Object-oriented programming - lab in .NET environment

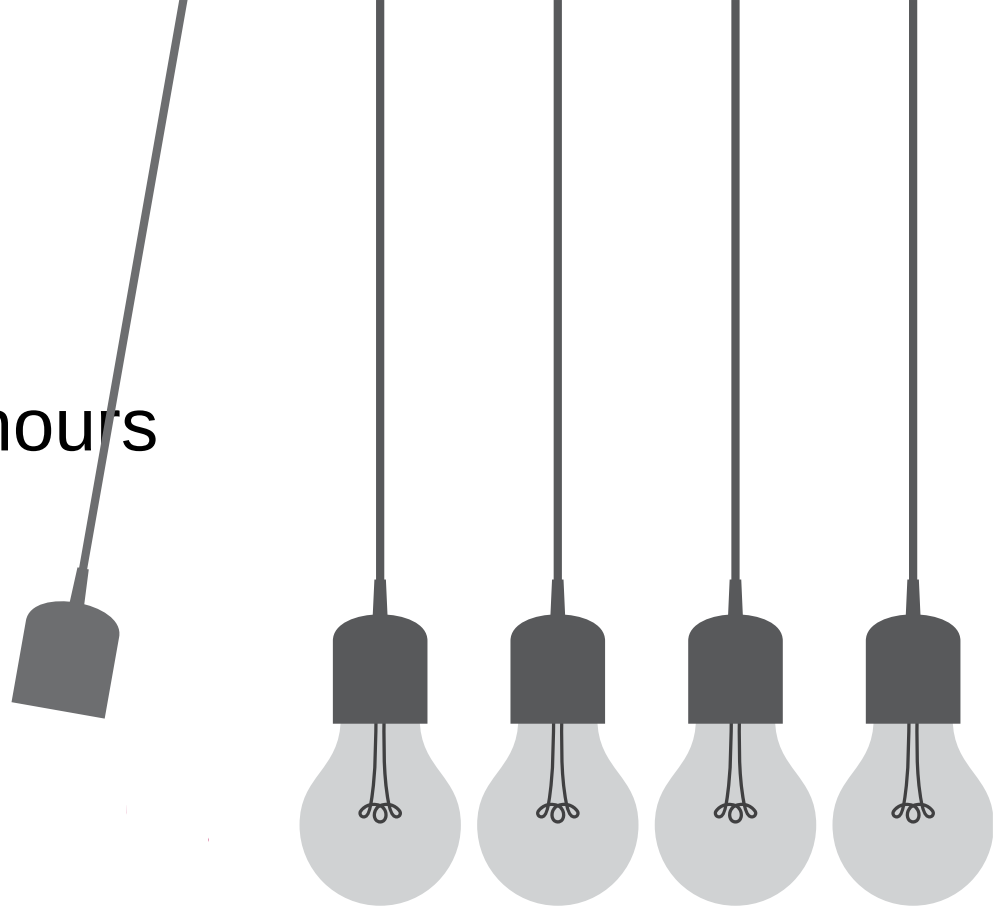
Lecture 01

Organization of lectures and practical classes

Course coordinator:	Đani Pašić	dani.pasic@algebra.hr
	Silvije Davila	silvije.davila@algebra.hr
Conducting classes:	Lectures	1 hour per week: - Every odd week Tuesday - 15 hours total
	Practical classes	3 hours per week - By groups according to the schedule - 45 hours total

Course information

- 5 ECTS credits = 150 student working hours
  - 15 hours of lectures
  - 45 hours of practical classes
  - 90 hours of working from home
- Obligatory course



Course objective

The student will become familiar with methods and technologies programming Windows applications in the .NET object-oriented environment.

Learning outcome sets

Label of a learning outcome set	Name of a learning outcome set	ECTS credits
S1	The fundamentals of creating desktop applications	3
S2	The fundamentals of creating affluent graphic user interface in desktop applications	2

Learning outcomes

Set	Outcome	MINIMUM LEARNING OUTCOMES (upon succesful completion of the course, students will be able to)	DESIRED LEARNING OUTCOMES (succesful students should be able to)
S1	I1	Construct a desktop solution using standard components.	Construct a desktop solution using advanced components.
	I2	Recommend ways of improving and implementing user experience in desktop applications.	Ensure data storage and implement user interface desktop applications that will render based on the data storage.
	I3	Recommend ways of constructing and implementing the parent-child relationship in desktop applications.	Achieve advanced implementation of the parent-child relationship in desktop applications for improving user experience.
S2	I4	Implement an appropriate desktop application architecture.	Implement a responsive desktop application by using appropriate architecture.
	I5	Apply animations for achieving better user experience in desktop applications.	Implement dynamic rendering of the user interface desktop application based on the retrieved data.

Thematic units of the course

Course week	Unit	Course week	Unit
Week 1	**Introduction to the course. Introduction to Windows Forms.**	Week 9	
Week 2		Week 10	
Week 3	**Container controls. Dialog forms.**	Week 11	**Introduction to Windows Presentation Foundation (WPF).**
Week 4		Week 12	
Week 5	**Using web services from Windows Forms applications. Asynchronous operation in Windows Forms applications.**	Week 13	**Arrangement of elements in general. Arrange panels using Panels.**
Week 6		Week 14	
Week 7	**User controls. Globalization and localization. Print controls.**	Week 15	**Resources, animations and styles. Localization. User controls.**
Week 8			

Literature

Official literature

- Beerbohm, M. (2019) Visual C#.NET: Windows Forms Programming with C#. Independently published.
- Yuen, S. (2020) Mastering Windows Presentation Foundation: Build responsive UIs for desktop applications with WPF. 2nd edn. Birmingham: Packt Publishing.

Recommended literature

- Nathan, A. (2010) WPF 4: Unleashed. Carmel: Sams.

What is necessary to get a signature?

In order to obtain the right to a signature, it is necessary to participate in class at the percentage rate prescribed by the Book of Regulations on studies and studying.

Lectures and practical classes participation	
At least 50 % of physical presence in lectures	At least 60 % of physical presence in practical classes

Whoever fails to obtain a signature will have to enroll in the same course the following year, to pay the enrollment and does not have the right to take the exam.

Passing courses

- A course has 5 defined learning outcomes divided into 2 learning outcome sets.
- **In order for students to pass a course, they need to achieve at least 50% of credits of the total credit amount within each of the learning outcome**.

How does this relate to learning outcomes

Set	Outcome	MAX (Project assignment)
S1	I1	20
	I2	20
	I3	20
S2	I4	20
	I5	20
	Total	100

Grading

Number of points achieved	Grade
0,00 – 50,00	1 (insufficient)
50,01 – 58,00	2 (sufficient)
58,01 – 75,00	3 (good)
75,01 – 92,00	4 (very good)
92,01 – 100,00	5 (excellent)

Exams

- Each course complies with the **3 + 1 rule**.
  - This means that a student can take an exam a maximum of 4 times.
    - 3 regular exams – included in the tuition fee
    - 1 extraordinary exam – 4th registration of exam is charged extra, pursuant to the Decision on Reimbursement of Expenses
  - The deadline for passing an exam is **12 months** from the day of enrolment in the course.
  - If a student fails to pass a course within 12 months, **he/she must re-enrol in the course and re-take all learning outcomes defined in the course**.
- **Keep track of deadlines for registering and cancelling exams on IE.**
  - If you failed to register an exam on time, you cannot present your project assignment.
  - If a student registers for multiple examination periods of the same course, after obtaining a satisfying grade, he/she must cancel his/her registration for all subsequent examination periods of that course. Otherwise, an insufficient (1) will be recorded in Infoeduka for that student.

Project asignment...

- On Infoeduka under Course materials -> OOP - lab in .NET environment -> Other course documents:
  - UCA-OOPNET-Project_Asignment.pdf
  - worldcup-sfg-io.zip

Academic standard of conduct

- **In written and oral communication it is necessary to follow the rules of business communication appropriate for the academic level.**
- **It is necessary to abide by the strictly defined deadlines for task submissions (project).**
  - **Every project submitted after the defined deadline will not be evaluated nor graded.**
- **Only those students who can confirm their attendance, will be considered as present.**
  - **Signing other students, or registering their card is not allowed and may be subject to disciplinary action. The teacher will delete the student's attendance if they determines that the student is registered and is not present at the class.**

Rules of conduct during classes

- One has to come to class on time.
- Upon entering the classroom, student registers for classes with a card and then sits in an accessible place for work.
- Disruption of class and inactive class participation is not allowed.
  - Continuous breaking of this rule is sanctioned by reporting students to the Disciplinary Board.

Introduction to Windows Forms

Windows Forms

- A set of classes, controls, and tools for **developing Windows applications**
- Part of the .NET framework
- Emphasis on rapid development (RAD - rapid application development)

Assistive technologies

- Graphic Device Interface + (**GDI+**)
  - **API** for drawing graphical interface

- ADO.NET and Entity Framework for data access

An alternative to Windows Forms applications

- Windows Presentation Foundation (**WPF**) applications
  - Emphasis on graphic capabilities
  - Instead of GDI+, they use **DirectX** for rendering

Visual Studio

- Integrated development platform
  - It is used for the development of all types of applications

- Contains several useful tools:
  - **Visual designers** for Windows forms with drag-and-drop controls
  - **Code-aware editors** that include command completion, syntax checking, and other IntelliSense features
  - **Integrated translation and error detection**

Solution architecture

- The resulting element in the creation of applications in Visual Studio is called a solution.
- Each solution contains one or more projects.
- Converting a project to executable code:
  - **Build -> Build Solution** creates executable code
  - **Debug -> Start Debugging** creates executable code and runs the application in debug mode
  - **Debug -> Start Without Debugging** creates executable code and runs the application

Parts of the working environment

- The working environment of Visual Studio consists of a number of parts, the most important of which are:
  - Middle part:
    - Frame **Design**
    - Frame **Code**

  - **Solution Explorer** (Ctrl + Alt + L)
  - **Properties** (Alt + Enter ili F4)
  - **Toolbox** (Ctrl + Alt + X)

Forms

- **Forms are the basic building blocks of the user interface in desktop applications**
- A form is a **container** that contains **controls**:
  - It enables the application to be presented in a familiar and consistent way
  - It can receive user input via keyboard or mouse
  - It can display data in its controls
- Simpler applications use one, and more complex applications use several forms
- The appearance of the form is defined in the Design frame, and form instances are created and displayed at run-time

Class Form

- All form functionalities are implemented in the built-in **Form** class
- When we add a new form to the project, we inherit that class
  - The class is **partial** because it is defined in two files:
    - **.Designer.cs** is used by the designer and we do not write anything there
    - We use **.cs** and the designer does not write anything there

Windows form properties

- We have a number of properties available to define the appearance and behavior of the form:
  - Using the **Properties** box
  - In code using the **IntelliSense** tool

- For example:
  - We can change the name of the form in the designer, adjust the **Text**, **Size**, **StartPosition** properties
  - We can change the **BackColor** property through code
    - We can set it either to a named color or use **Color.FromArgb()**

Working with controls

- We add controls to the form:
  - From **Toolbox**
  - Through the code
- A form keeps all controls in the **Controls** collection
- Whatever is there, the form will outline
- For example, we can add two buttons to the form and adjust their **Text**, **Size**, **BackColor**, **Location**:
  - One from the Toolbox
  - Others through code:

```
Button btnClose = new Button();
this.Controls.Add(btnClose);
```

Work with events

- Forms and controls can "trigger" a series of **events**
- We **subscribe** to those that interest us
  - We define the methods that we want to be called when the event occurs
- For example, a subscription to a Click event:
  - Through the Properties (Events)
  - Or through code

```
public Form1()
{
    Button btn = new Button
    {
        Text = "Close"
    };
    btn.Click += Btn_Click;
    this.Controls.Add(btn);
}

private void Btn_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```