**Book Selling System**

**Group (VII ) Member List Roll Number**

1. **Khin Nyein Chan 5CS-2**
2. **Yadanar Myo 5CS-5**
3. **Khaing Linn Wai 5CS-8**
4. **Khin Thuzar Win 5CS-9**
5. **Ayme Htun 5CS-16**
6. **Ay Sea 5CS-20**
7. **Nway Oo San 5CS-29**
8. **Ji Aung 5CS-40**

**Introduction**

* The system is Book Selling System.
* This system can be used to find books that the customers want .
* In this system, we describe some books : Head First Java, Java in A Nutshell and so on.
* In editon, you can see some details of books such as the amount of book.
* This system is easy to use and can save time to search books.
* So, we decided to create this system.

**Abstract**

Our project “Book Selling System” is dealing with searching of the books in the store by Client-Server application. The application have action for show the books in store and another action is to find the book that you want. The Server has to produce every information which is stored in the system. The Server consists of the use of the List, serialVersionUID. The Server will keep running until we close its window. Even after the Client closes we can open a new one or even multiple Clients that run simultaneously.

**Objectives**

* To provide book stores and customers.
* To save time and money for customers.
* To search the books easily.
* To improve the distributed applications with the help of RMI – Java program.

**File Names**

1. RMIInterface.java (Interface)
2. Bookstore.java (Server)
3. Book.java
4. Customer.java (Client)

**RMIInterface.java**

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.List;

public interface RMIInterface extends Remote {

Book findBook(Book b) throws RemoteException;

List<Book> allBooks() throws RemoteException;

}

**Bookstore.java**

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

import java.util.List;

import java.util.function.Predicate;

public class Bookstore extends UnicastRemoteObject implements RMIInterface {

private static final long serialVersionUID = 1L;

private List<Book> bookList;

protected Bookstore(List<Book> list) throws RemoteException {

super();

this.bookList = list;

}

//The client sends a Book object with the isbn information on it

//(note: it could be a string with the isbn too)

//With this method the server searches in the List bookList

//for any book that has that isbn and returns the whole object

@Override

public Book findBook(Book book) throws RemoteException {

Predicate<Book> predicate = x -> x.getIsbn().equals(book.getIsbn());

return bookList.stream().filter(predicate).findFirst().get();

}

public List<Book> allBooks() throws RemoteException {

return bookList;

}

private static List<Book> initializeList() {

List<Book> list = new ArrayList<>();

list.add(new Book("Head First Java, 2nd Edition", "978-0596009205", 31.41));

list.add(new Book("Java In A Nutshell", "978-0596007737", 10.90));

list.add(new Book("Java:The Complete Reference", "978-0071808552", 40.18));

list.add(new Book("Head First Servlets and JSP", "978-0596516680", 35.41));

list.add(new Book("Java Puzzlers: Traps, Pitfalls, and Corner Cases", "978-0321336781", 39.99));

return list;

}

public static void main(String[] args) {

try {

Naming.rebind("//localhost/MyBookstore", new Bookstore(initializeList()));

System.err.println("Server ready");

} catch (Exception e) {

System.err.println("Server exception: " + e.getMessage());

}

}

}

**Book.java**

import java.io.Serializable;

public class Book implements Serializable {

private static final long *serialVersionUID* = 1190476516911661470L;

private String title;

private String isbn;

private double cost;

public Book(String isbn) {

this.isbn = isbn;

}

public Book(String title, String isbn, double cost) {

this.title = title;

this.isbn = isbn;

this.cost = cost;

}

public String getTitle() {

return title;

}

public String getIsbn() {

return isbn;

}

public double getCost() {

return cost;

}

public String toString() {

return "> " + this.title + " ($" + this.cost + ")";

}

}

**Customer.java**

import java.net.MalformedURLException;

import java.rmi.Naming;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.util.List;

import java.util.NoSuchElementException;

import javax.swing.JOptionPane;

public class Customer {

private static RMIInterface *look\_up*;

public static void main(String[] args) throws

MalformedURLException, RemoteException, NotBoundException {

*look\_up* = (RMIInterface) Naming.*lookup*("//localhost/MyBookstore");

boolean findmore;

do {

String[] options = {"Show All", "Find a book", "Exit"};

int choice = JOptionPane.*showOptionDialog*(null, "Choose an action", "Option dialog",

JOptionPane.*DEFAULT\_OPTION*,

JOptionPane.*INFORMATION\_MESSAGE*,

null, options, options[0]);

switch (choice) {

case 0:

List<Book> list = *look\_up*.allBooks();

StringBuilder message = new StringBuilder();

list.forEach(x -> {

message.append(x.toString() + "\n");

});

JOptionPane.*showMessageDialog*(null, new String(message));

break;

case 1:

String isbn = JOptionPane.*showInputDialog*("Type the isbn of the book you want to find.");

try {

Book response = *look\_up*.findBook(new Book(isbn));

JOptionPane.*showMessageDialog*(null, "Title: " +

response.getTitle() + "\n" + "Cost: $" +

response.getCost(),

response.getIsbn(), JOptionPane.*INFORMATION\_MESSAGE*);

} catch (NoSuchElementException ex) {

JOptionPane.*showMessageDialog*(null, "Not found");

}

break;

default:

System.*exit*(0);

break;

}

findmore = (JOptionPane.*showConfirmDialog*(null, "Do you want to exit?", "Exit",

JOptionPane.*YES\_NO\_OPTION*) == JOptionPane.*NO\_OPTION*);

} while (findmore);

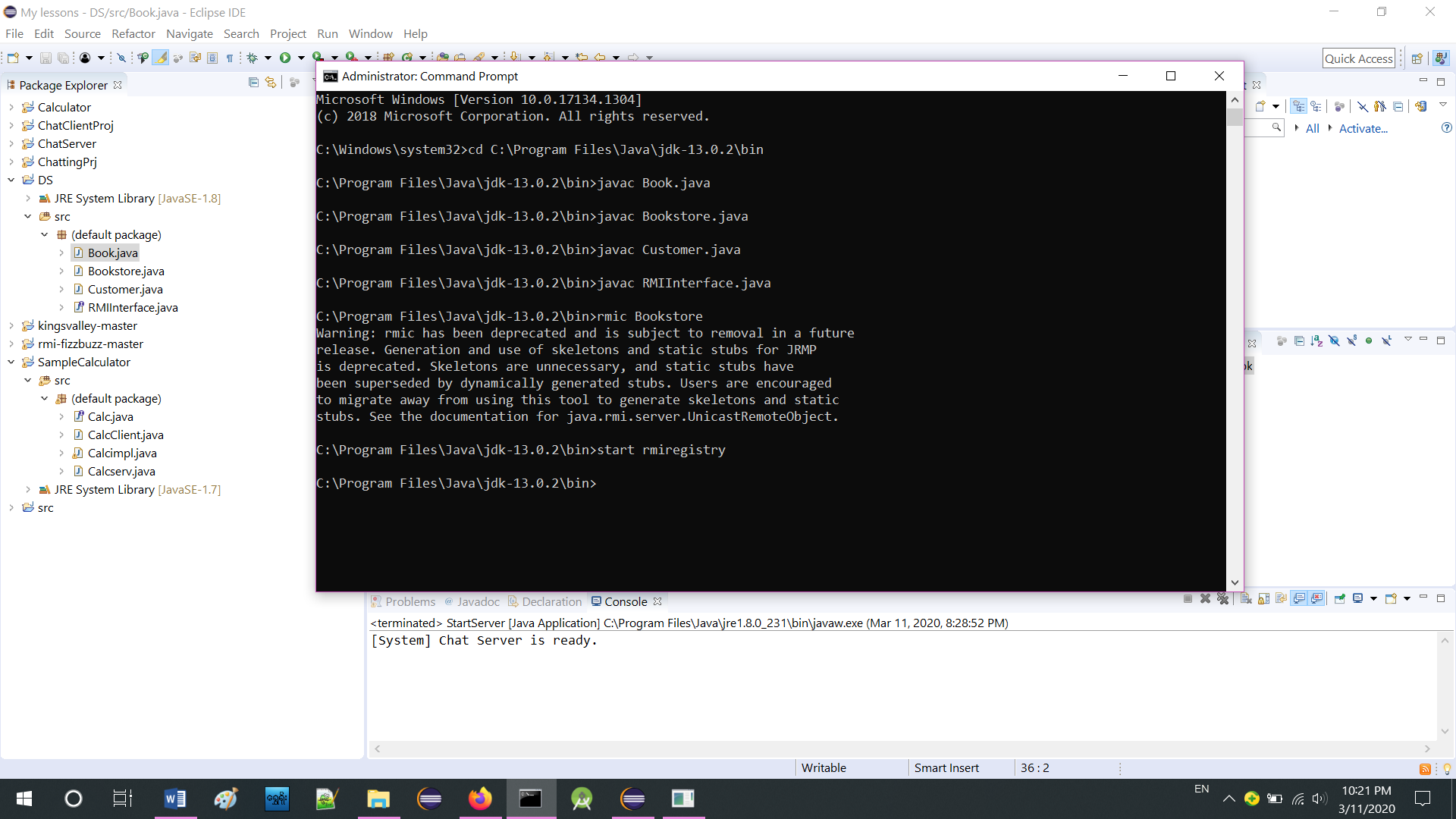
}

}

**System Implementation**

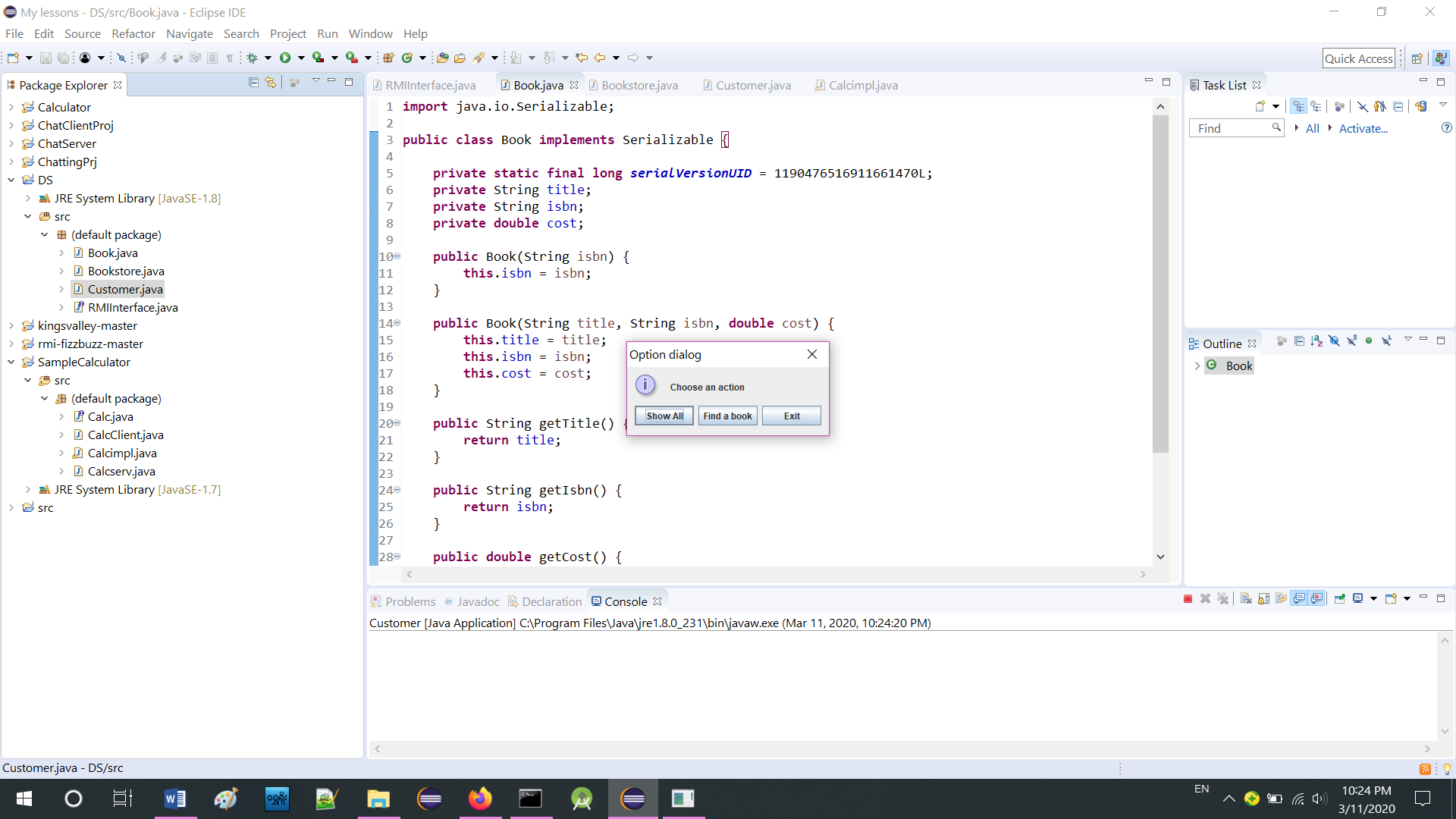
**Command Prompt**

Compile all java file and start rmiregistry.

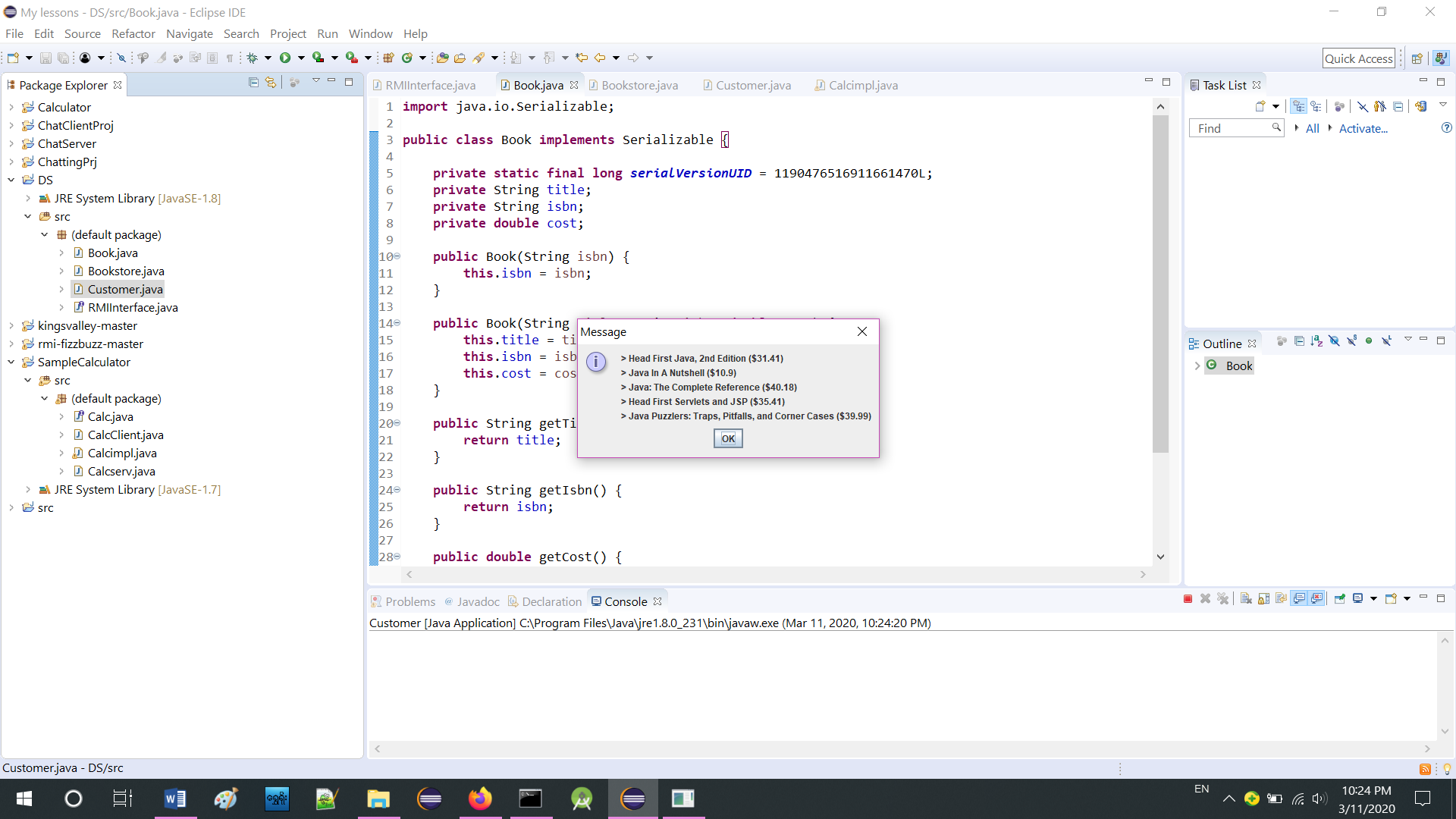


**Outputs**

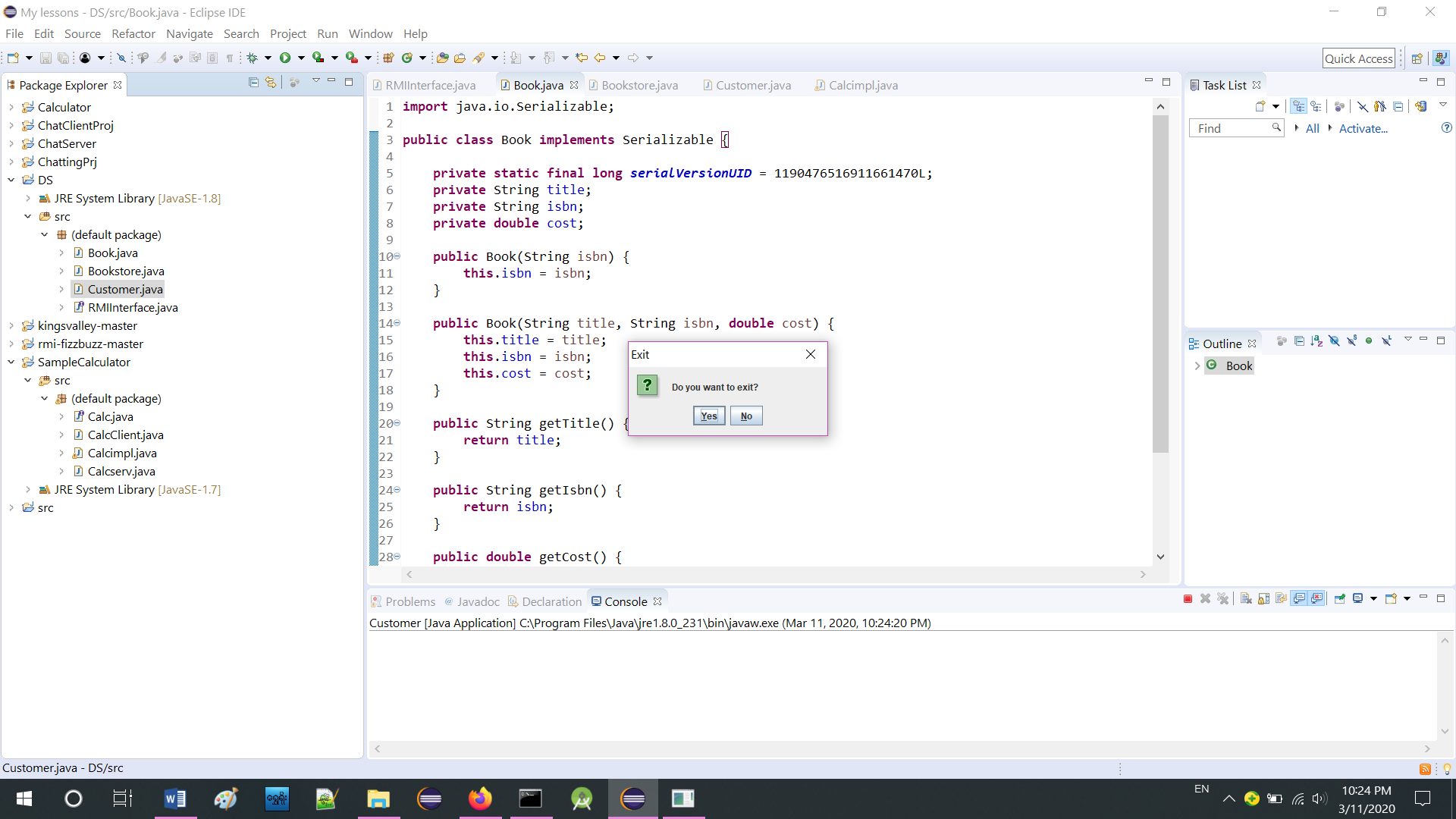
The Customer class runs and prompts us for action.



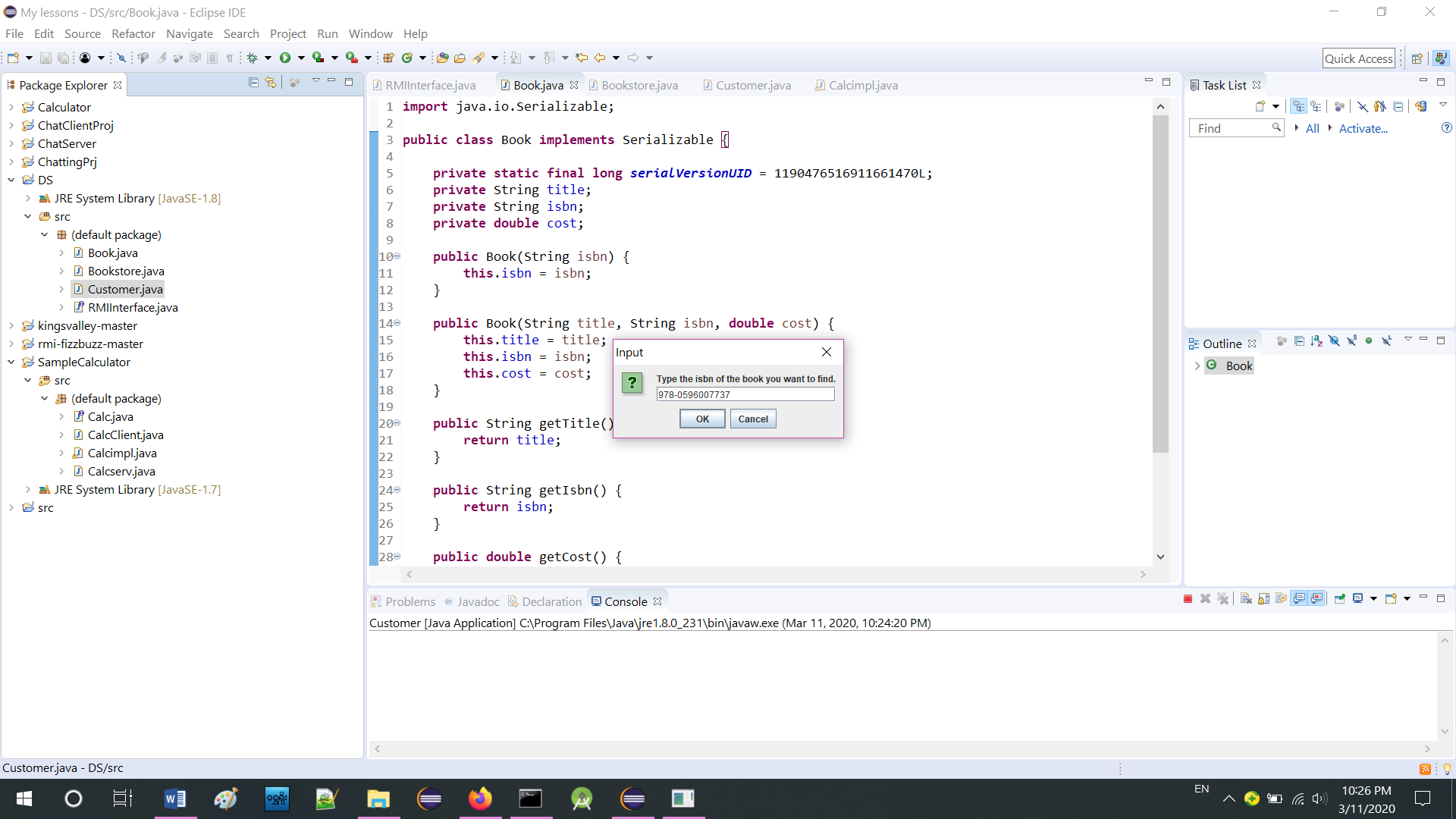
When we click “Show All” button.



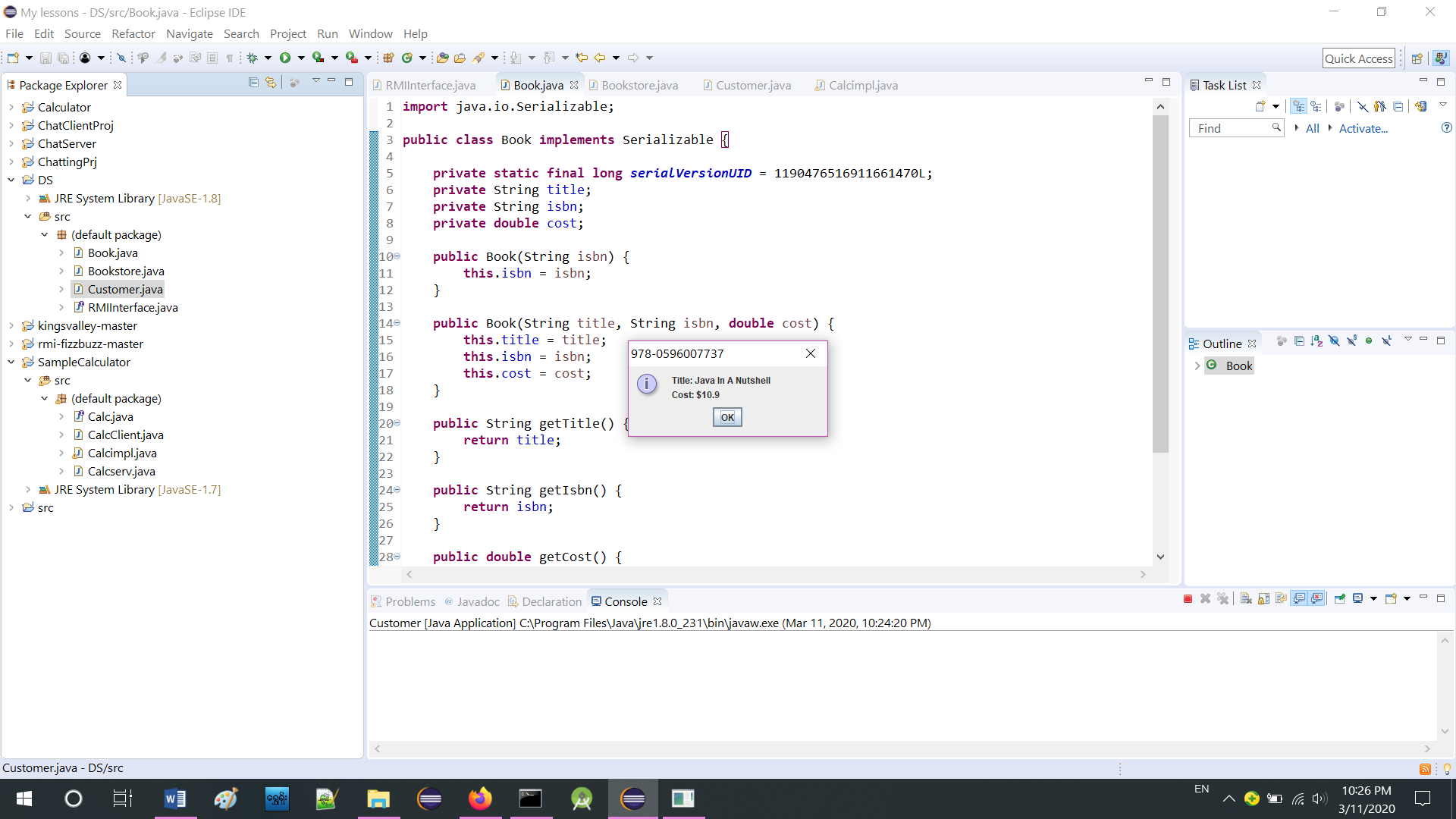
After clicking “OK” button.



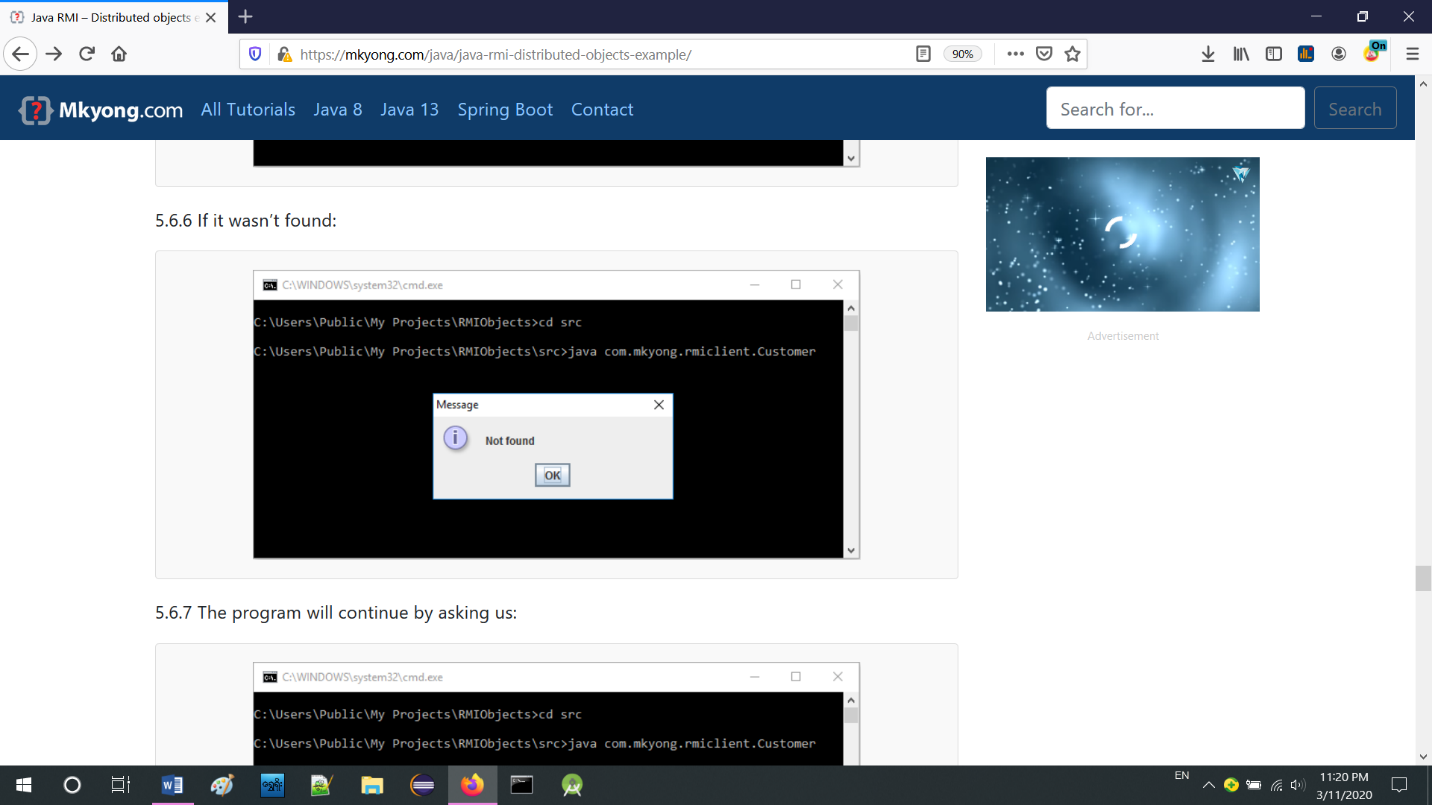
We click “No” button, since we don’t want to exit yet and the following dialog comes up. We type an ISBN (ex. “978-0596007737”) and click “OK”.



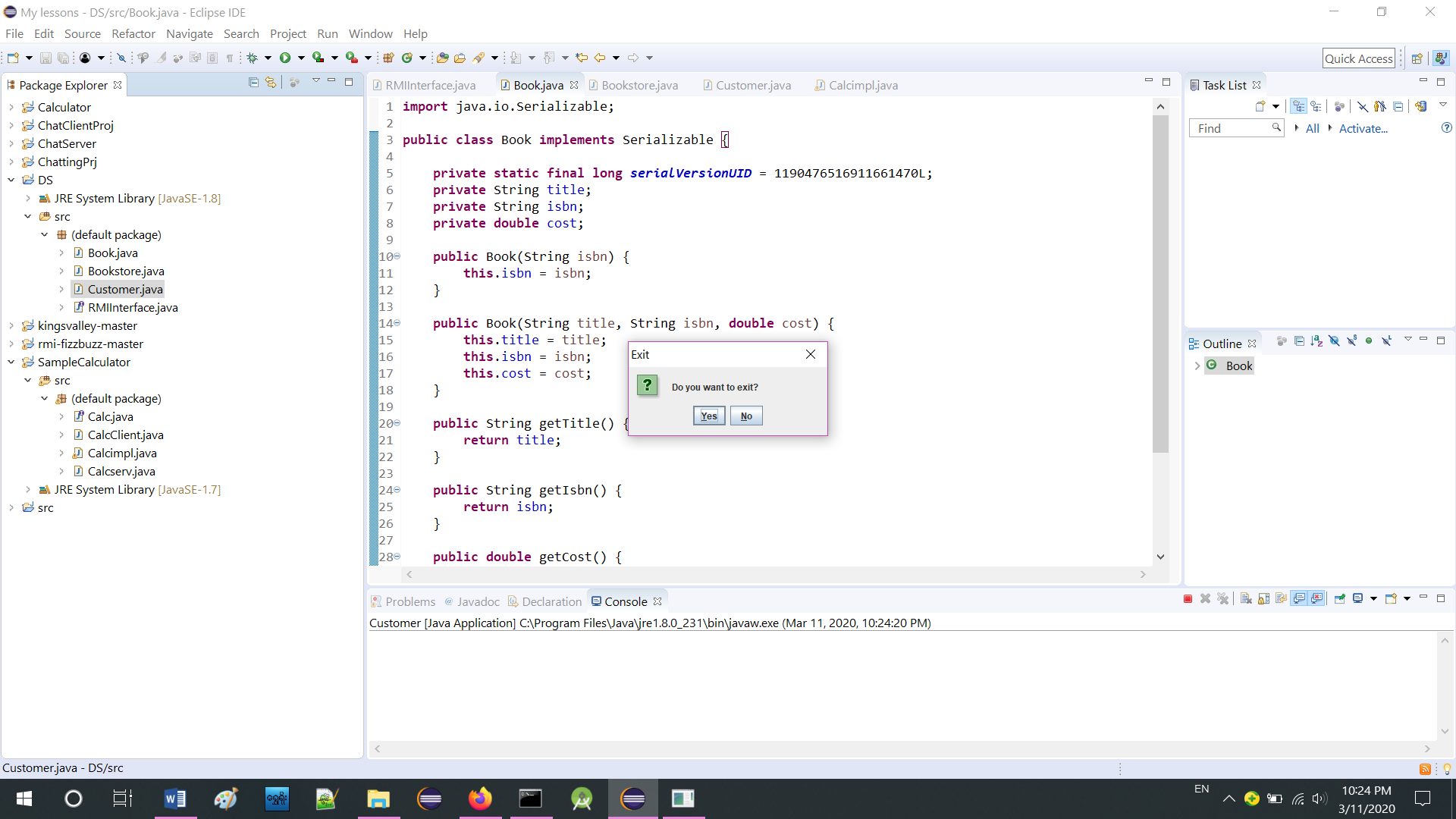
If the book was found in the Server’s list.



If it wasn’t found.



The program will continue by asking us.



If we click “Yes” button the program will exit. If we click “No” it will bring us back to the main menu to choose an action. The Server will keep running until we close its window. Even after the Client closes we can open a new one or even multiple Clients that run simultaneously.

**Requirements**

* Can use jdk 1.5.0\_10 and upper in this system.
* Can use in Linux, all windows such as window- 7,8,10 and upper.
* Can use various types of eclipse versions.