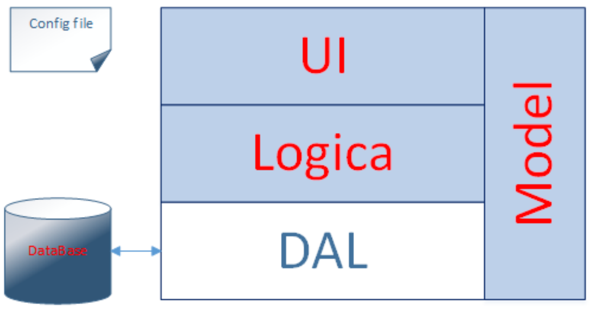
Book reservation system



As an exercise for the Database Project (term 1.3) and the Application Building Project (term 1.4), we will be working on a project in which we will create a reservation system for books. This system will record book reservations made by customers. The system will be constructed with the same layer architecture as the 1.3 and 1.4 projects.

Layer Architecture

Each layer of the architecture has specific responsibilities and contains its own specific classes. These responsibilities and classes are described below.

The Model

This layer contains the Model classes, classes that represent things in our system. In the book reservation system, these classes are Customer, Book and Reservation. We use instances of these classes (objects) in all layers of the system. Note that a Customer object always represents a customer.

Note: it is a bad habit to fill only parts of an object with data.

The Data Access Layer (DAL)

The Data Access Layer (DAL) consists of Data Access Objects (DAOs). The DAL contains a DAO class for each class from the model. This class is responsible for converting the data from the database into objects, and vice versa.

The CustomerDAO is responsible for converting Customer data from the database into Customer objects. That is all a DAO does. The DAOs are the only classes in the project that contain SQL and other database-related code. A DAO contains no intelligence/logic otherwise.

In addition to the CustomerDAO, we also have a BookDAO and a ReservationDAO in the reservation project.

The User Interface Layer (UI)

The UI layer is responsible for (partially) displaying objects in the user interface and processing user input.

The UI layer contains classes that represent a component of the user interface, such as LogInForm, ReservationsForm and SearchForm. The UI layer can also contain auxiliary classes for UI elements.

Like the DAOs, the Forms contain no intelligence. Only simple input verification is used.

The Logic Layer

The logic layer contains the actual system. This is where the business logic is performed.

The logic layer contains Service classes. In the case of simple systems, the services are organised per model class. In this book reservation system, we have a BookService, a CustomerService and a ReservationService.

The ReservationService contains, for example, a method ‘CreateReservation(Customer customer, Book book)’. This method verifies whether the customer and the book exist via the CustomerDAO and the BookDAO. If the customer and the book exist, the ReservationService creates a Reservation object. The ReservationService will use the ReservationDAO to write this Reservation to the database.

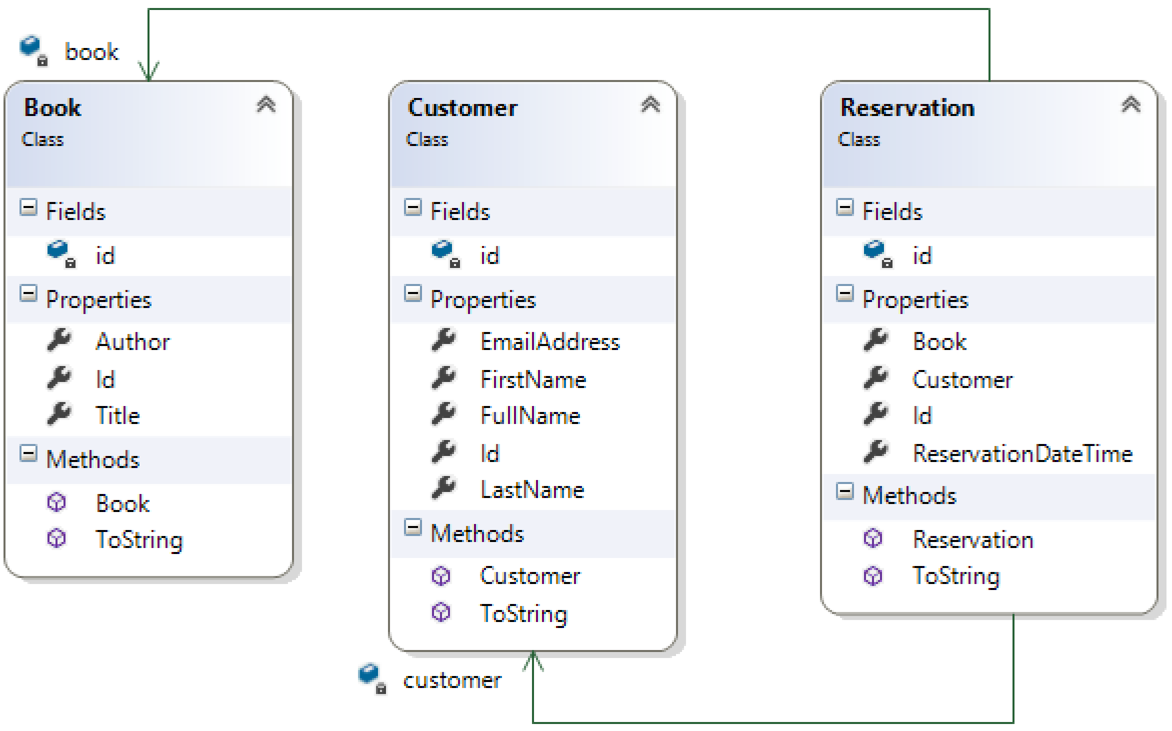
Introduction to the Book Reservation System

We will build the book reservation system in a number of steps:

1. **Model**: we will begin by creating the model classes, which we will test in a console application.
2. **DAL**: subsequently, we will create the DAL layer with the DAO classes that retrieve objects from our database. We will test this layer in our console application as well.
3. **Form** + **Logic**: for each screen in the UI Design, we will add a Form to the UI and place the required functionalities in the Logic layer. We will test the operation of our WinForm application for each functionality.

Assignment 1 – Model classes

Shown below is the class diagram of the model classes of the Book Reservation System. We will create these classes and test them using a console application.



Create a class library with the name Model for the model classes. Create the classes according to the class diagram.

Note the symbols in the class diagram. 🔧 represents a (public) property, ⬡ represents a public method, ⬡ represents a private method and ● represents a private field.

Use the following constructors for the classes:

```
public Customer(int id, string firstName, string lastName, string emailAddress) { ... }
public Book(int id, string title, string author) { ... }
public Reservation(int id, Customer customer, Book book) { ... }
```

To test your model classes, create a Console application that uses the class library Model.

Create different instances of Customer, Book and Reservation and show them using the ToString method in the console.

Assignment 2 – Data Access Layer (DAL)

The Data Access Layer contains DataAccessObject classes (DAOs). The task of a DAO class is to facilitate communication with the database.

For example, the `public class CustomerDAO` has the method `public List<Customer> GetAll()`. This method yields (returns) a list with customer objects. The implementation of this method determines how this list is created (faked or with a query to the database).

Methods that you will typically find in a DAO are:

- GetAll(), which performs a 'SELECT * FROM Table' query and converts the results of this query into a list with objects;
- GetById(id), which searches the record with the specified id and uses it to create an object;
- Store(object), which saves the specified object in the database;
- Update(object), which changes the specified object in the database;
- Delete(object), which removes the object from the database.

Try to keep this naming convertion, in order to make working together on the same code easier.

The class library DAL

☐ Add your database connection information to the App.config of your (Console) project. The information for this has been sent to you by email (or you can use the database for the Database project).

☐ Create a new class library with the name DAL. We will place the DAO classes in this class library.

The database table Customers

Before we implement the DAOs, it is useful to create a database and link it to your application.

☐ Create a table named ‘Customers’ in your database with the following fields:
Id: int
FirstName: nvarchar(100)
LastName: nvarchar(100)
EmailAddress: nvarchar(100)
Make the ‘Id’ field the Primary Key and make sure that this value is automatically raised in the event of new records (set the property ‘Identity’ to True).

☐ Populate the ‘Customers’ table with a number of records.

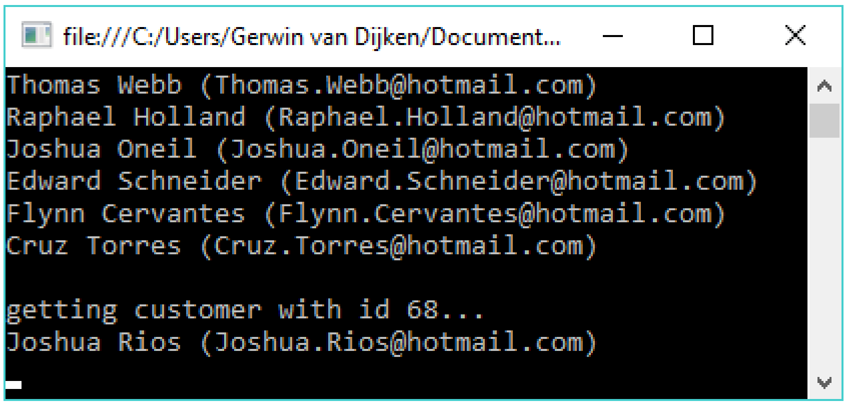
The CustomerDAO class

☐ Create the CustomerDAO class in the DAL.
Create the GetAll() method of the CustomerDAO that returns a list with Customer objects. The signature of the GetAll method is: `public List<Customer> GetAll()`

☐ Create the `public Customer GetForId(int customerId)` method as well, which creates the requested Customer object based on the data from the database.

☐ Test the methods of the CustomerDAO in your Console application.

```
CustomerDAO customerDAO = new CustomerDAO();
foreach (Customer customer in customerDAO.GetAll())
    Console.WriteLine(customer);
```



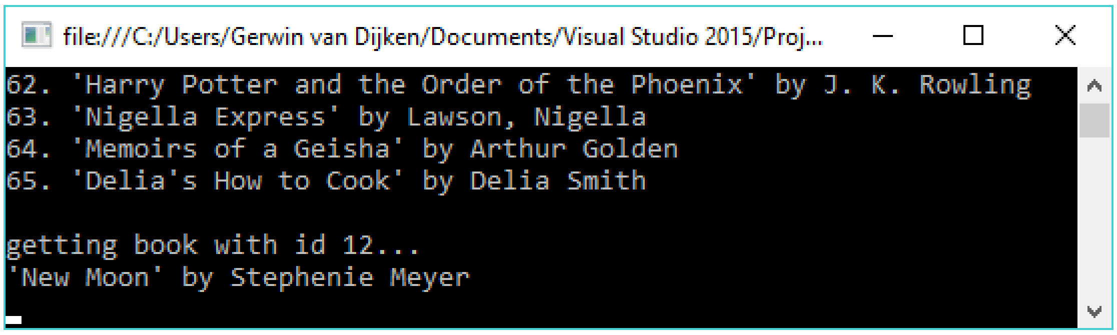
```
Thomas Webb (Thomas.Webb@hotmail.com)
Raphael Holland (Raphael.Holland@hotmail.com)
Joshua Oneil (Joshua.Oneil@hotmail.com)
Edward Schneider (Edward.Schneider@hotmail.com)
Flynn Cervantes (Flynn.Cervantes@hotmail.com)
Cruz Torres (Cruz.Torres@hotmail.com)

getting customer with id 68...
Joshua Rios (Joshua.Rios@hotmail.com)
```

The database table Books and the BookDAO

Just like the customers, the books (to be reserved) must be read from the database. This is the responsibility of the BookDAO class.

- ☐ Create a table named 'Books' in your database with the following fields:
  - Id: int
  - Title: nvarchar(200)
  - Author: nvarchar(200)

  Make the 'Id' field the Primary Key and make sure that this value is automatically raised in the event of new records (set the property 'Identity' to True).
- ☐ Populate the 'Books' table with a number of records.
- ☐ Create a BookDAO class in DAL.
- ☐ Give the BookDAO class a GetAll() method, which returns a list with Book objects from the database.
- ☐ Also, give the BookDAO class a GetById(int bookId) method, which creates the requested Book object based on the data from the database.
- ☐ Test the methods of the BookDAO in your Console application.



file:///C:/Users/Gerwin van Dijken/Documents/Visual Studio 2015/Proj...

```
62. 'Harry Potter and the Order of the Phoenix' by J. K. Rowling
63. 'Nigella Express' by Lawson, Nigella
64. 'Memoirs of a Geisha' by Arthur Golden
65. 'Delia's How to Cook' by Delia Smith

getting book with id 12...
'New Moon' by Stephenie Meyer
_
```

The database table Reservations and the ReservationDAO

Just like the customers and books, the reservations must be read from the database. This is the responsibility of the ReservationDAO.

- ☐ Create a table named 'Reservations' in the 'Reservations' database with the following fields:
  - Id: int
  - CustomerId: int
  - BookId: int

  Make the 'Id' field the Primary Key and make sure that this value is automatically raised in the event of new records (set the property 'Identity' to True). Make the 'CustomerId' field a Foreign Key to the 'Customers' table, and the 'BookId' field a Foreign Key to the 'Books' table.
- ☐ Populate the 'Reservations' table with a number of records.
- ☐ Create a ReservationDAO class in the DAL.
- ☐ Give the ReservationDAO class a GetAll() method, which returns a list with Reservation objects from the database. You must formulate your query (using a JOIN) in such a way that you also immediately get the correct customer and book data for each reservation.
- ☐ Give the ReservationDAO class a GetAllForBook(Book book) method as well, which retrieves a list with reservations from the database for the specified book. Method signature:
  ```
  public List<Customer> getAllForBook(Book book)
  ```
- ☐ Give the ReservationDAO class a GetAllForCustomer(Customer customer) method as well, which retrieves a list with reservations from the database for the specified customer. Method signature:
  ```
  public List<Book> getAllForCustomer(Customer customer)
  ```
- ☐ Test the methods of the ReservationDAO in your Console application.