# Diagram stanów
# Laboratorium 9

Zofia Kruczkiewicz

# Modelowanie zachowania obiektu za pomocą diagramu stanów.

**Opracowanie diagramu stanów dla obiektu wybranej klasy, reprezentującego wpływ różnych przypadków użycia na zmiany stanów tej klasy, modelowanych za pomocą diagramów sekwencji**

# Modelowanie zachowania obiektu za pomocą diagramu stanów.

## Cel laboratorium:

### Definiowanie diagramu stanów dla wybranej klasy

**Uwaga:** Należy rozwijać projekt UML, wykonany podczas lab3-8.

1. Należy wybrać klasę, która należy do modelu danych zdefiniowanego podczas lab5-7, zawierającą logikę biznesową, podobnie jak klasa **TitleBook** i wykonać dla wybranej klasy diagram stanów podobnie jak przedstawiono to dla klasy **TitleBook** (wykład 6, przykład w tej instrukcji).
2. Zgodnie z definicją diagramów stanów w języku UML (wykład 6), **zdarzeniem dla obiektu danej klasy jest wywołanie operacji tego obiektu przez inny obiekt, a akcjami są operacje użyte do definicji tej operacji.** Podczas laboratoriów 5-7 należało wykonać modele operacji poszczególnych klas, wykonane za pomocą diagramów sekwencji. Taki diagram sekwencji operacji wywołanej podczas zdarzenia zawiera operacje, które są akcjami. Są to: operacje wywołane od innych obiektów oraz operacje typu Message-to-Self. Dodatkowo, do akcji należy dołączyć różne wyrażenia należące do algorytmu operacji zdarzeniowej, ale niemodelowane za pomocą diagramów sekwencji np. wyrażenia matematyczne.

# Modelowanie zachowania obiektów za pomocą diagramu stanów

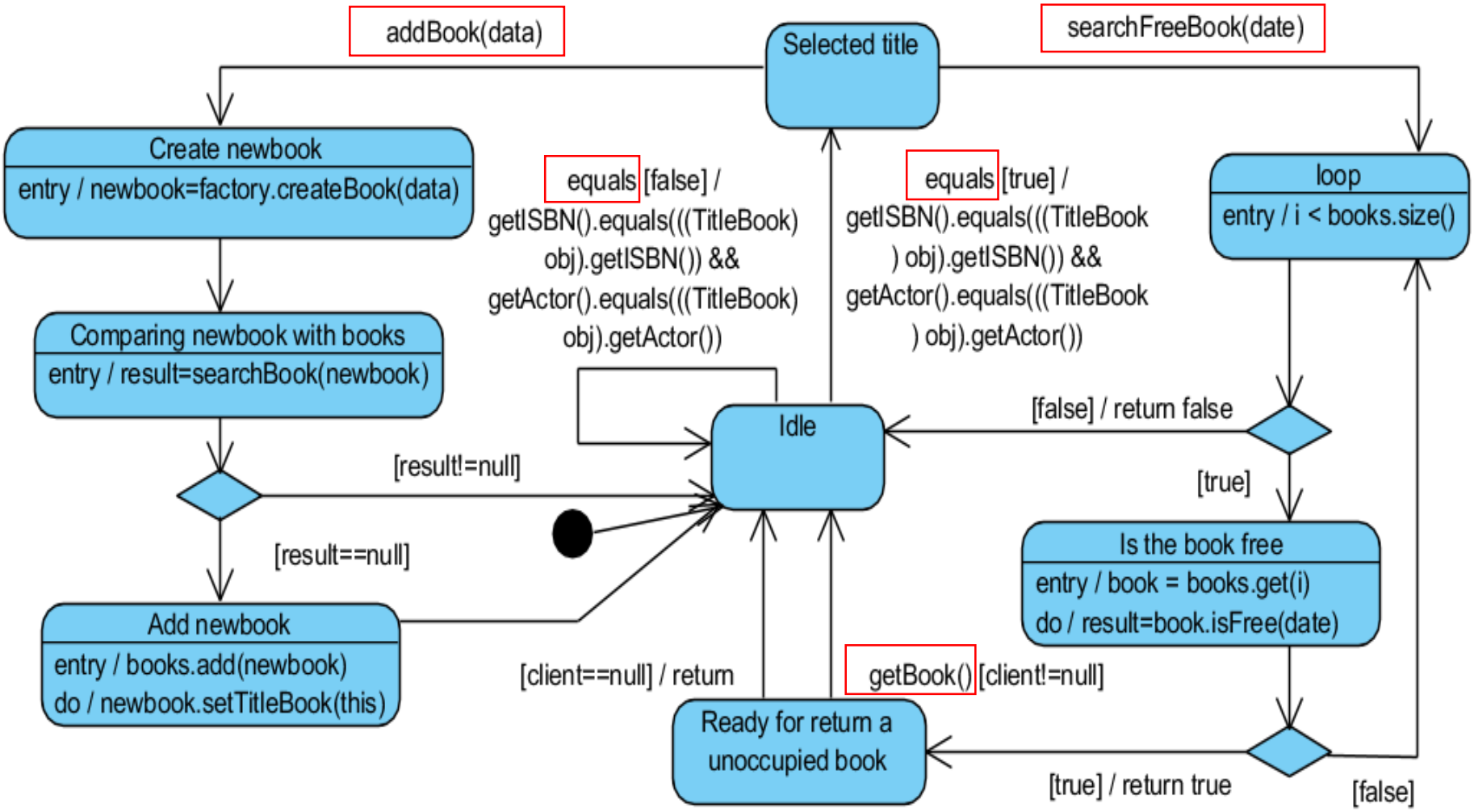
**Przykłady z wykładów 4-6**

**Przykład tworzenia diagramu stanów w środowisku Visual Paradigm**

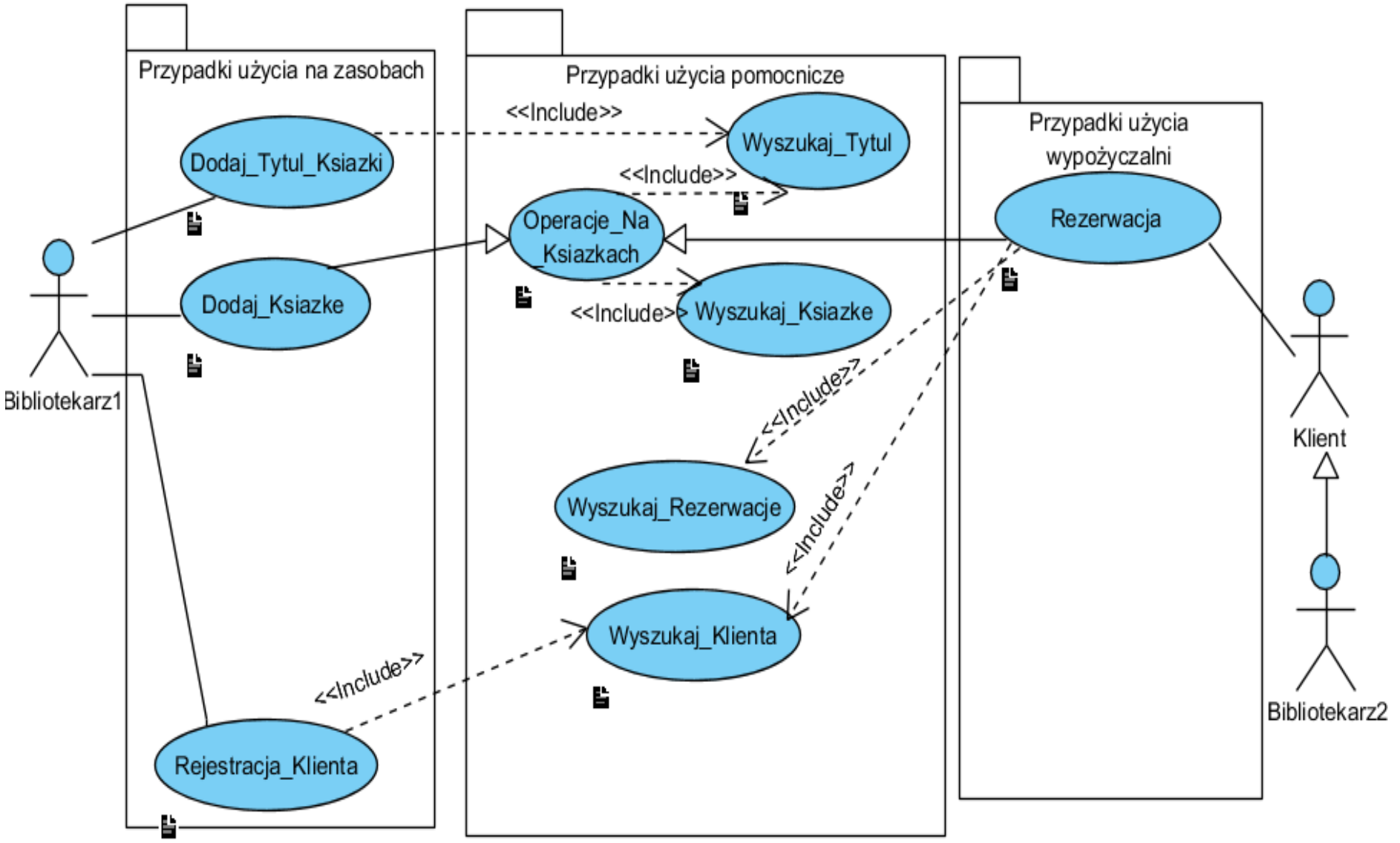
Creating state machine diagrams

# Diagram stanów klasy *TitleBook*

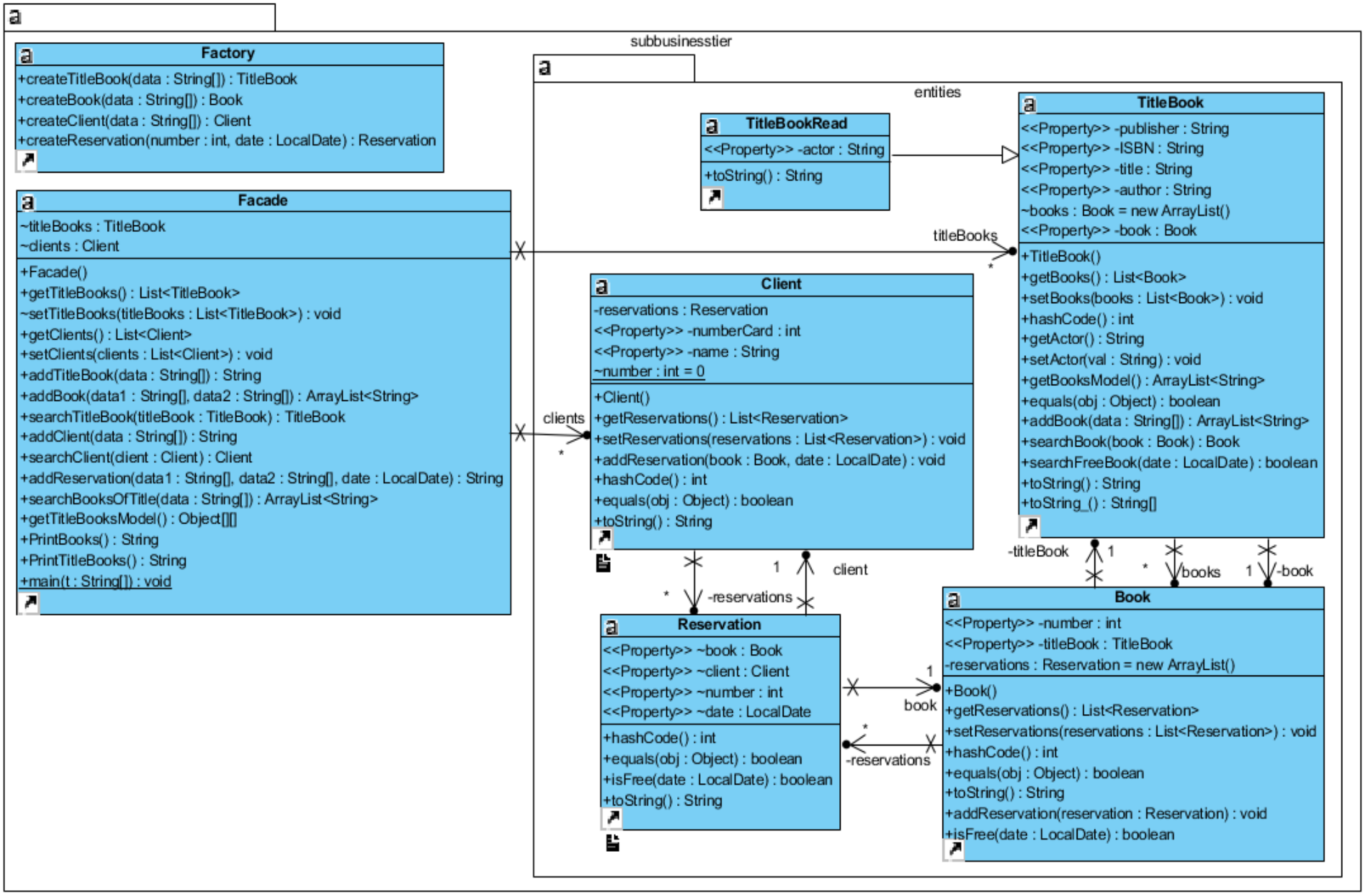
## *Zdarzenia: equals, addBook, searchFreeBook, getBook*



# Diagram przypadków użycia (wykład 4 część 1, przykład 3) – wybrany fragment



# Diagram klas uzyskany w procesie projektowania (przebieg pokazany w dodatku do wykładu 5)



**Klasa Facade udostępnia metody logiki biznesowej – generuje bezpośrednio 3 zdarzenia na obiektach z rodziny TitleBook przez wywołanie jego metod: addBook, searchFreeBook, getBook oraz 1 zdarzenie generuje pośrednio: equals**

```
package subbusinesstier;
import java.time.LocalDate;
import java.time.Month;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import subbusinesstier.entities.Client;
import subbusinesstier.entities.TitleBook;
public class Facade {
   List<TitleBook> titleBooks;
   List<Client> clients;
   public Facade()                                          {  }
   public List<TitleBook> getTitleBooks()                   {  }
   public void setTitleBooks(List<TitleBook> titleBooks)    {  }
   public List<Client> getClients()                         {  }
   public void setClients(List<Client> clients)             {  }
```

**Zdarzenia wywołane na obiektach z rodziny TitleBook przez obiekt typu Facade oraz jego atrybut titleBooks**

**equals, addBook, searchFreeBook, getBook**

```
public TitleBook searchTitleBook(TitleBook titleBook) {}

public Client searchClient(Client client) {}
public String addClient(String data[]) {}
public String addTitleBook(String data[]) {}

public ArrayList<String> addBook(String data1[], String data2[]) {}
public String addReservation(String data1[], String data2[], LocalDate date) {}
//pomocnicze metody
   public ArrayList<String> searchBooksOfTitle(String data[])  {}
   public Object[][] getTitleBooksModel()                      {}
   public String PrintBooks()                                  {}
   public String PrintTitleBooks()                             {}
   public static void main(String t[])                         {}
}
```

PU Operacje_Na_Ksiazkach

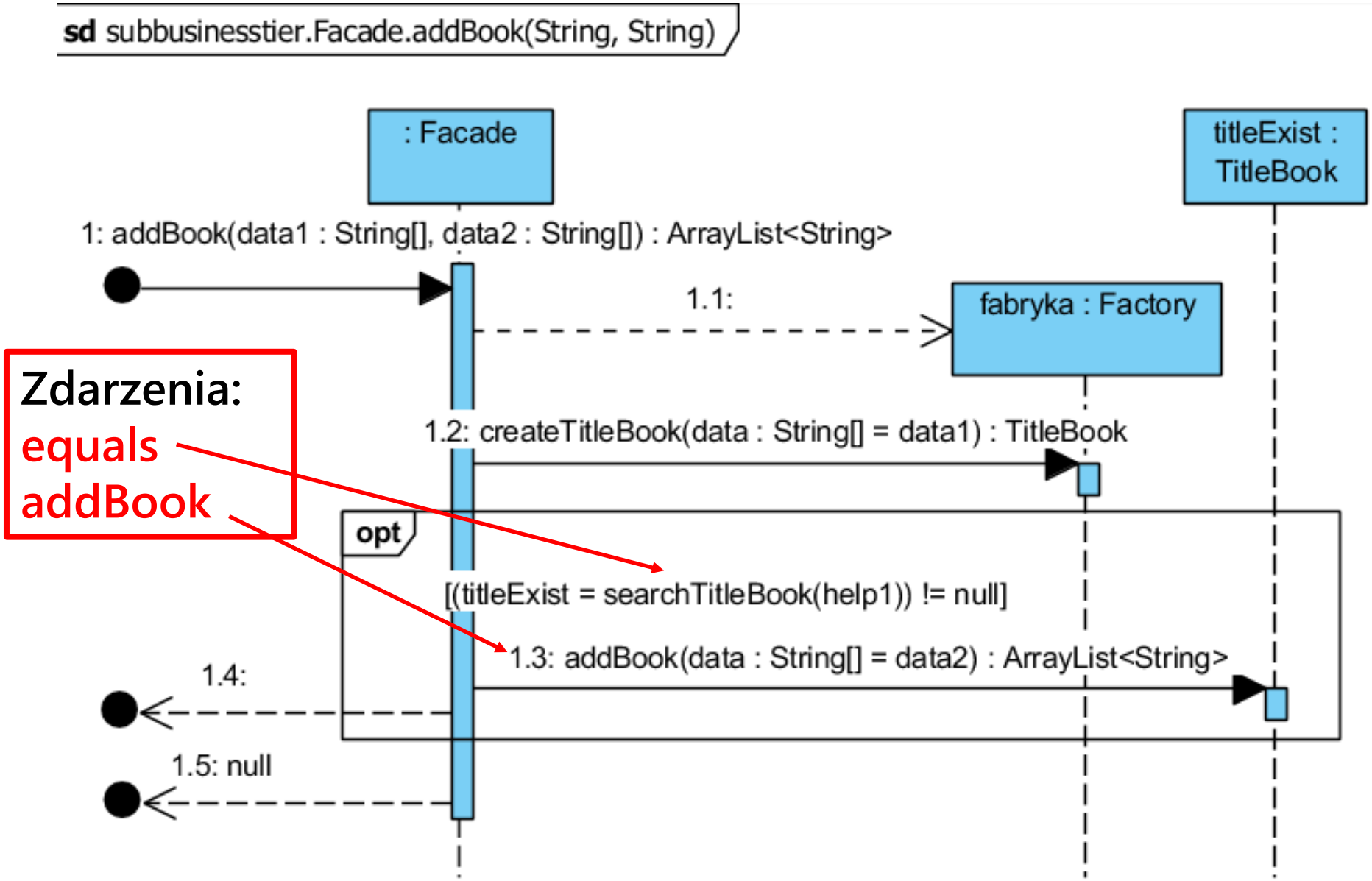
PU Rejestracja_Klienta

PU Dodaj_Tytul_Ksiazki

PU Dodaj_Ksiazke

PU Rezerwacja

# public ArrayList<String> addBook(String data1[], String data2[])



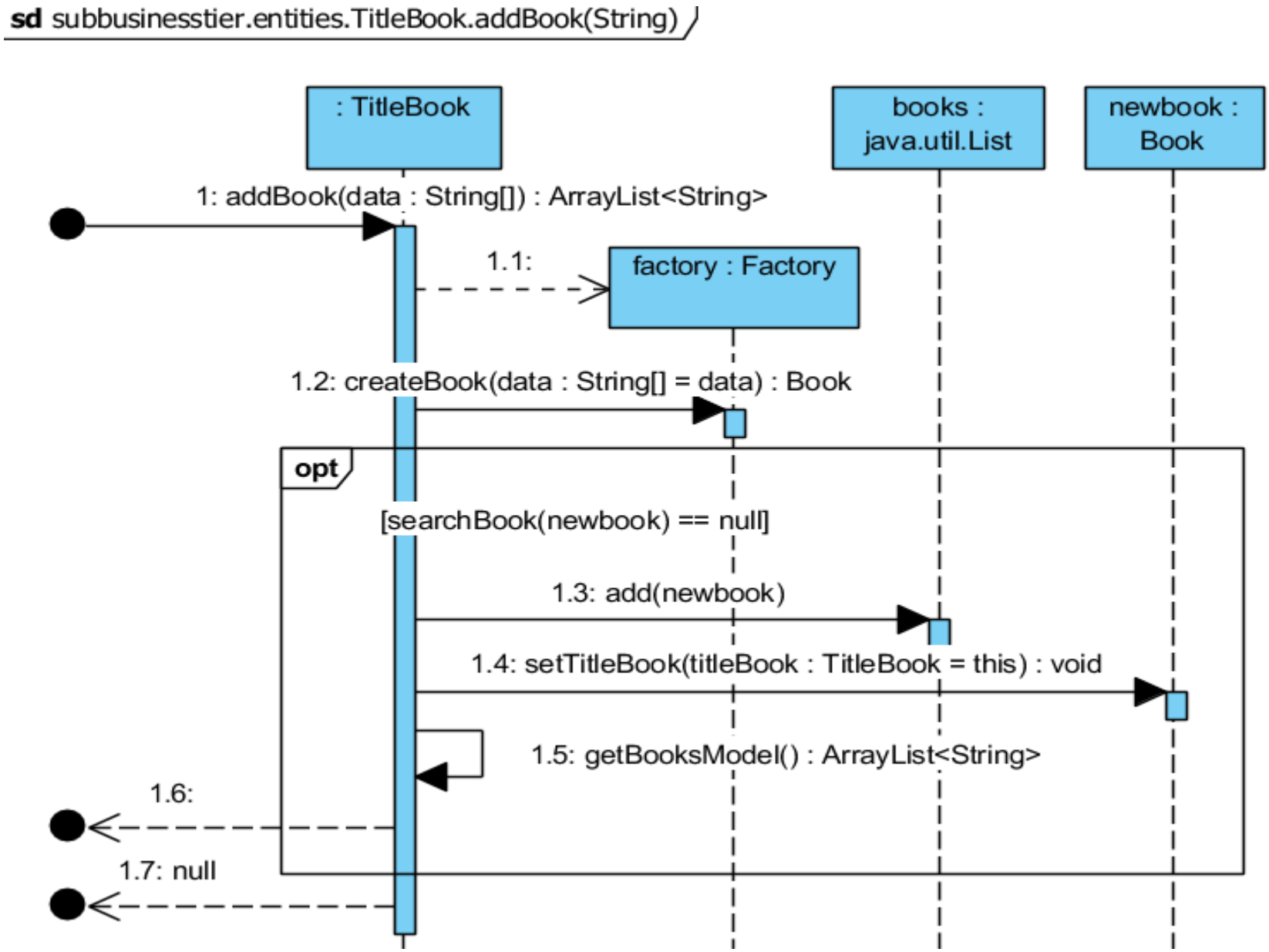
```
//class Facade
  List<TitleBook> titleBooks;
  List<Client> clients;
  public Facade() {
     titleBooks = new ArrayList<>();
     clients = new ArrayList();
  }

public ArrayList<String> addBook(String data1[], String data2[]) {
     TitleBook help1, titleExist;
     Factory fabryka = new Factory();
     help1 = fabryka.createTitleBook(data1);
     if ((titleExist = searchTitleBook(help1)) != null) {   //equals
         return titleExist.addBook(data2);                  //addBook
        }
     return null;
  }
```

```
//class Facade
public TitleBook searchTitleBook(TitleBook titleBook) {
    int idx;
    if ((idx = titleBooks.indexOf(titleBook)) != -1) {
      return  titleBooks.get(idx);
    }
    return null;
  }
```

```
public int indexOf(Object o) {
    if (o == null) {
      for (int i = 0; i < size; i++)
        if (elementData[i]==null)
          return i;
    } else {
      for (int i = 0; i < size; i++)
        if (o.equals(elementData[i]))
          return i;  }
    return -1;  }
```

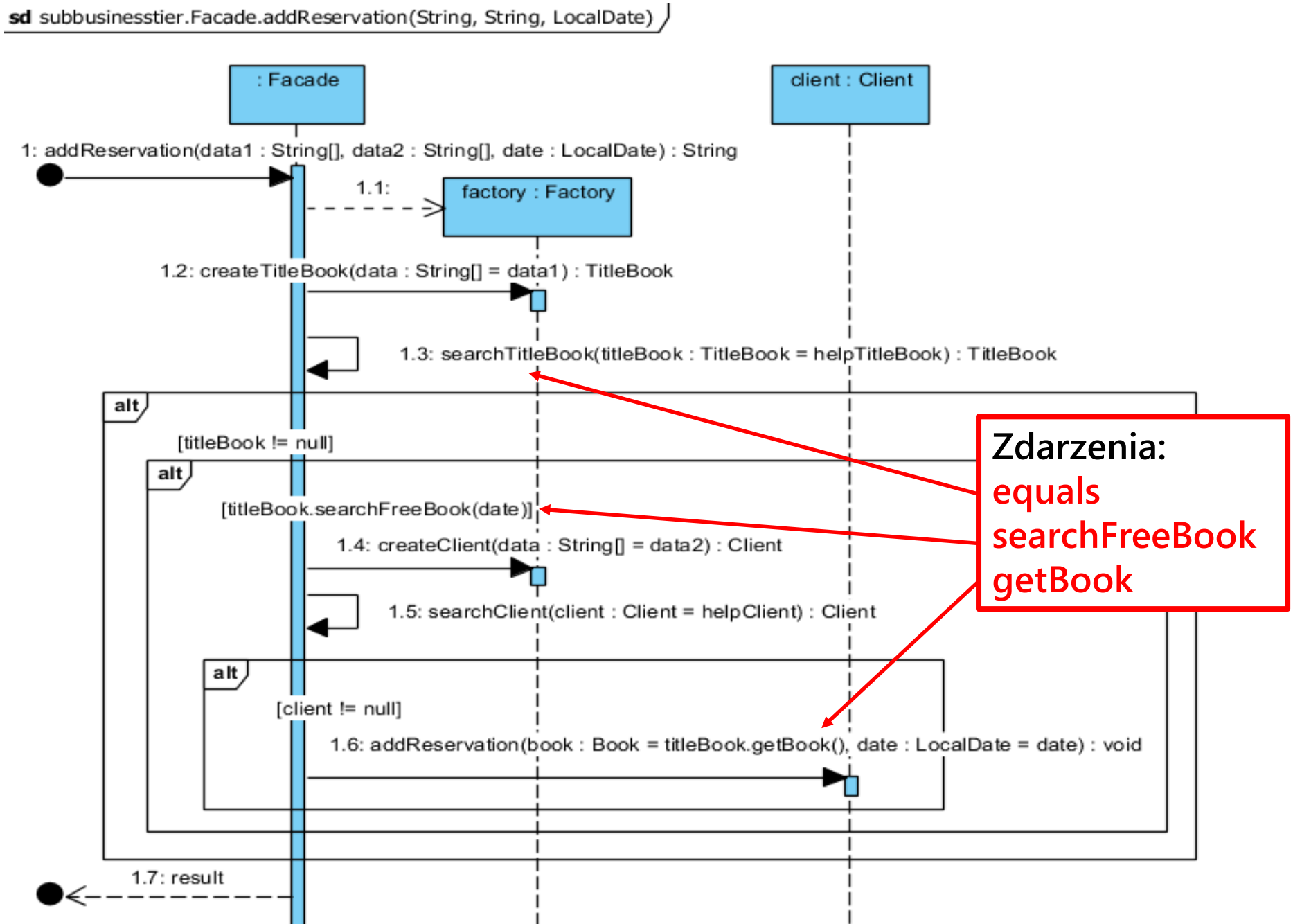
# Wiadomości metody addBook wywołanej jako zdarzenie na obiekcie z rodziny TitleBook przez obiekt typu Facade - odwzorowane na akcje na diagramie stanów



```
//class TitleBook
  List<Book> books;
  public TitleBook() {
     books = new ArrayList();
  }

public ArrayList<String> addBook(String data[]) {
     Factory factory = new Factory();
     Book newbook;
     newbook = factory.createBook(data);
     if (searchBook(newbook) == null) {
        books.add(newbook);
        newbook.setTitleBook(this);
        return getBooksModel();
     }
     return null;
  }
```

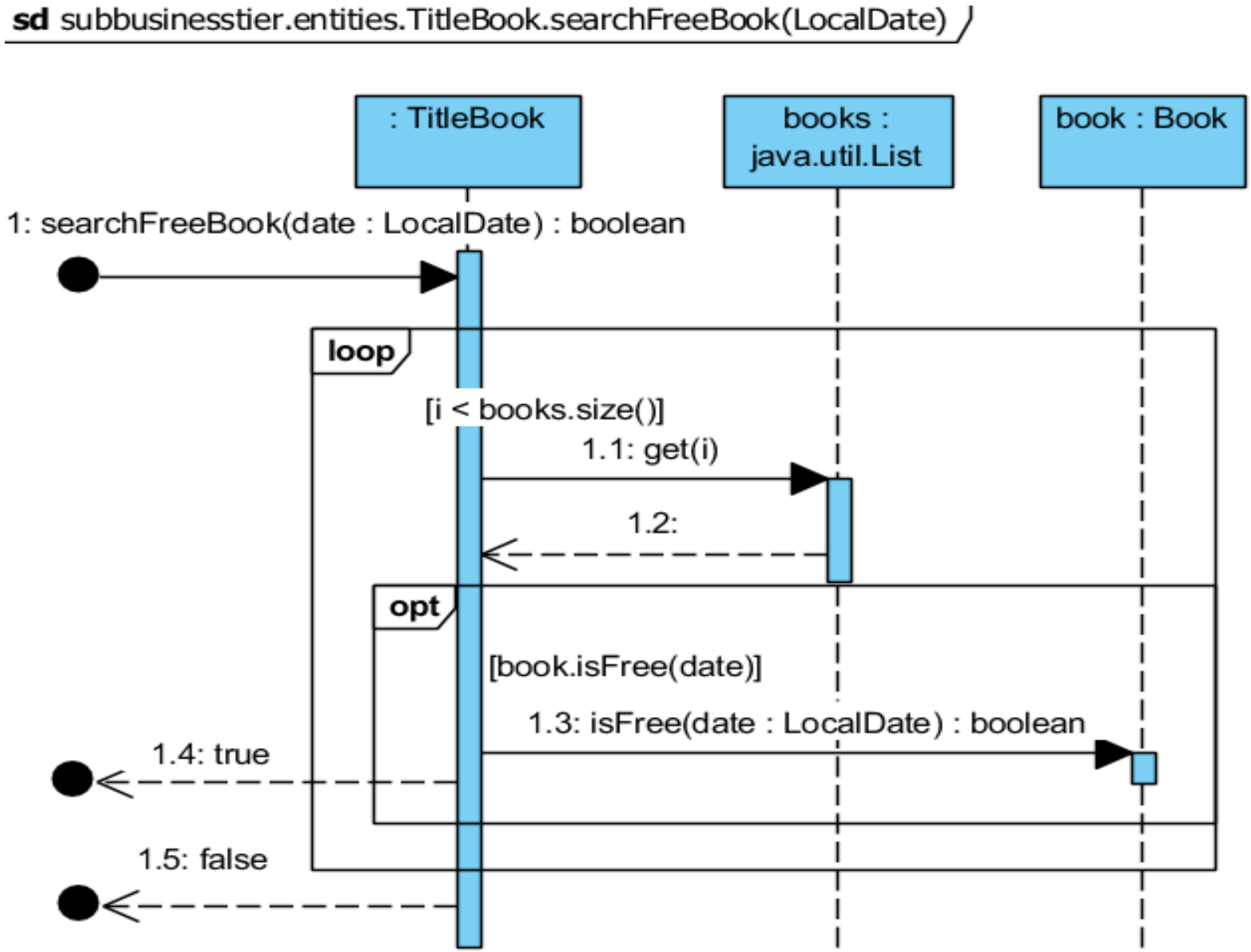
## 2) public String addReservation(String data1[], String data2[], LocalDate date)



```
//class Facade – obiekt tej klasy generuje zdarzenia na obiekcie typu TitleBook

public String addReservation(String data1[], String data2[], LocalDate date) {
    String result;
    Factory factory = new Factory();
    TitleBook helpTitleBook = factory.createTitleBook(data1), titleBook;
    titleBook = this.searchTitleBook(helpTitleBook);          //equals
    if (titleBook != null)
       if (titleBook.searchFreeBook(date)) {                  //searchFreeBook
          Client helpClient = factory.createClient(data2), client;
           client = this.searchClient(helpClient);
          if (client != null) {
             client.addReservation(titleBook.getBook(), date);    //getBook
             result = "reserved";
          } else  result = "no such a client";
       } else  result = "no free book";
     else  result = "no such a title";
    return result;
 }
```

# Wiadomości metody searchFreeBook wywołanej jako zdarzenie na obiekcie z rodziny TitleBook przez obiekt typu Facade – odwzorowane na akcje na diagramie stanów



```
//class TitleBook
   List<Book> books;
   public TitleBook() {
      books = new ArrayList();
   }
  private Book book;   //atrybut book przechowuje obiekt typu
                       //Book wyszukany do rezerwacji

public boolean searchFreeBook(LocalDate date) {
      for (int i = 0; i < books.size(); i++) {
         book = books.get(i);
         if (book.isFree(date))
            return true;
      }
      return false;
   }
```