Programmeren 3

Programma periode 1.3

01 (wk-05)	classes / constructors / this / static
02 (wk-06)	inheritance / override methods / abstract classes
03 (wk-07)	access modifiers / properties
04 (wk-08)	vakantie
05 (wk-09)	database access / database layer
06 (wk-10)	User Interface / UI + service layer
07 (wk-11)	customizing UI
08 (wk-12)	oefententamen

---

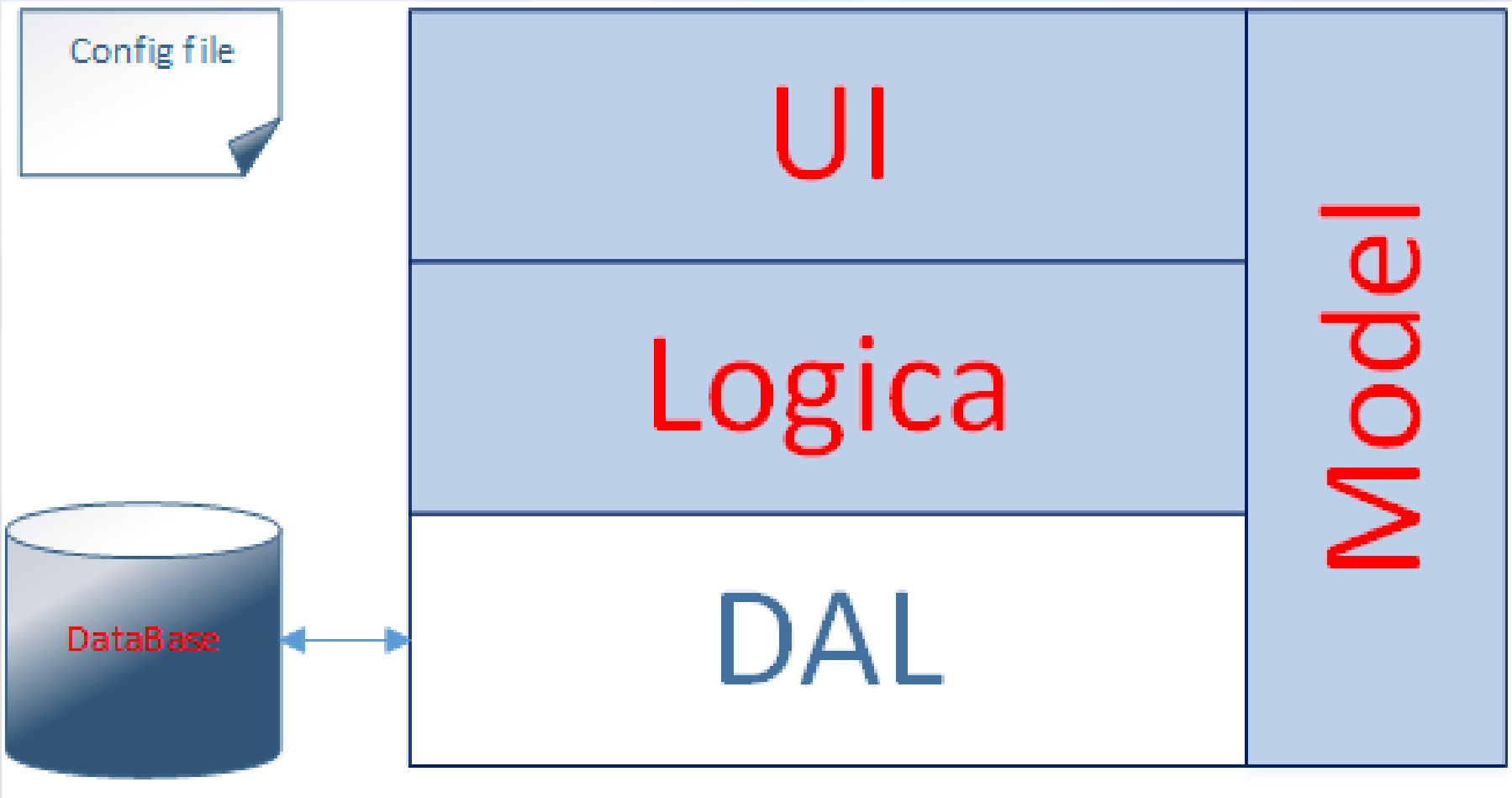
09 (wk-13)	*tentamens*
10 (wk-14)	*hertentamens*

Programmeren 3 – week 4, 5 en 6

- Informatie voor project Database (1.3)
  - Gelaagde architectuur
  - Model classes
  - Database toegang
- Informatie voor project Applicatiebouw (1.4)
  - Userinterface

gelaagde architectuur

Gelaagde Architectuur



Lagen: Model

- 'Model' laag bevat Model classes
  - representeren de dingen in het systeem
  - Model-objecten worden in alle lagen gebruikt
  - vb: Customer, Book, Menu, Employee, …

Lagen: Data Access Laag (DAL)

- bevat Data Access Objecten (DAO's)
- voor elke model-class een DAO-class
- DAL is verantwoordelijk voor het omzetten van data uit de database naar objecten, en vice versa

- enige plaats waar SQL wordt gebruikt!!

- vb: CustomerDAO, BookDAO, MenuDAO, EmployeeDAO, …

Lagen: User Interface Laag (UI)

- bevat Windows Form (WPF, …) classes
- verantwoordelijk voor het (gedeeltelijk) zichtbaar maken van objecten en verwerken van gebruikersinvoer
- enige plaats waar UI-componenten worden gebruikt!!
- vb: LoginForm, CustomerForm, SearchForm, PaymentForm

Lagen: Logica Laag

- De logica laag bevat het eigenlijke systeem
- Het is de plaats waar de bussiness logica wordt uitgevoerd
- De logica laag bevat Service-klassen
- Bij eenvoudige systemen zijn de services georganiseerd per model-klasse

- vb: CustomerService , BookService, ReservationService, …

Deze week

- Model
- DAL
- Console-applicatie om Model+DAL te testen

Volgende week

- Logica Laag
- UI Laag
- Windows Forms applicatie

Laatste week

- UI Laag: Style Guide m.b.v. overerving

Model laag

Model classes

- Classes zoals Book, Customer, Programmer, Team, PlayingCard, uit de opgaven

Voorbeeld: Customer (Model)

```
public class Customer
{
    private int id;
    public int Id { get { return id; } set { id = value; } }

    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string EmailAddress { get; set; }

    // calculated property
    public string FullName { get { return $"{FirstName} {LastName}"; } }

    public Customer(int id, string firstName, string lastName, string emailAddress)
    {
        // ...
    }

    public override string ToString()
    {
        return $"{FullName} ({EmailAddress})";
    }
}
```

Database laag

Voorbeeld: CustomerDAO

```
public class CustomerDAO
{
    private SqlConnection dbConnection;

    public CustomerDAO()
    {
        string connString = ConfigurationManager
                                .ConnectionStrings["DBConnectionString"]
                                .ConnectionString;
        dbConnection = new SqlConnection(connString);
    }

    public List<Customer> GetAll()...

    public Customer GetById(int customerId)...

    private Customer ReadCustomer(SqlDataReader reader)...
```

Voorbeeld: CustomerDAO

```
public List<Customer> GetAll()
{
    dbConnection.Open();
    SqlCommand command = new SqlCommand("SELECT * FROM Customers", dbConnection);
    SqlDataReader reader = command.ExecuteReader();
    List<Customer> customers = new List<Customer>();
    while (reader.Read())
    {
        Customer customer = ReadCustomer(reader);
        customers.Add(customer);
    }
    reader.Close();
    dbConnection.Close();

    return customers;
}
```

Voorbeeld: CustomerDAO

```
public Customer GetById(int customerId)
{
    dbConnection.Open();
    SqlCommand command = new SqlCommand(
       "SELECT * FROM Customers WHERE Id = @Id", dbConnection);

    command.Parameters.AddWithValue("@Id", customerId);

    SqlDataReader reader = command.ExecuteReader();
    Customer customer = null;
    if (reader.Read())
    {
        customer = ReadCustomer(reader);
    }
    reader.Close();
    dbConnection.Close();

    return customer;
}
```

Voorbeeld: CustomerDAO

```
private Customer ReadCustomer(SqlDataReader reader)
{
    // retrieve data from all fields
    int id = (int)reader["id"];
    string firstName = (string)reader["FirstName"];
    string lastName = (string)reader["LastName"];
    string emailAddress = (string)reader["EmailAddress"];

    // return new Customer object
    return new Customer(id, firstName, lastName, emailAddress);
}
```

UI laag

Gebruik/testen van CustomerDAO

```
void Start()
{
    CustomerDAO customerDAO = new CustomerDAO();

    // display all customers
    List<Customer> customers = customerDAO.GetAll();
    foreach (Customer cust in customers)
    {
        Console.WriteLine(cust);
    }

    // display a specific customer
    Customer customer = customerDAO.GetById(2);
    if (customer != null)
    {
        Console.WriteLine(customer);
    }
    else
    {
        Console.WriteLine("Customer not found");
    }
}
```

Huiswerk voor volgende week

- Bestudeer de aangegeven paragrafen uit het ‘Yellow Book’ (zie Moodle)
- Week 4 opdrachten *(deel 1 van ‘reservation system’)* (zie Moodle)