Title and Code of Course: Introduction to programming using C#

Instructor's Name: Ivan Devosa Ph.D.			
Instructor's Email Address: devosa.ivan@kre.hu			
Credit Point Value: **6**	Number of Lessons per Week: **2**	Type of Course: **Seminar** ☒ **Lecture** ☐	Method of Evaluation: **Oral Examination** ☐ **In-Class Group Presentation** ☐ **Other** ☒

Course Description:

1. Introduction to module (deadlines, used programming environments etc.)

2. Introduction to C# Applications

Writing simple C# applications.

Input and output statements.

C#'s primitive types.

Arithmetic operators.

The precedence of arithmetic operators.

Decision-making statements.

Relational and equality operators.

3. Conditional Statements

Use the "if, if...else" switch selection statements to choose among alternative actions.

Using the "for" repetition statement to execute statements in a program repeatedly.

Using counter-controlled repetition and sentinel-controlled repetition.

Using the compound assignment, increment and decrement operators.

Using primitive data types

4. Loop Statements

Using the "while" and "do...while" repetition statements to execute statements in a program repeatedly.

Understanding multiple selection using the switch selection statement.

Using the "break" and "continue" program control statements to alter the flow of control.

Using the logical operators to form complex conditional expressions in control statements.

5. Methods

How static methods and fields are associated with an entire class rather than specific instances of the class.

Using common mathematical methods available in the C# API.

Understanding the mechanisms for passing information between methods.

How the method call/return mechanism is supported by the method-call stack and activation records.

How packages group related classes.

How to use random-number generation to implement game-playing applications?

How the visibility of declarations is limited to specific regions of programs.

What method overloading is and how to create overloaded methods?

Writing and using recursive functions, i.e., functions that call themselves.

6. Arrays

Using arrays to store data in and retrieving data from lists and tables of values.

Declaring arrays, initializing arrays and referring to individual elements of arrays.

Using the enhanced for statement to iterate through arrays.

Passing arrays to methods.

Declaring and manipulating multidimensional arrays.

Writing methods that use variable-length argument lists.

Reading command-line arguments into a program.

7. Introduction to Classes and Objects

What are classes, objects, methods and instance variables are?

How to declare a class and use it to create an object?

How to declare methods in a class to implement the class's behaviours?

How to declare instance variables in a class to implement the class's attributes?

How to call an object's methods to make those methods perform their tasks?

The differences between instance variables of a class and local variables of a method?

How to use a constructor to ensure that an object's data is initialized when the object is created?

The differences between primitive and reference types.

8. Classes and Objects

Encapsulation and data hiding.

The notions of data abstraction and abstract data types (ADTs).

Using keyword "this".

Using static variables and methods.

Importing static members of a class.

Using the enum type to create sets of constants with unique identifiers.

Declaring enum constants with parameters.

Organizing classes in packages to promote reuse.

9. Object-Oriented Programming: Inheritance

How inheritance promotes software reusability.

The notions of superclasses and subclasses.

Using keyword extends to create a class that inherits attributes and behaviours from another class.

Using access modifier protected to give subclass methods access to superclass members.

Accessing superclass members with super.

How constructors are used in inheritance hierarchies.

The methods of class Object, the direct or indirect superclass of all classes in C#.

10. Object-Oriented Programming: Polymorphism

The concept of polymorphism.

Using overridden methods to effect polymorphism.

Distinguishing between abstract and concrete classes.

Declaring abstract methods to create abstract classes.

How polymorphism makes systems extensible and maintainable.

Determining an object's type at execution time.

Declaring and implementing interfaces.

Bibliography:

- Csallner András Erik – Devosa Iván (2010): Introduction to Algorithms and Data Structures – JGYF Kiadó, Szeged
- Larry O'Brien, Bruce Eckel (2007):Thinking in C# – ISBN 01365872221 Safari books