**Assignment SE2**

**Breakdown:**

1. The assignment is worth 20% of your overall grade.
2. Late submissions will lose 40%. If there are special circumstances that prevent you from completing your assignment in time, you should contact me or the course coordinator as soon as possible.
3. You have 5 weeks to complete the assignment 🡪 **due date: Tuesday, 28th March, 2017, by 2.00 pm.**
4. All files must be submitted using Moodle; other methods will not be accepted.
5. **Academic honesty**:

* You are required to acknowledge all the resources that you have used to complete the assignment, such as Java api, and/or any articles, books, lecture notes/labs, help from colleagues/friends etc.
* failure to provide referencing and a reference list will result in result being withheld and case forwarded to course coordinator/head of department. You may use any referencing style you want (APA, Harvard, etc.)

**Deliverables:**

1. A folder containing your **java files**.
2. A **document/report** that contains descriptions of your classes and their respective fields, and methods, and screenshots of your application running.
3. The source code (java files) and the report should be included in a zipped folder with your student number (e.g. **B0001234567.zip**) and uploaded to Moodle by the due date.

**Specifications:**

Create a Java application for a book shop that allows a user to perform the following operations using appropriate graphical user interface:

* view all books in the store;
* update the price of a book using its name;
* insert a book into the database;
* delete a book from the database using its name.

Your application should use the bookshop.sql file uploaded on Moodle in order to create your database bookshop, and the table book; the book table has the following fields: bookID (which is auto-incremented, so no need to worry about it), **bookName, authorName**, and **price**.

**Required Steps:**

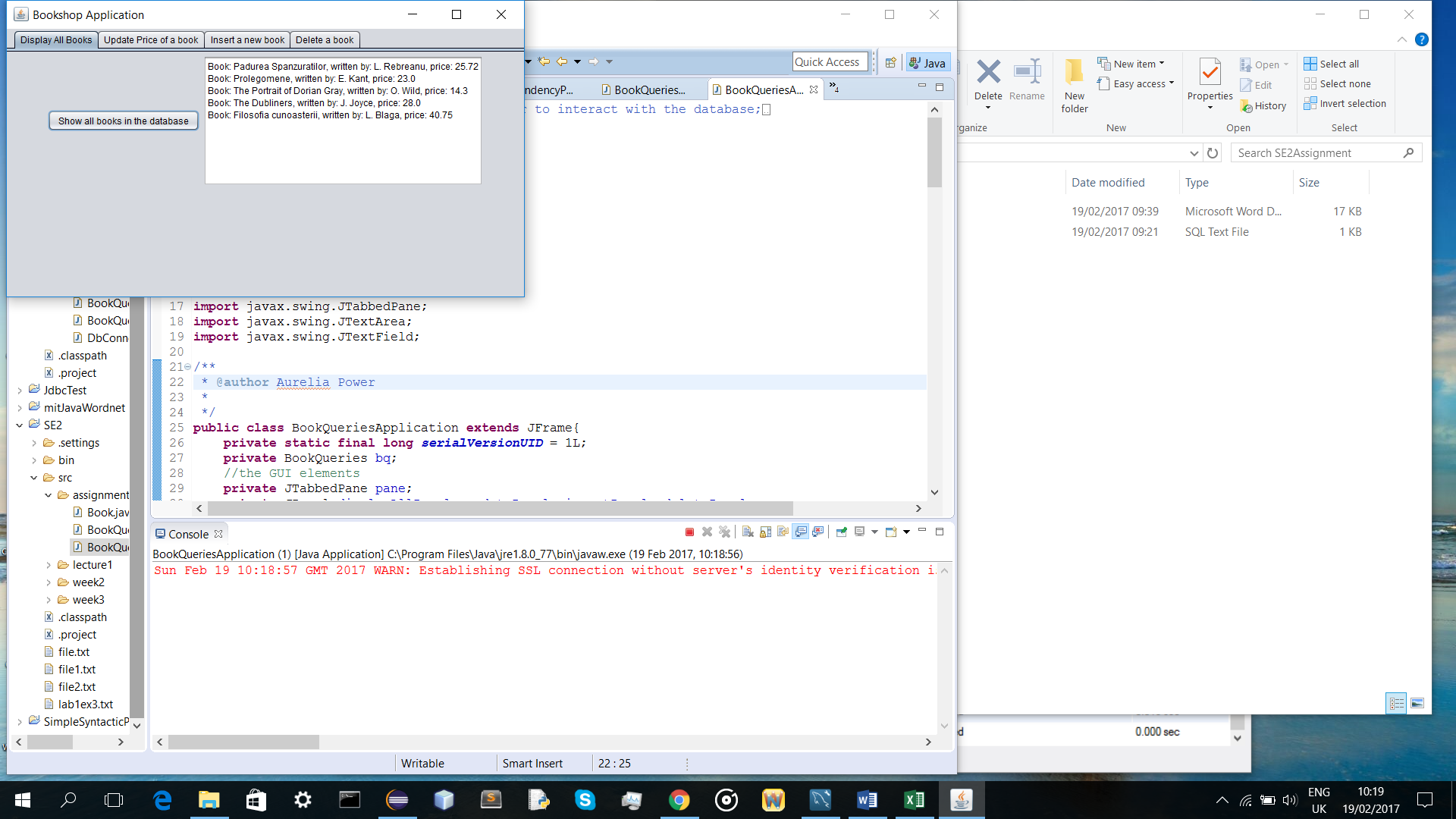
1. **Worth 10%:** You need to create a class **Book** which has variables that reflect the 3 fields in the database: **bookName, authorName**, and **price**; the class should provide a **constructor that takes in 3 parameters** for the 3 fields, and appropriate **setters and getters** for each of the fields. In addition, you should also override the toString method which has the following signature: **public String toString**(){/\*code in here\*/};
2. **Worth 30%:** You need to create a class **BookQueries** that establishes connection to the bookshop database, and uses **4 prepared statements**:

* one for selecting all books from the book table (your query should not include bookId);
* one for updating the price of a book given its name;
* one for inserting a book;
* and one for deleting a book given its name.

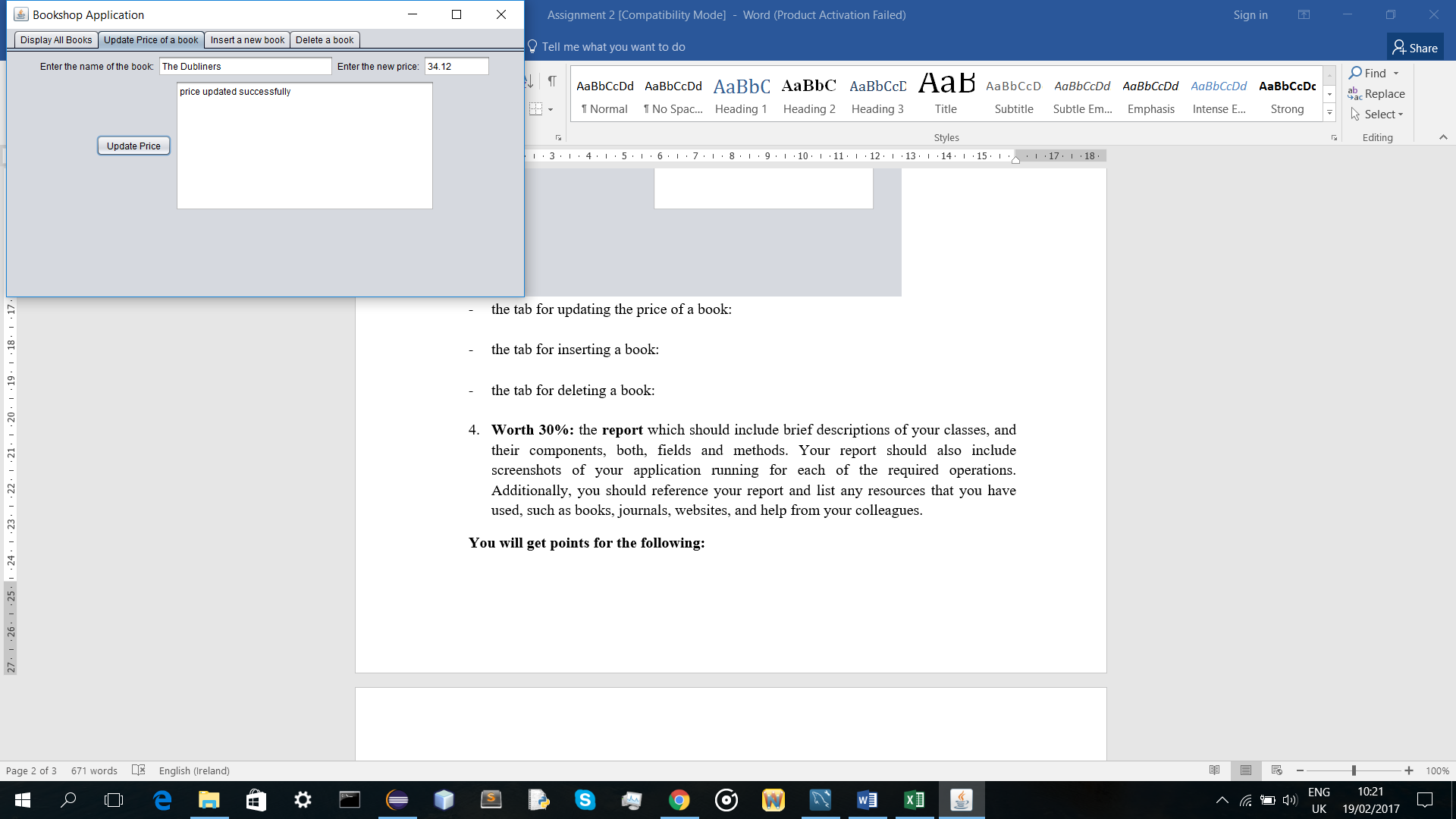
The statements should be initialised using appropriate sql queries. For each of the 4 operations, you should have appropriate methods that return appropriate values, using appropriate execute methods;

1. **Worth 30%:** you need to create a class **BookShopApplication** which contains the GUI elements; it should make use of a **JTabbedPane** that has 4 panes, each pane being associated with one of the above 4 operations on the database: display all books, update the price of a book, insert a book, and delete a book. This class should handle events associated with each operation, and it should also be executable (have a main method); the application should end once you close the window. A simple **possible** layout and the way the user should interact with the application is shown in the following screenshots (NOTE: you are not required to deal with NumberFormatException):

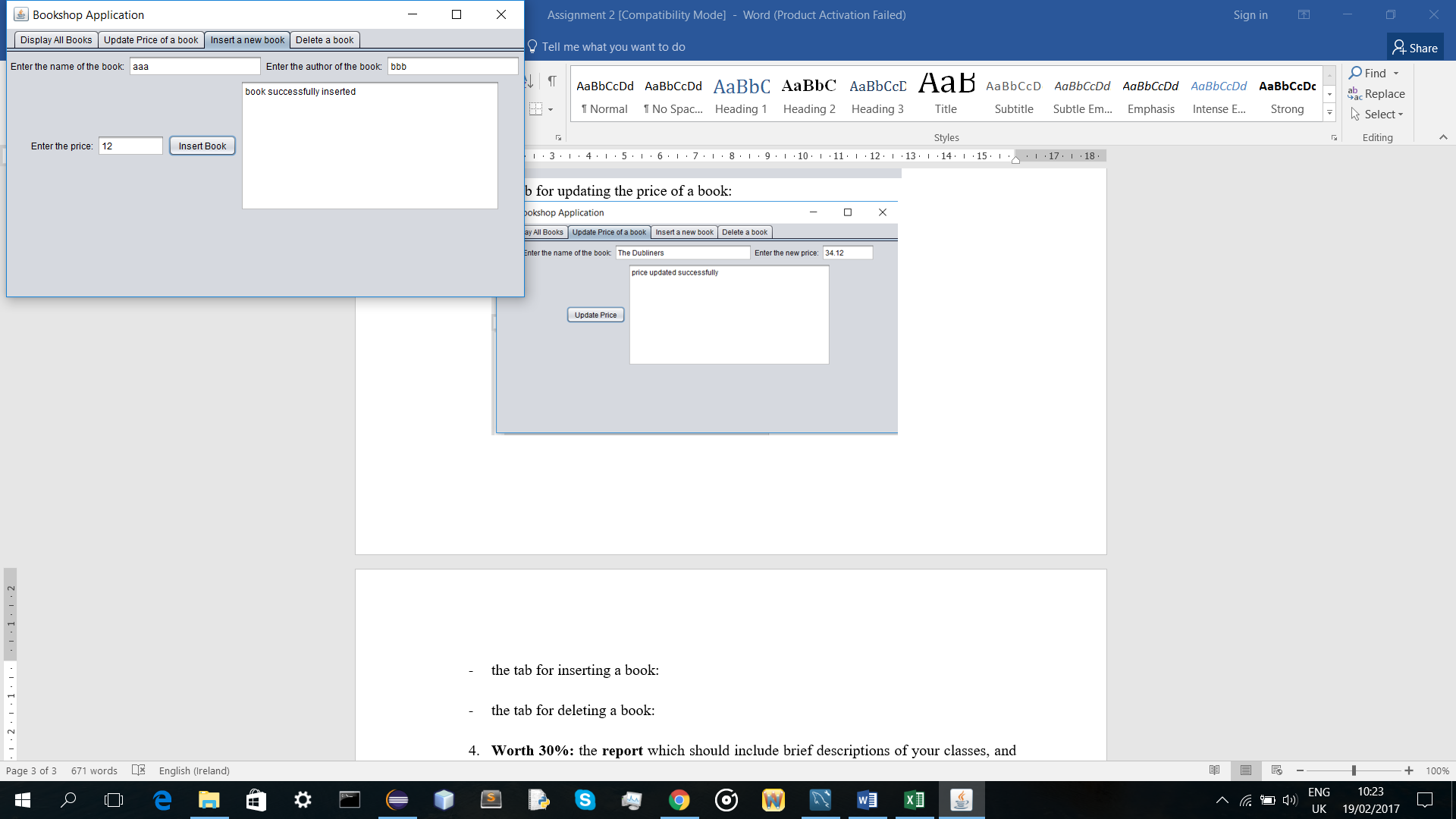
* the tab for displaying all books:



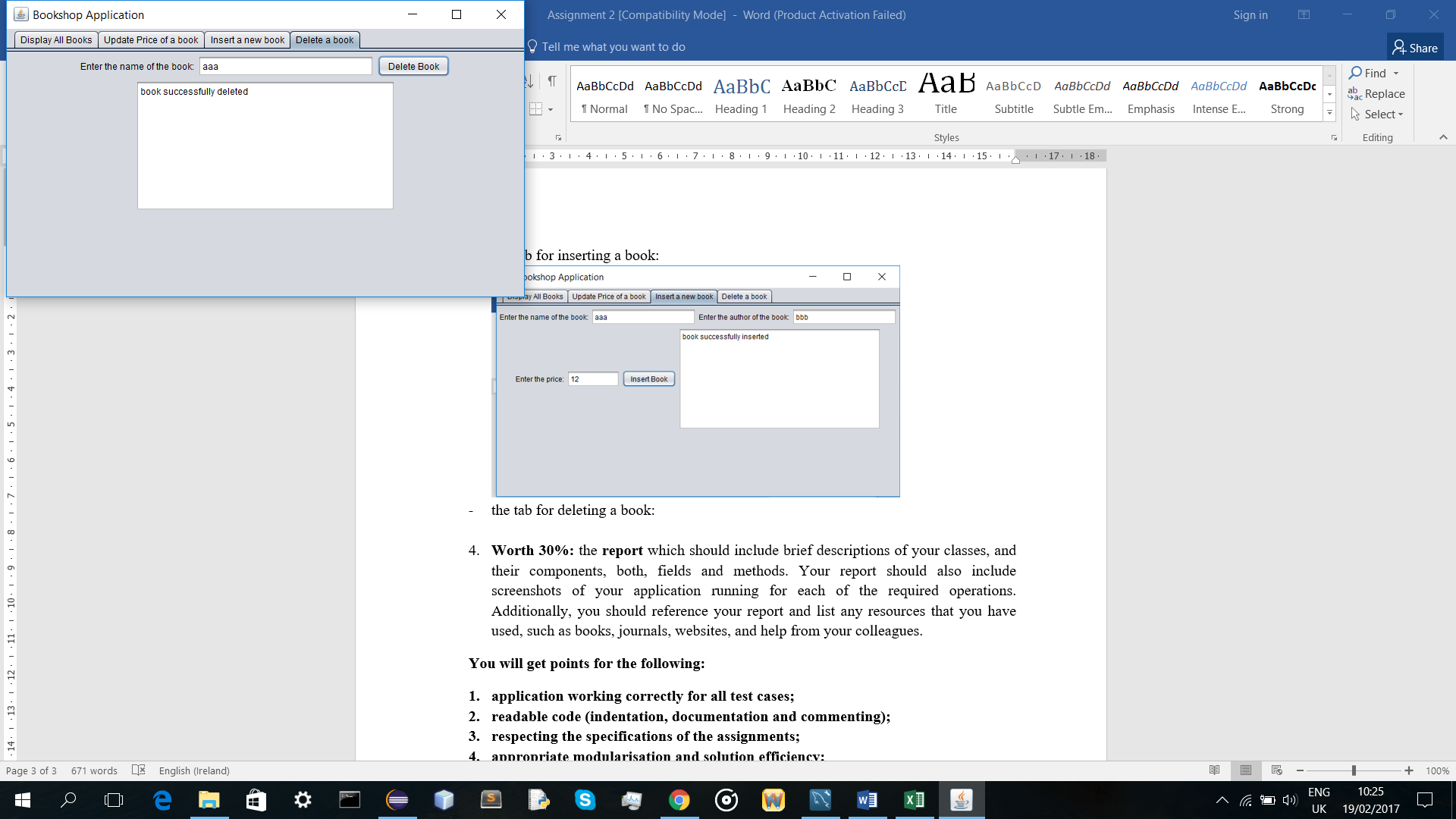
* the tab for updating the price of a book:



* the tab for inserting a book:



* the tab for deleting a book:



1. **Worth 30%:** the **report** which should include descriptions of your classes, and their components, both, fields and methods. Your report should also include screenshots of your application running for at least each of the required operations. Additionally, you should reference your report and list any resources that you have used, such as books, journals, websites, and help from your colleagues.

**You will get points for the following:**

1. application working correctly for all test cases;
2. readable code (indentation, documentation and commenting);
3. respecting the specifications of the assignments;
4. appropriate modularisation and solution efficiency;
5. well written report that follows a logical flow.