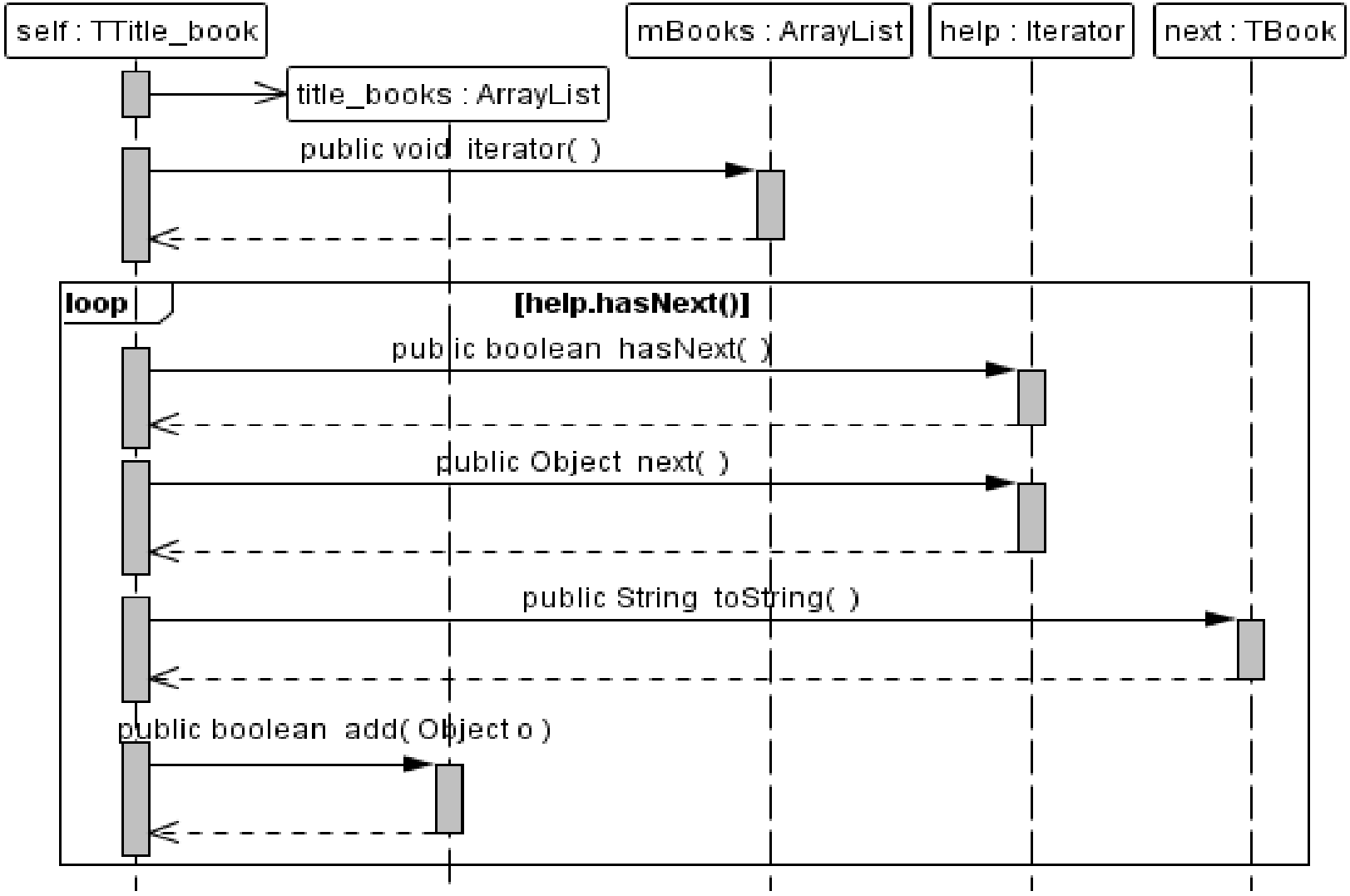
6.2.2. TTitle_book_add_book

Title_book: public ArrayList<String> add_book(String data[]) ()

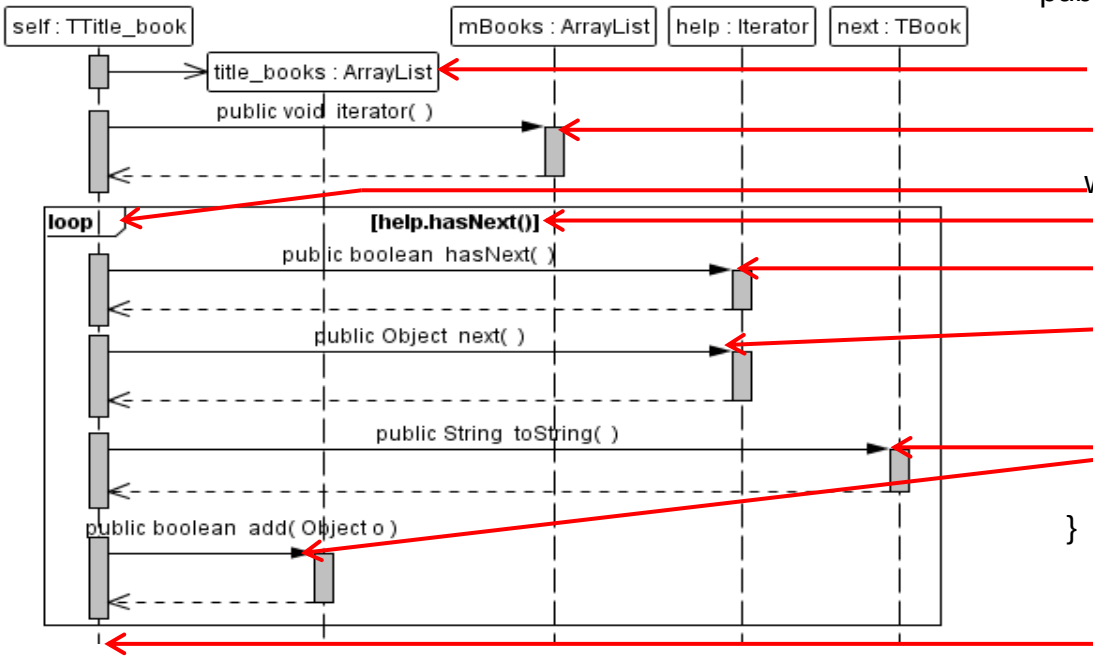


6.2.3. TTitle_book_getbooks

TTitle_book: public ArrayList<String> getbooks()



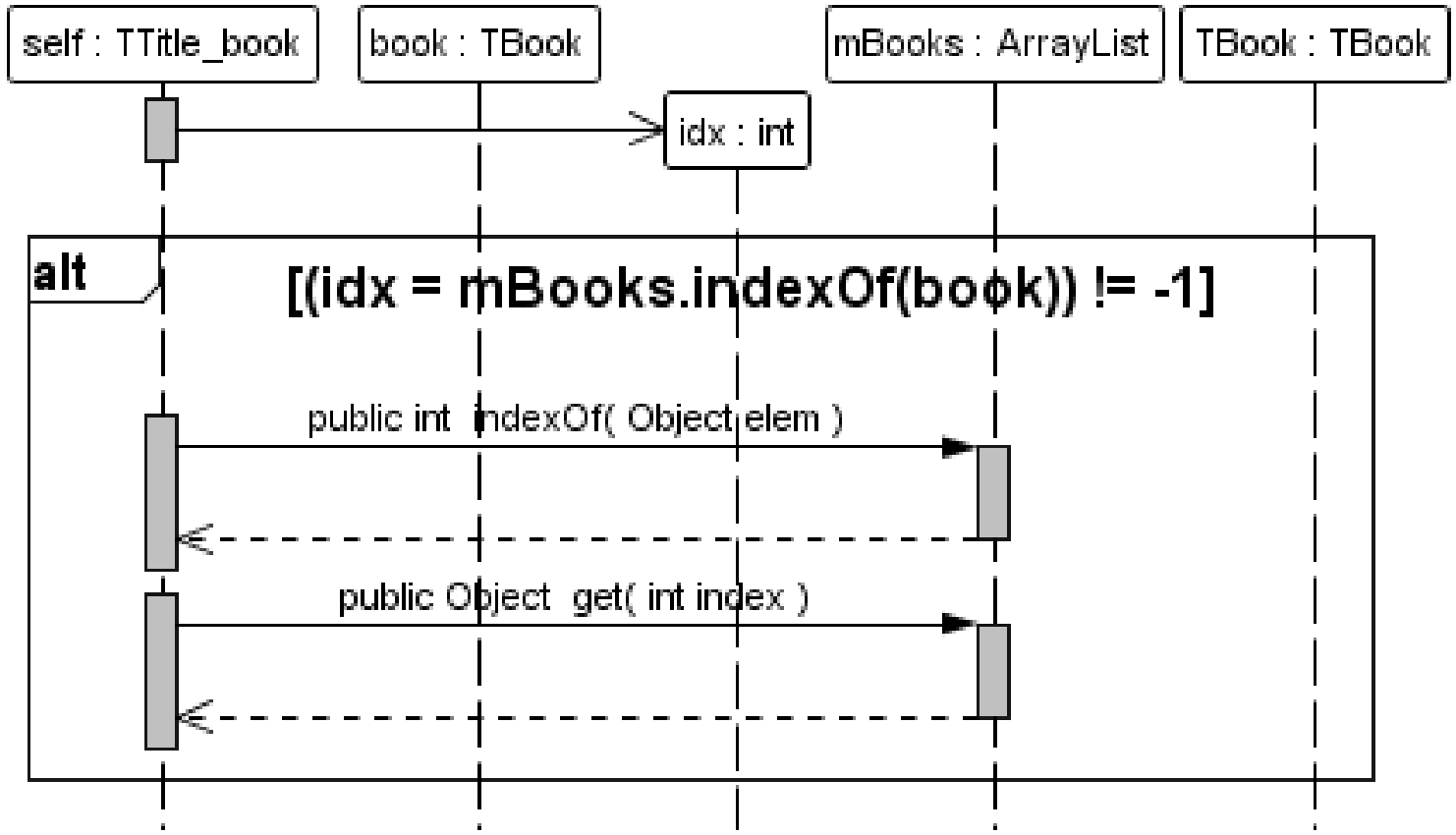
6.2.3. Continuation




```
public ArrayList<String> getbooks() {
    ArrayList<String> title_books = new ArrayList<>();
    Iterator<TBook> help = mBooks.iterator();
    while (help.hasNext()) {
        TBook next = help.next();
        title_books.add(next.toString());
    }
    return title_books;
}
```

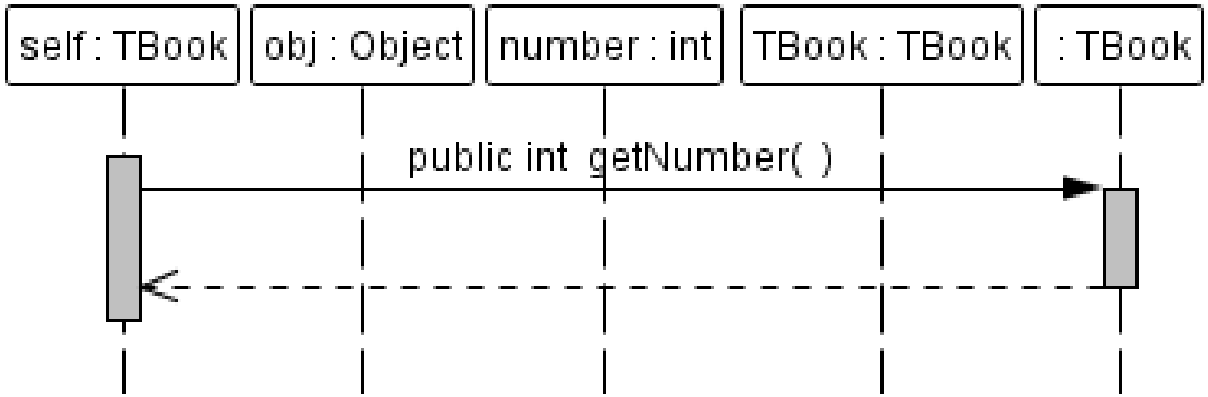
6.2.4. UC_TTitle_book_search_book

TTitle_book: public TBook search_book(TBook book)

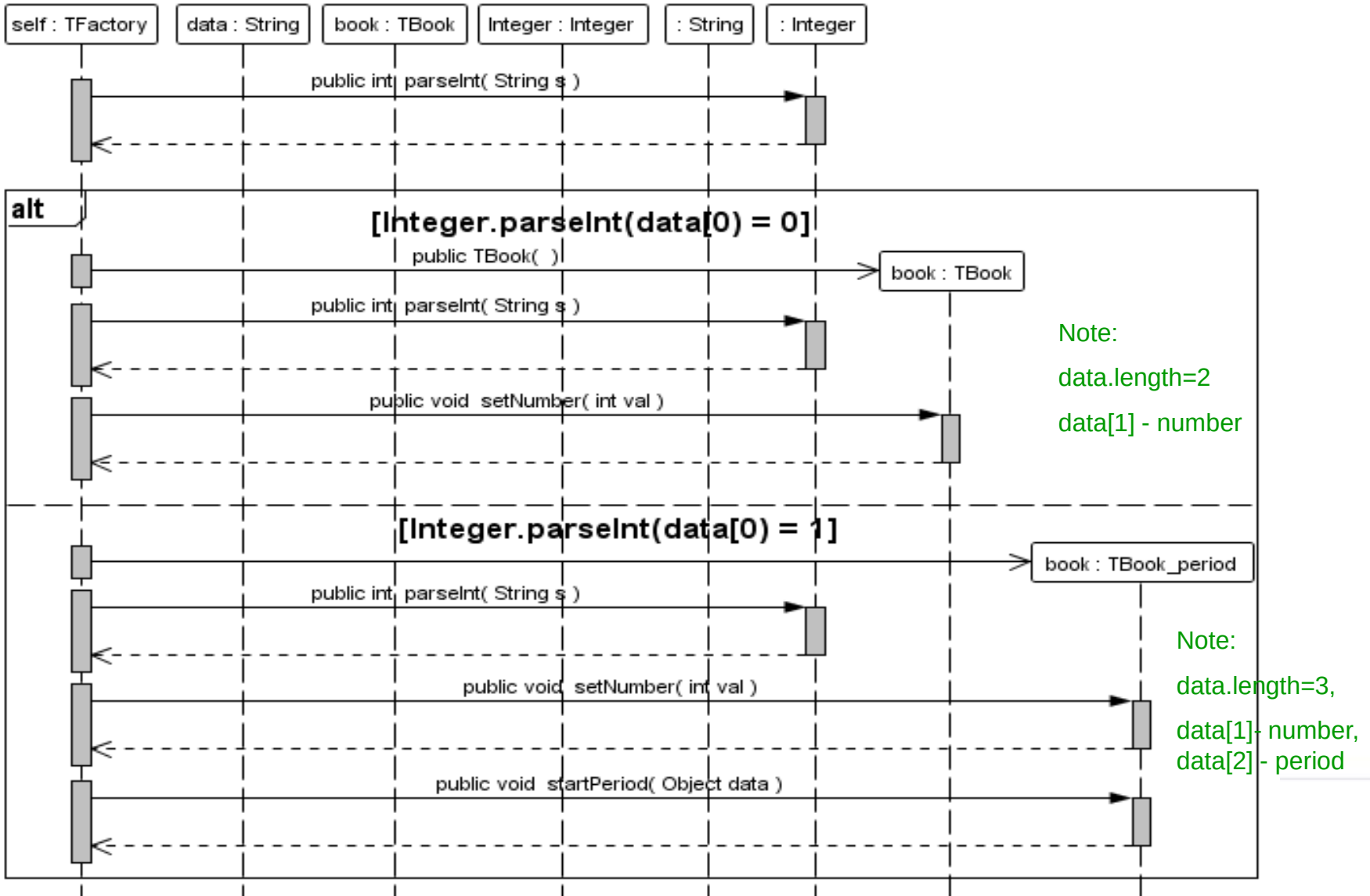


6.2.5. TBook_equals

TBook: public boolean equals(Object obj)

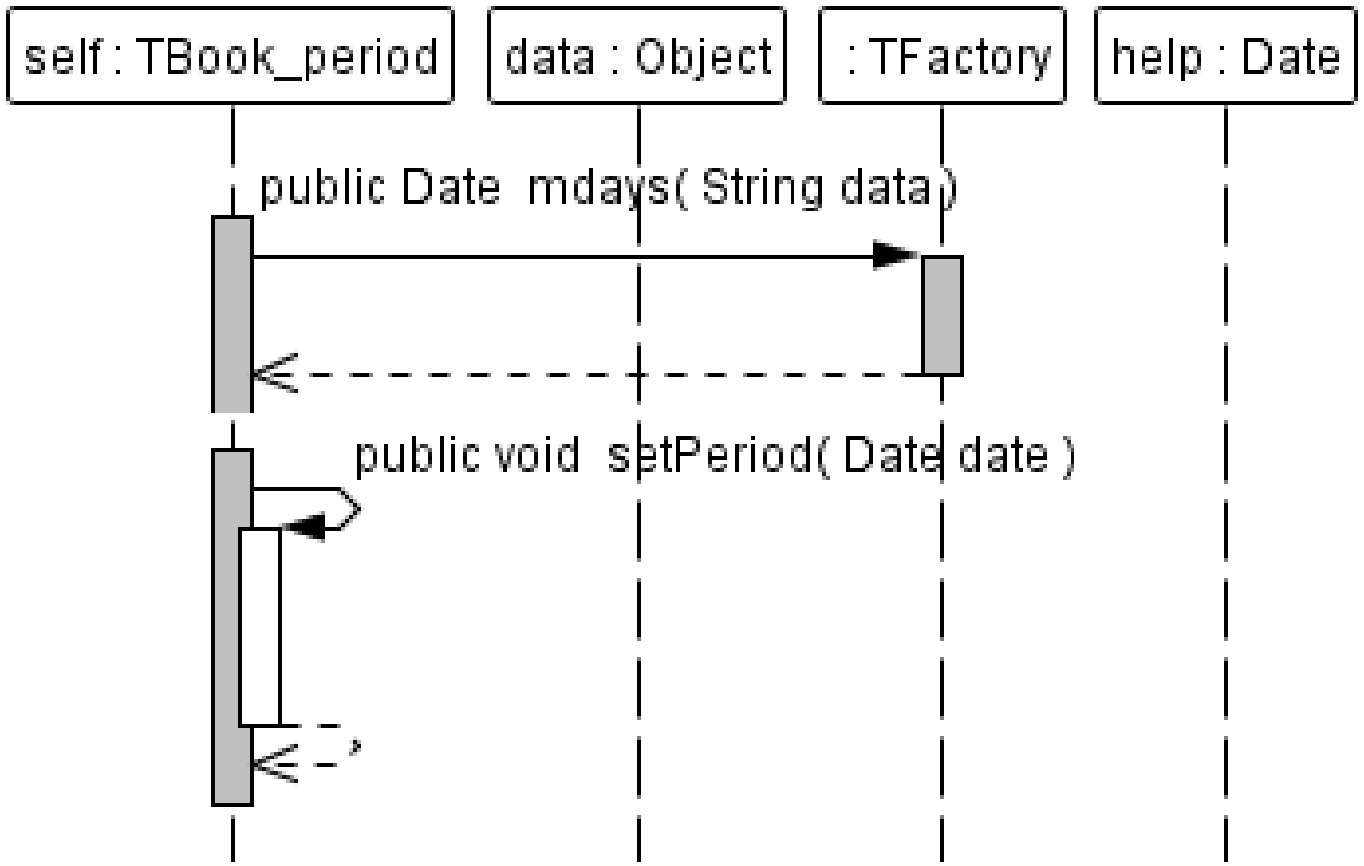


6.2.6. TFactory_create_book TFactory: public TBook create_book(String data[])



6.2.7. TBook_period_startPeriod

TBook_period: public void startPeriod(Object data)



6.2.8. The second iteration

package library1;6.2.7.

```
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

6.2.9. The second iteration

```
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"};
ArrayList<String> pom = ap.add_book(d1, tr1);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d2, tr1);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d2, tr1);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d2, tr2);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d3, tr2);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d4, tr3);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d4, tr3);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d4, tr4);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d5, tr2);
if (pom != null) {        System.out.print(pom);      }
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

6.2.10. The second iteration - 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: 2][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Tue Feb 17 17:17:52 CET 2015][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Tue Feb 17 17:17:52 CET 2015,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Fri Feb 13 17:17:52 CET 2015][
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2]

6.3. Third Iteration - obligatory

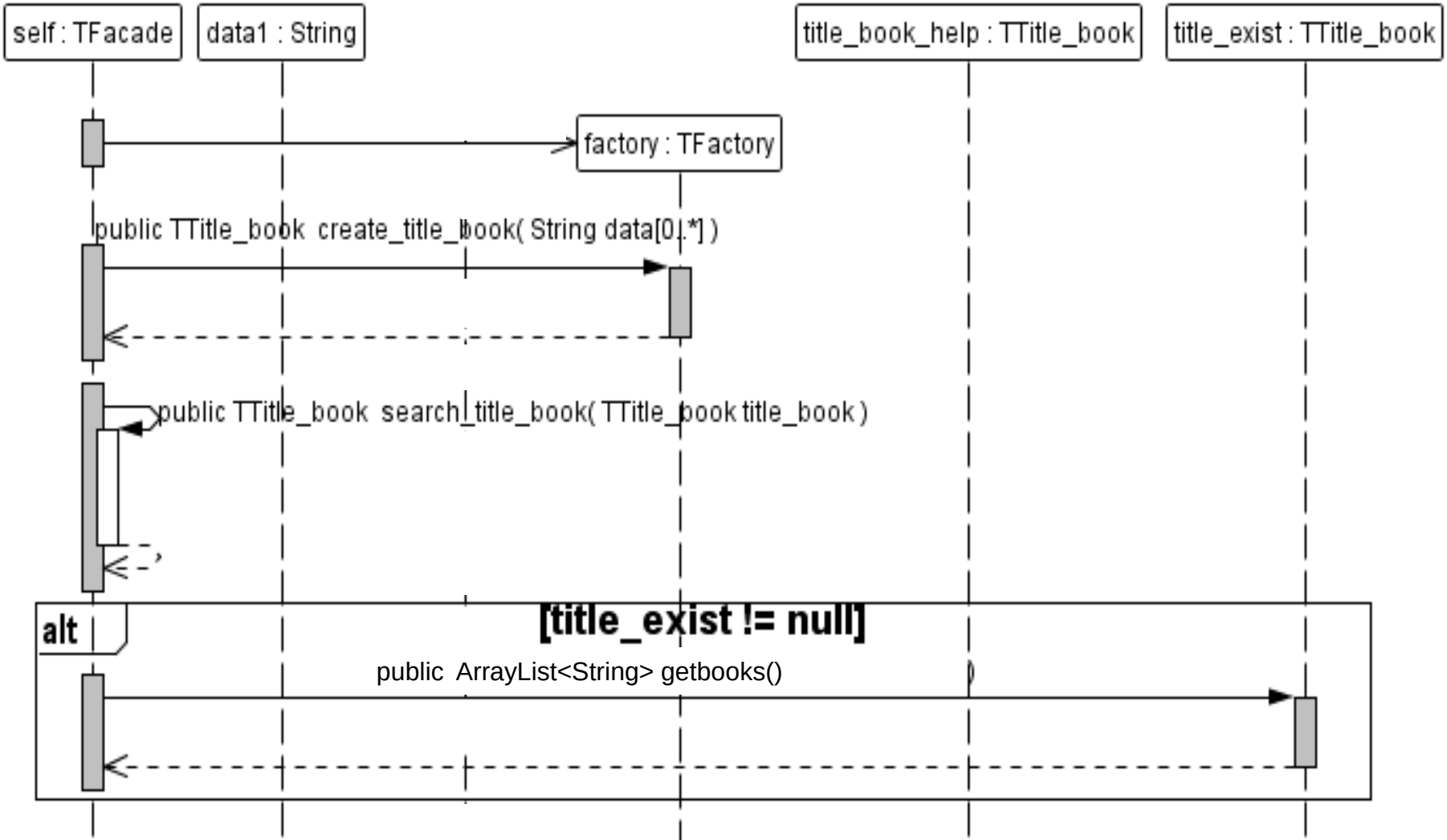
Create the code of methods based on the class diagram and sequence diagrams.

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

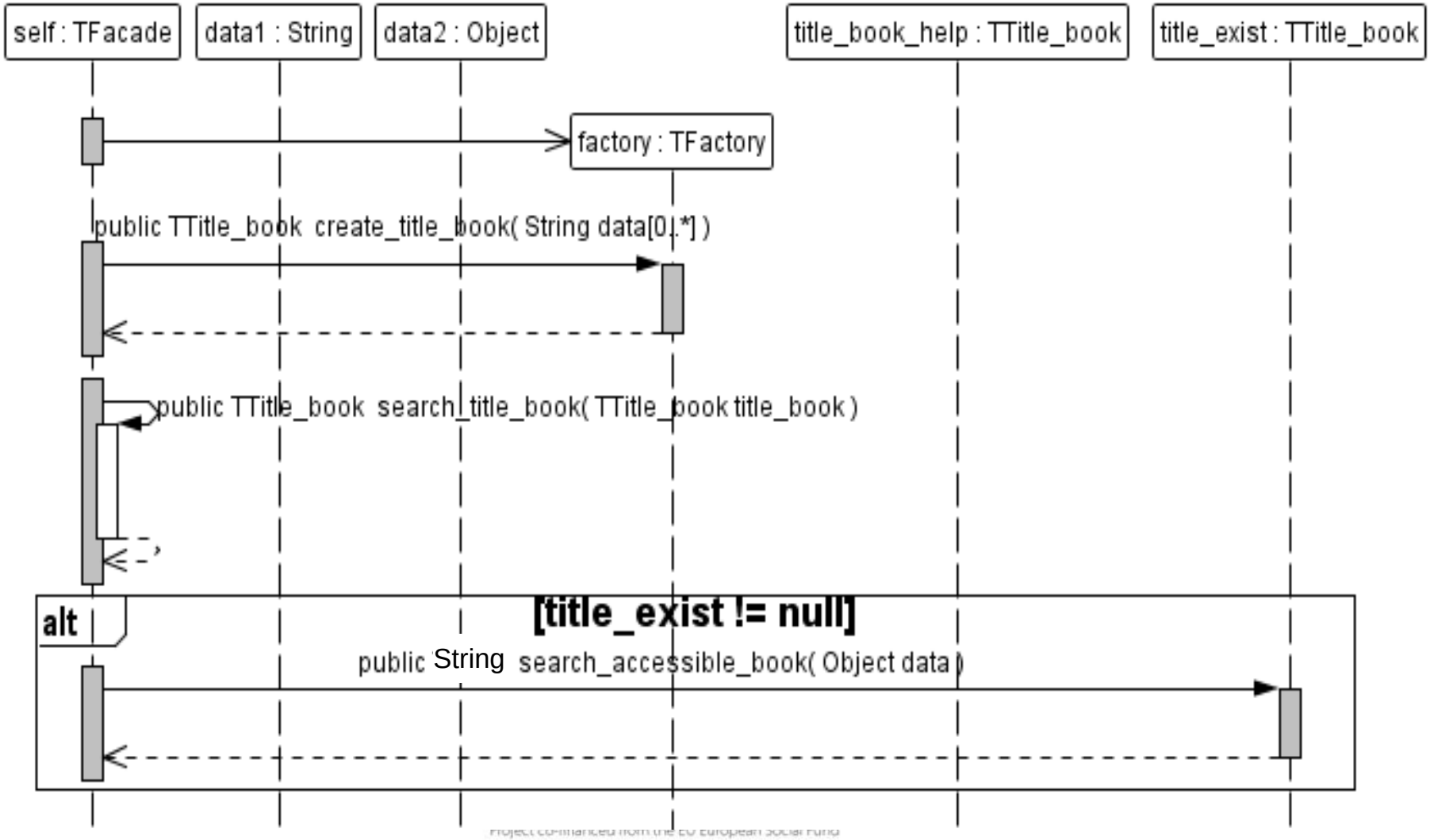
6.3.1. UC_TFacade_Search_title_book

TFacade: public ArrayList <String> Search_title_book(String data[])



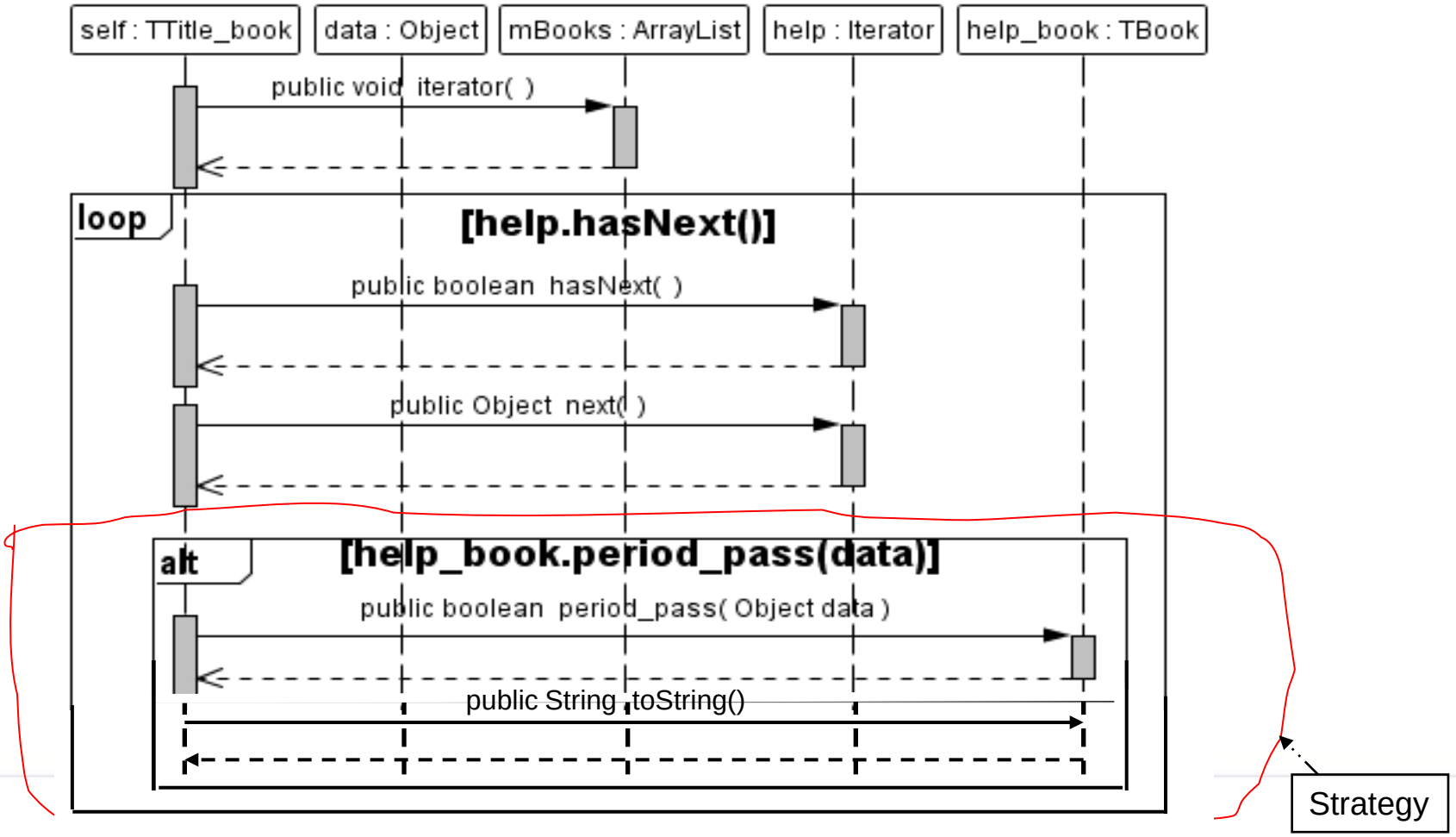
6.3.2. UC_TFacade_Search_accessible_book

TFacade: public String Search_accessible_book(String data1[], Object data2)



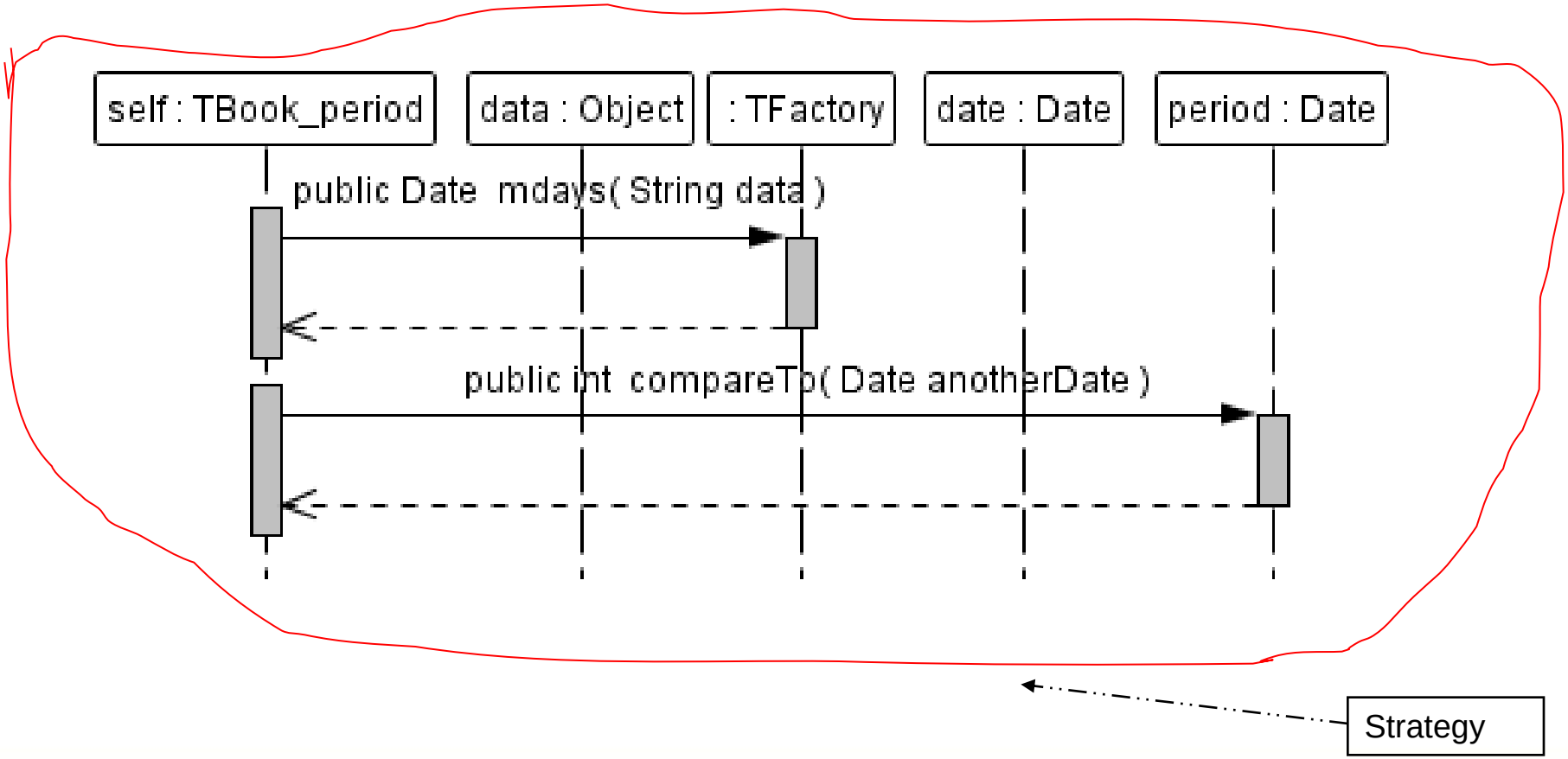
6.3.4. TTitle_book_search_accessible_book

TTitle_book: public String search_accessible_book(Object data)



6.3.5. TBook_period_period_pass

TBook_period: public boolean period_pass(Object data)



6.3.6. The third 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);
```

6.3.7. The third iteration

```
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"};
ArrayList<String> pom = ap.add_book(d1, tr1);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d2, tr1);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d2, tr1);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d2, tr2);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d3, tr2);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d4, tr3);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d4, tr3);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d4, tr4);
if (pom != null) {        System.out.print(pom);      }
pom = ap.add_book(d5, tr2);
if (pom != null) {        System.out.print(pom);      }
System.out.println();
```

6.3.8. The third iteration

```
 System.out.print("\nSearching of a title");
 System.out.print(ap.Search_title_book(t6));
 System.out.print("\nSearching of an accessible book of a select title");
 System.out.print(ap. Search_accessible_book(d4, "2"));
 System.out.println();
 }
}
```

6.3.9. The third iteration - 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: 2][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Tue Feb 17 17:17:52 CET 2015][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Tue Feb 17 17:17:52 CET 2015,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Fri Feb 13 17:17:52 CET 2015][
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2]
Searching of a title[
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2]
Searching of an accessible book of a select title
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Fri Feb 13 17:17:52 CET 2015

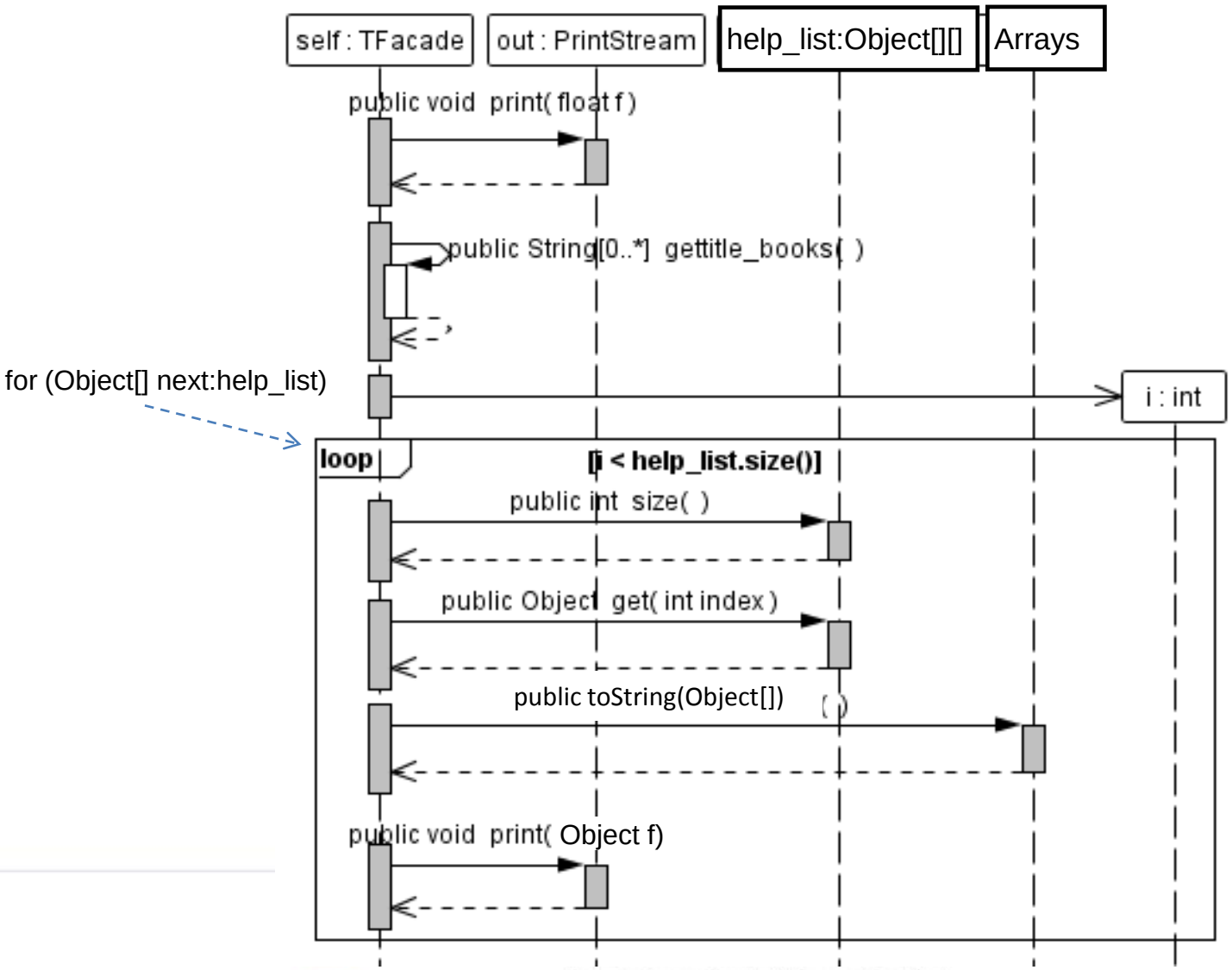
6.4. Fourth Iteration – additional (for improving assessment to 4.5-5.0)

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.

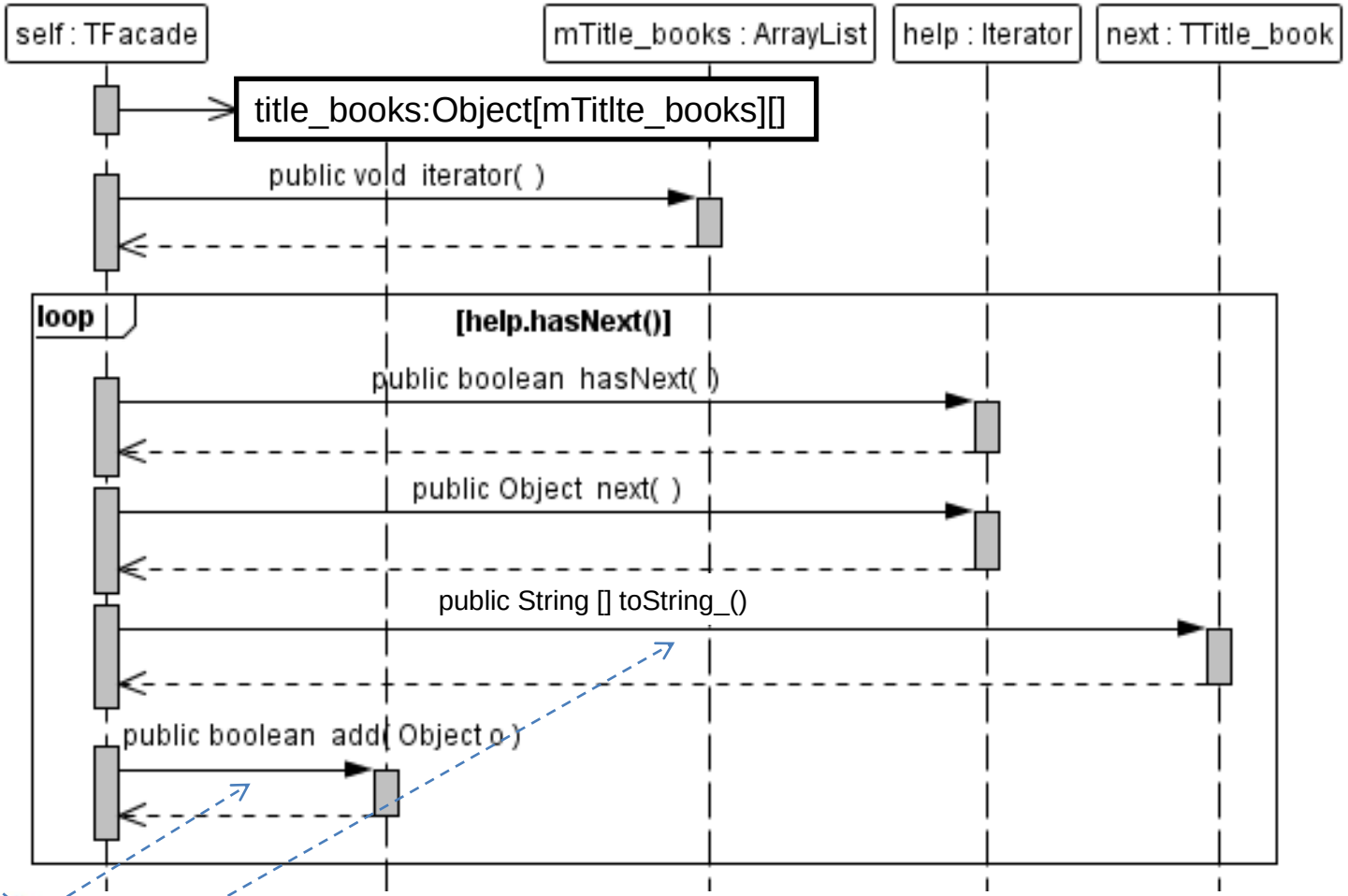
Console presentation

6.4.1. Temp_TFacade_Print_title_books TFacade: public void Print_title_books()



6.4.2. TFacade_gettitle_books

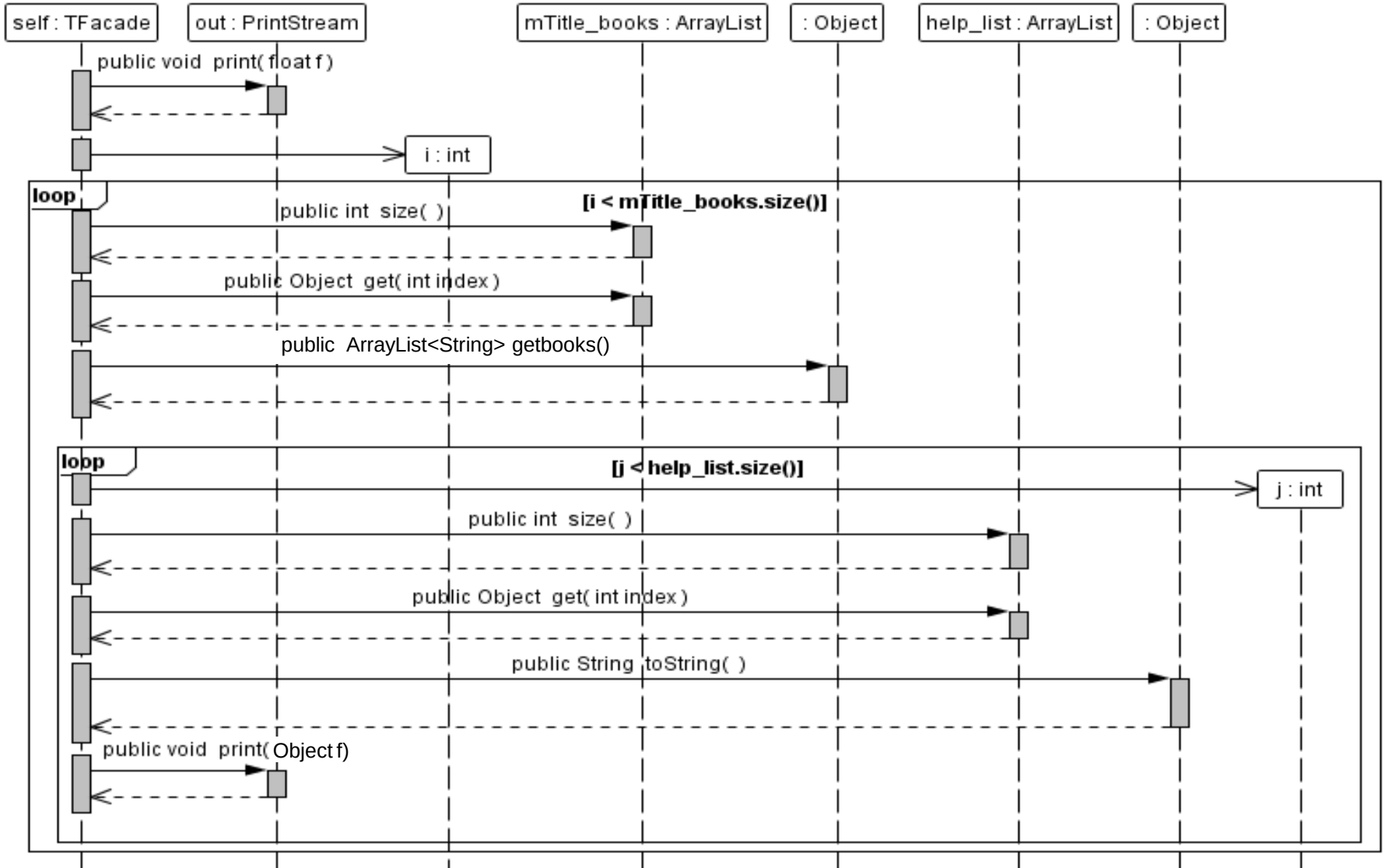
TFacade: public Object[][] gettitle_books()




```
public String[] toString_() {
      String[] title1 = new
String[5];
      title[0] = getPublisher();
      title[1] = getISBN();
      title[2] = getTitle();
      title[3] = getAuthor();
      title[4] = getActor();
      return title1;
   }
```

title_books[i++]=next.toString_();

6.4.3. Temp_TFacade_Print_books TFacade: public void Print_books()



6.4.4. The fourth 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);
```

6.4.5. The fourth iteration

```
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"};
 ArrayList<String> pom = ap.add_book(d1, tr1);
 if (pom != null) {          System.out.print(pom);      }
 pom = ap.add_book(d2, tr1);
 if (pom != null) {          System.out.print(pom);      }
 pom = ap.add_book(d2, tr1);
 if (pom != null) {          System.out.print(pom);      }
 pom = ap.add_book(d2, tr2);
 if (pom != null) {          System.out.print(pom);      }
 pom = ap.add_book(d3, tr2);
 if (pom != null) {          System.out.print(pom);      }
 pom = ap.add_book(d4, tr3);
 if (pom != null) {          System.out.print(pom);      }
 pom = ap.add_book(d4, tr3);
 if (pom != null) {          System.out.print(pom);      }
 pom = ap.add_book(d4, tr4);
 if (pom != null) {          System.out.print(pom);      }
 pom = ap.add_book(d5, tr2);
 if (pom != null) {          System.out.print(pom);      }
 System.out.println();
```

6.4.6. The fourth iteration

```
  ap.Print_title_books();
  ap.Print_books();
  System.out.print("\nSearching of a title");
  System.out.print(ap.Search_title_book(t6));
  System.out.print("\nSearching of an accessible book of a select title");
  System.out.print(ap. Search_accessible_book(d4, "2"));
  System.out.println();
  }
}
```

6.4.7. The fourth iteration - 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: 2][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Tue Feb 17 17:17:52 CET 2015][
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 3 Period: Tue Feb 17 17:17:52 CET 2015,
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Fri Feb 13 17:17:52 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]

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: Tue Feb 17 17:17:52 CET 2015
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Fri Feb 13 17:17:52 CET 2015
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2
Searching of a title[
Title: Title4 Author: Author4 ISBN: ISBN4 Publisher: Publisher4 Actor: Actor4 Number: 2]
Searching of an accessible book of a select title
Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Actor: Actor1 Number: 2 Period: Fri Feb 13 17:17:52 CET 2015
```