VIA University College
Software Technology Engineering
Written examination in DNP
(3 hours)

You are allowed to use any IDE (Visual Studio, VS Code, Rider, etc) and to browse the Internet for information (however, no live communication, e.g. chats, or contact with other people).
When finished, hand in your entire solution in a zip file.

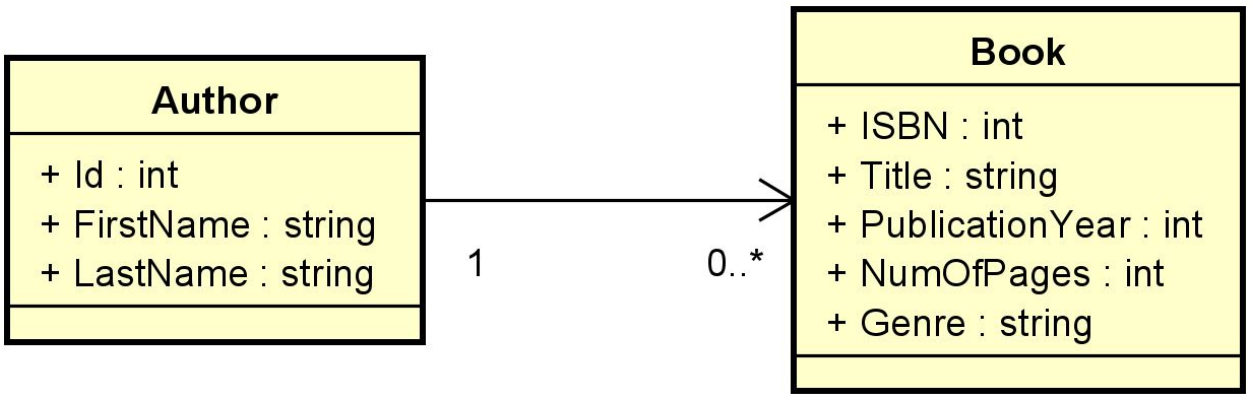
Please note: We have removed part 1. This programming exercise now counts for 100% of the exam grade.

In general, you should remember to follow the conventions and theory taught in class.

Part 1 - Domain classes: Book, Author (10%)

Create a new Solution (or just a folder), call it: [student-number]-Exam-A20.
Create a Web API project (dotnet new webapi), call it AuthorAPI.

Within the newly created project, create a folder, name it appropriately. It must contain the following model classes:



Part 2 - Database (20%)

Add data annotations to the model classes to provide the following constraints:
Author:

- Id - is the primary key
- FirstName - it is required, and the maximum length is 15

- LastName - it is required, and the maximum length is 15

Book:

- ISBN - is the primary key
- Title - it is required, and the maximum length is 40

Create a folder for your database access, name it appropriately.

Use Entity Framework Core to create an SQLite database with tables that can contain Author and Book objects.
ISBN must be used as the id of a Book.

Hint: You will need these two packages:

- Microsoft.EntityFrameworkCore.Design
- Microsoft.EntityFrameworkCore.Sqlite

[NB! If you are having trouble implementing this step, you can resort to using the InMemory provider instead (worth up to 10% instead of 20%), or simply use a list of Books and a list of Authors as your data source (worth up to 5%)].

Part 3 - Web API (30%)

This part is divided into two sub-parts. If you only manage sub-part A, you can still continue to Part 4.

Sub-part A (12%)

Create a Web API Controller for Authors, in an appropriate folder.
Inject your database context (from part 2) into the Controller so that you can persist Author objects to the database as well as retrieve them.

You must create the following endpoints for this controller:

POST: Author - takes an author object to be persisted to the database.
GET: Authors - gets a list of all Authors. *(Hint: you may need to Include() the books for each author for later use)*

You must include model validation, and error handling.

Sub-part B (18%)

Create a Web API Controller for Books, in the same folder as above.

Inject your database context (from part 2) into the Controller so that you can persist Book objects to the database as well as retrieve them.

You must create the following endpoints for this controller:

GET: Books - return all books.

POST: Book - takes a book object, which must be stored in the database. You must also indicate which author wrote the book.
The URL to POST a book should look something: **localhost:5001/Book/3**
Which should add a book, written by the author with id 3.

DELETE: Book (taking the id of the book)

You must include model validation, and error handling.

Part 4 - Web app (40%)

Create a new Blazor project (server, or wasm) in the same solution/folder, call it AuthorBlazor.
If you are not familiar with Blazor, you may use MVC or Razor Pages.
Overview
You must create a number of pages, see below.

- Create author
- Create book, select existing author
- View all books, with attached author name:
    - Filter, by criteria:
        - Author first and last name
        - Book title

Sub-part a
Create a page which lets the user create new authors. It must be possible to provide a first name and a last name of the author (books will be added later). The page must call to the web api so that the created author is persisted in the database.

Sub-part b
If you didn't do Part 3B, skip this step.
Create a page which lets the user create new books. Each book should take an ISBN, a title, year of publication, number of pages and an existing author who wrote the book.
For selecting the author you have two options:

1. Get all available authors from the Web API and put them in a drop down or similar (4%)
2. Insert an ID for the author (1%)

Sub-part c

Create a page which lets the user view all books, with the attached author name. An example table is displayed below.

If you didn't do Part 3B, or Part 4B, show only the authors in the table.

Title	PubYear	NumOfPages	AuthorFirstNam e	AuthorLastName
Rythm of War	2020	1095	Brandon	Sanderson
Database Systems	2006	1233	Thomas	Connelly
The Dying of the Light	2014	605	Derek	Landy

Sub-part d
Implement the functionality to delete a Book.

Sub-part e
On the page that lets the user view all books, implement functionality to filter the data by the following:

- Author's first name
- Book title.

(If you didn't do Part 3B, Part 4B, the filtering should be done on the first name of the author)