Is disability/medical conditions part of your life?

Get appropriate support for exams/coursework from

Equity and Social Inclusion

Call: 9360 6084
Email: equity@murdoch.edu.au

Murdoch
UNIVERSITY

ICT365

Software Development Frameworks

Dr Afaq Shah

Murdoch
UNIVERSITY

Introduction to the Unit

Murdoch
UNIVERSITY

Teaching

Unit Coordinator : Dr Afaq Shah,

Science & Computing 1.008

Lectures: 2 hours per week (Online)

Labs: 2 hours per week (245.3.062 and online)

Contacts

Preferred method of contact is by email:

afaq.shah@murdoch.edu.au

Email Subject: ICT365

Response time: 24 hours (Monday to Friday: 8.30am to 4.30pm)

Other methods:

Phone: x2801

Office: 245.1.008

Unit description

This unit aims to provide a general understanding of software development frameworks, and the practical experience and skills in using an important software development framework, with an emphasis on language interoperability, platform independence and software reuse using Microsoft .NET Framework. Topics include: Common Language Runtime, .NET Framework Class Library, C# and other .NET languages, and application packaging and deployment. It also discusses the history and background of .NET and its relationship with J2EE.

Learning Objectives

1. Demonstrate fluency in a contemporary programming language and software development framework.
2. Implement and document an object-oriented programming solution using object-oriented analysis and design techniques.
3. Evaluate and demonstrate the theory and concepts of contemporary/ industry standards programming and design in the software development life cycle.
4. Demonstrate awareness of industry standards of software development.
5. Critically appraise the use of various software development frameworks.

Learning Objectives: Assignments

1. Demonstrate fluency in a contemporary programming language and software development framework.
2. Implement and document an object-oriented programming solution using object-oriented analysis and design techniques.
3. Evaluate and demonstrate the theory and concepts of contemporary/ industry standards programming and design in the software development life cycle.
4. Demonstrate awareness of industry standards of software development.
5. Critically appraise the use of various software development frameworks.

Learning Objectives: Exam

1. Demonstrate fluency in a contemporary programming language and software development framework.
2. Implement and document an object-oriented programming solution using object-oriented analysis and design techniques.
3. Evaluate and demonstrate the theory and concepts of contemporary/ industry standards programming and design in the software development life cycle.
4. Demonstrate awareness of industry standards of software development.
5. Critically appraise the use of various software development frameworks.

Topic Guide

This module takes a student from being an advanced beginner to an ‘entry-level’ object-oriented programmer, and includes topics such as the following: software development framework, language interoperability, platform independence and software reuse.

Additionally, software analysis, design and implementation:

- Refactoring to design patterns;
- Design principles (open-closed, Liskov substitution, interface segregation, dependency inversion, single responsibility);
- Design by contract;
- Test-driven development;
- Object-oriented design; principles of abstraction; inheritance; encapsulation;
- UML (class diagrams).

Textbook 1 *(available as ebook)*

C# 7 and .NET Core: Modern Cross-Platform Development - Second Edition Paperback – March 24, 2017 by Mark J. Price (Author)

http://prospero.murdoch.edu.au/record=b2962782~S1

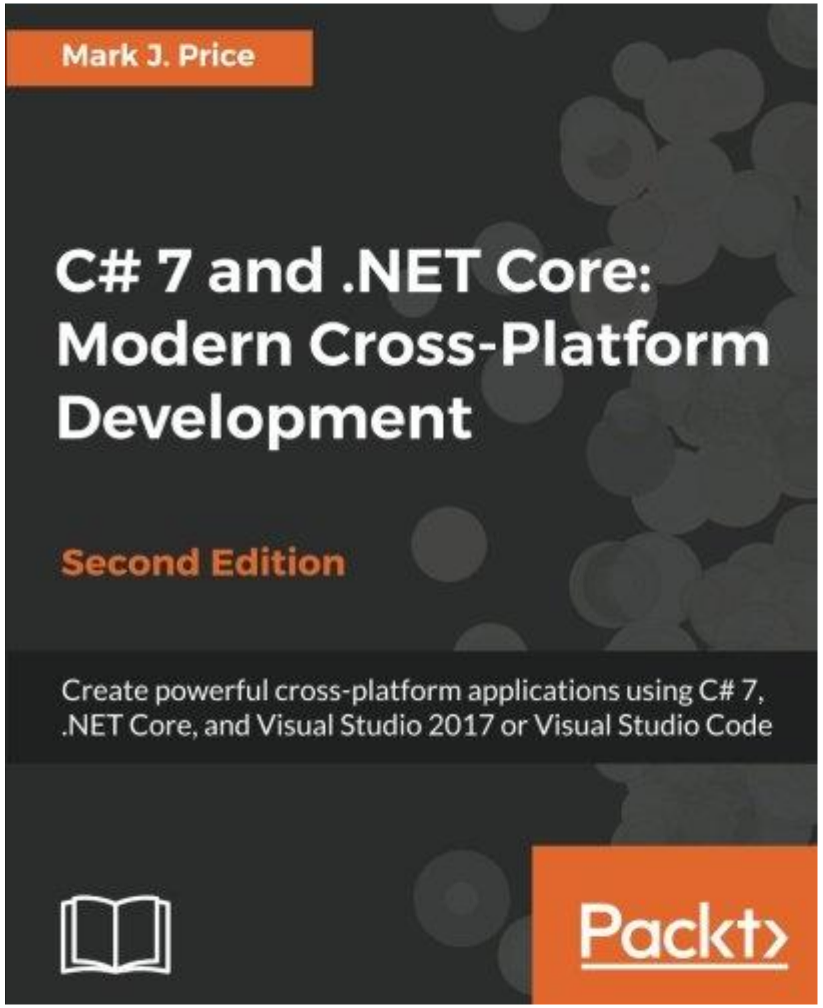
Publisher: Packt Publishing - ebooks Account; 2nd Revised edition edition (March 24, 2017)

Language: English

ISBN-10: 1787129551

ISBN-13: 978-1787129559

https://www.packtpub.com/books/content/support/27464


Mark J. Price
C# 7 and .NET Core: Modern Cross-Platform Development
Second Edition
Create powerful cross-platform applications using C# 7, .NET Core, and Visual Studio 2017 or Visual Studio Code
Packt>

Textbook 2 *(available as ebook)*

C# 6.0 and the .NET 4.6 Framework 7th ed. Edition by ANDREW TROELSEN (Author), Philip Japikse

http://prospero.murdoch.edu.au/record=b2962780~S1

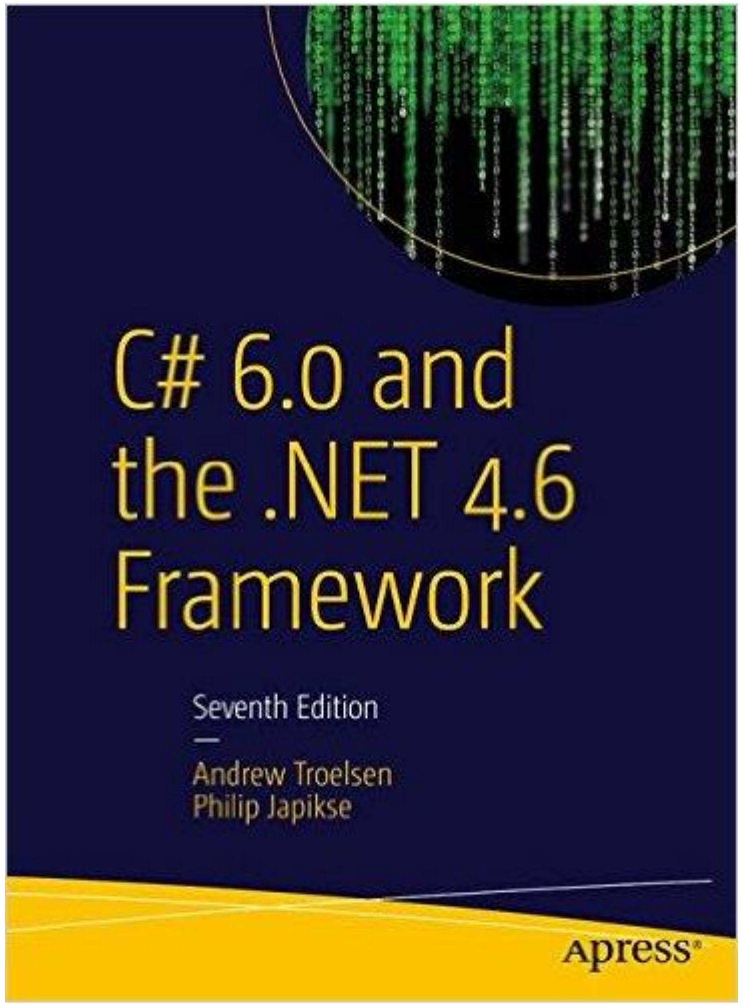
Publisher: Apress; 7th ed. edition (November 11, 2015)

Language: English

ISBN-10: 1484213335

ISBN-13: 978-1484213339

https://github.com/apress/csharp-6.0-and-.net-4.6



Textbook 3 *(available as ebook)*

C# 7 and .NET Core Cookbook - Second Edition Paperback – April 25, 2017 by Dirk Strauss (Author)

http://prospero.murdoch.edu.au/record=b2962781~S1

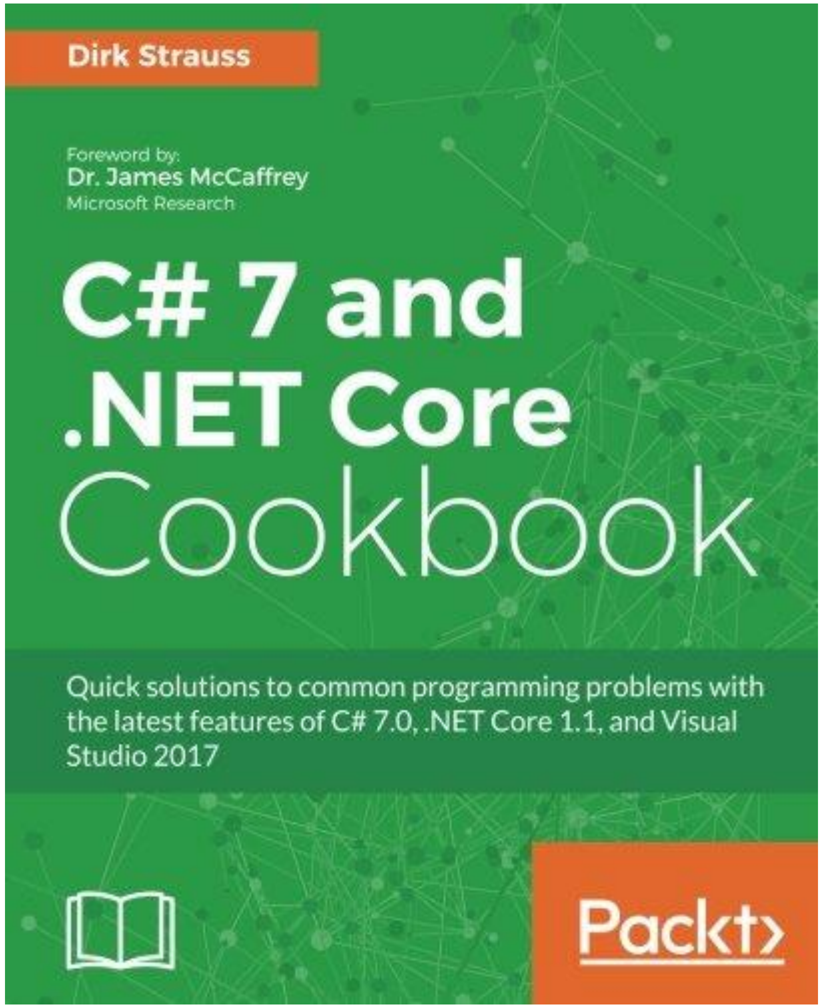
Publisher: Packt Publishing - ebooks Account (April 25, 2017)

Language: English

ISBN-10: 1787286274

ISBN-13: 978-1787286276

https://github.com/PacktPublishing/CSharp-7-and-DotNET-Core-Cookbook


Dirk Strauss
Foreword by:
Dr. James McCaffrey
Microsoft Research
C# 7 and .NET Core Cookbook
Quick solutions to common programming problems with the latest features of C# 7.0, .NET Core 1.1, and Visual Studio 2017
Packt>

Textbook 4 *(available as ebook)*

CLR via C#

Fourth Edition

By: Jeffrey Richter

Publisher: Microsoft Press

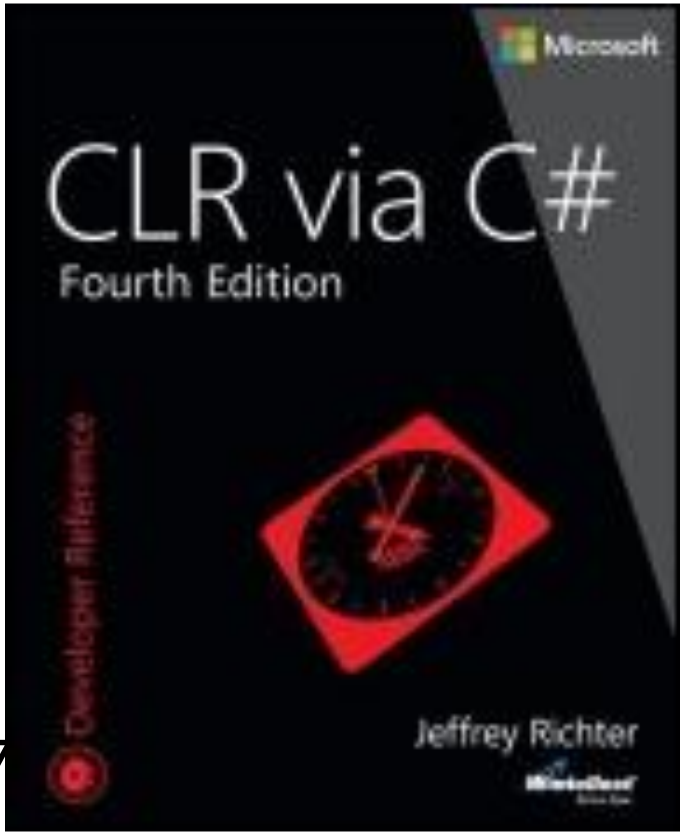
Pub. Date: November 15, 2012

Print ISBN-10: 0-7356-6745-4

Web ISBN-10: 0-7356-6873-6

Web ISBN-13: 978-0-7356-6873-7

Print ISBN-13: 978-0-7356-6745-7



Other useful books

Whenever this finally becomes available, it will be worthwhile getting a copy:

Language (C#) learning

C# 7.0 in a Nutshell - O'Reilly Media

https://www.amazon.com/C-7-0-Nutshell-Definitive-Reference/dp/1491987650/ref=pd_sbs_14_5?_encoding=UTF8&pd_rd_i=1491987650&pd_rd_r=8T3GT68YRC9Z4FTBQBZE&pd_rd_w=6L6HV&pd_rd_wg=EX7YF&psc=1&refRID=8T3GT68YRC9Z4FTBQBZE

http://shop.oreilly.com/product/0636920083634.do

Other useful books

Language (C#) learning

- C# 5.0: programmer's reference (Ebook available from Library)
- Microsoft Visual C# 2013 step by step (Ebook available from Library)
- C#: A Beginner's Tutorial (Ebook available from Library)
- C# 5 First Look (Ebook available from Library)
- C# 5.0 All-in-One For Dummies (Ebook available from Library)
- Head First C#, 3rd ed (Ebook available from Library)

Other books

Software Engineering

- Professional Test Driven Development with C#: Developing Real World Applications with TDD (Ebook available from Library)
- Design patterns in C# (Check Shelf, South St Campus South Wing Level 4, 005.133 MET 2004)

Other books

Additional features

- Beginning ASP.NET 4.5 in C# and VB (Ebook available from Library)
- Professional Windows 8 Programming: Application Development with C# and XAML (Ebook available from Library)
- Professional Cross-Platform Mobile Development in C# (Ebook available from Library)
- Professional Android Programming with Mono for Android and .NET/C# (Ebook available from Library)
- Professional iPhone Programming with MonoTouch and .NET/C# (Ebook available from Library)
- Functional programming in C#: classic programming techniques for modern projects (Ebook available from Library)

Lectures

- 11 topics plus 1 revision, approximately one topic per week (some topic may take slightly more than one week)
- Lectures: cover theory and programming
- Updated lecture notes for each topic will be posted in the Unit LMS on weekly basis
- Lecture notes will indicate the required readings
- Read both Lecture Notes and the required book chapters and online materials

Week	Topic
1	• Introduction to the Unit Overview of: • Microsoft .NET Framework • Visual Studio, C# • The CLR • Classes and Object Orientation
2	• Object Orientation • Inheritance • C# language features • Packages and Namespaces • Generics and collections
3	• ADO.net
4	• Introduction to Design patterns • The UML • Multithreading and GUI’s
5	• Design principles: SOLID • Design by contract • Platform independence and software reuse

Week	Topic
6	• Refactoring • Refactoring support in IDE/tools
7	• LINQ
8	• Collections and Generics • Exception Handling
9	• CLR • Other .NET languages • Language interoperability and platform independence • Packaging and deployment • CLR versus JVM • Visual Studio versus Eclipse/Netbeans
10	• XAML, (Cross-Platform) Mobile Development • Functional programing
11	• XAMARIN and Cross Platform Mobile Apps
12	• Unit Review and Conclusion

Labs

- There are 11 labs, starting from Week 2.
- Labs: using tools and writing programs
- The lab exercises are designed to
    - re-enforce materials in the lectures and to check on your understanding of the materials covered
    - gain necessary knowledge and programming skills incrementally for the two major assignments

Lab Assignments

- Certain lab exercises that illustrate core concepts required for the major assignments will be assessed, and the total will be worth 10%.
- These will be indicated through the LMS, and will be at approximately 3 milestone stages, for instance Weeks 4, 7 and 10.
- These assessable lab exercises are known as the *Lab Assignment* for the topic. The requirements for the assessment will be described in the lab sheet of that topic. The lab sheets will be posted in the Lab Sheets page of the Unit LMS week by week.

Lab Assignments

The deadline for each Lab Assignment is Friday (5PM) of the same week. For example, Lab 4 is scheduled in Week 4, therefore the Lab Assignment for Lab 4 is due on Friday of Week 4.

Major Assignments

There are two Major Assignments

Both Major Assignments (1&2) consist of:

- construction of a .NET solution
- theory and programming questions

The deadlines for the two major assignments will be given in the Teaching Schedule page of the Unit LMS.

Major Assignments

Major Assignment 1

concerns about the design and implementation of a .NET solution using C# and .NET Framework. The assignment will include a report style critical reflection component.

This will be a discussion of the choice of design/programming decisions made, possibly to include: **generics and collections, design patterns and principles.**

Major Assignments

Major Assignment 2

concerns about the design and implementation of a .NET solution using C# and .NET Framework. The assignment will include a report style critical reflection component.

This will be a discussion of the choice of design/programming decisions made, possibly to include: **unit testing, refactoring, comparison with VB.net, software reuse.**

Assessments

Lab Assignments: 10%

Major Assignment 1: 20%

Major Assignment 2: 30%

Final examination (closed book): 40%

Determination of Your Final Grade

Your final grade for the unit will be reported as a letter grade and a mark. In order to pass the unit you must

1. have an aggregate score for the combined assessment of 50% or better, and
2. achieve a satisfactory performance in the final examination. A satisfactory performance is normally considered to be 50% or higher.

Unit LMS

The Unit LMS contains nearly all unit materials.

- Unit Announcements
- Unit Information
- Lecture Notes
- Lab Sheets
- Assignment Questions
- Unit Readings
- Useful Links

Unit Announcements

The Unit Announcement page contains important information about the unit, particularly any changes to the deadlines, lecture and lab time or venue, cancellation of lectures or labs. I use this page to keep all students informed of what is happening about the unit.

This is the only way I will distribute unit-wide notices. I will *not* send separate notices to individual students for such unit-wide notices.

You are required to check this page regularly, at least once per week.

Unit Announcements

When an announcement has been made, we will assume that you are aware of it.

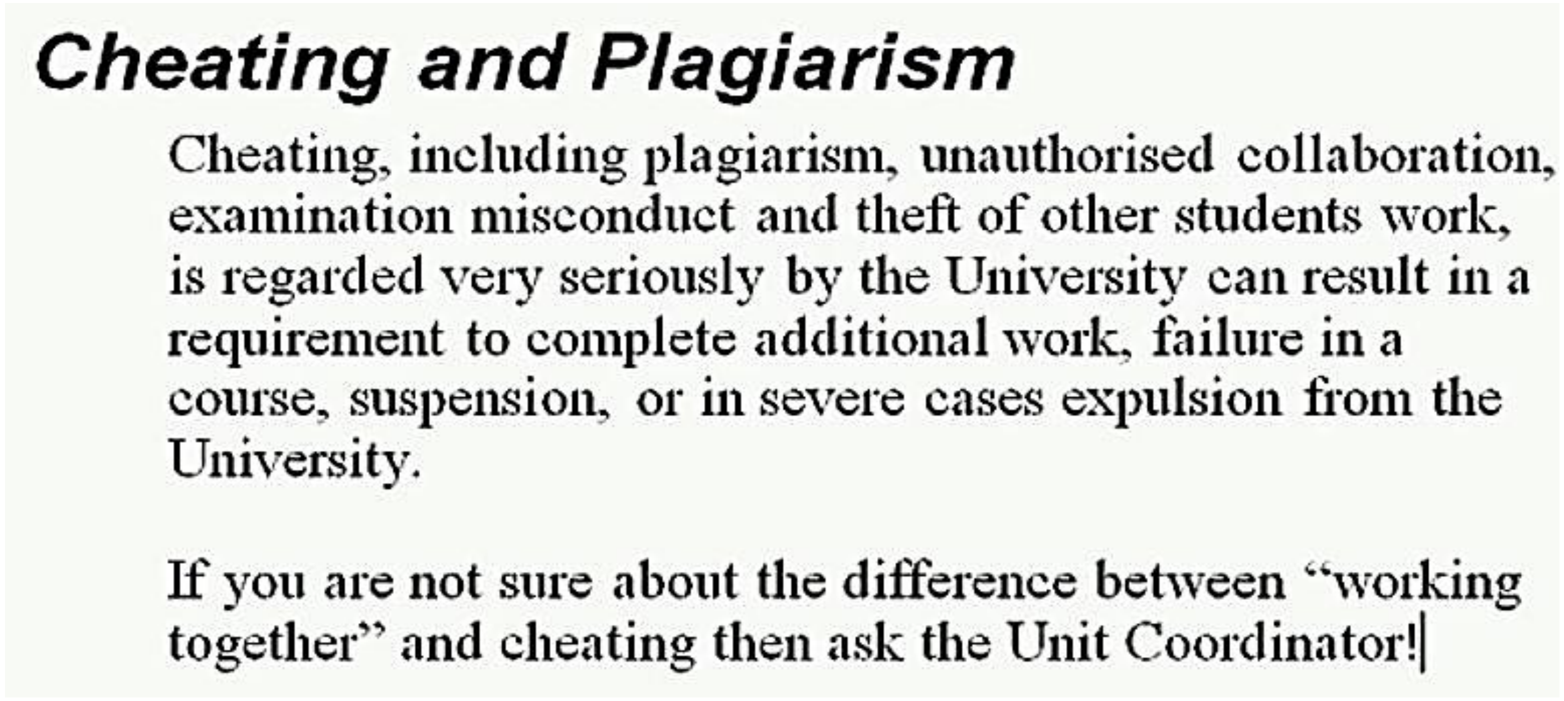
Accessing Teaching Materials

You can access lecture notes, sample programs, lab sheets, and major assignment questions from the Unit LMS.

Submission of Assignments

- The assignments must be submitted to the Unit LMS. No other forms of submission are acceptable.
- You can only submit an assignment within five days of the published deadline. Afterwards, you will not be able to submit your major assignment (we will not accept it anyway, unless there was a prior agreed arrangement with the unit coordinator).
- Assignments submitted after the due date without an extension having been granted will incur a flat 10% penalty per day late or part thereof, including weekends. Note that assignments are not normally accepted if more than five days late.

Cheating and Plagiarism



Cheating and Plagiarism

Cheating, including plagiarism, unauthorised collaboration, examination misconduct and theft of other students work, is regarded very seriously by the University can result in a requirement to complete additional work, failure in a course, suspension, or in severe cases expulsion from the University.

If you are not sure about the difference between "working together" and cheating then ask the Unit Coordinator!

What You Need to Know

The following web site contains extremely important information (including assessment and academic integrity) and you must read it carefully:

http://our.murdoch.edu.au/Educational-technologies/What-you-need-to-know/

Our Expectations

- Students enrolled in ICT365 have excellent programming skills
- Willing to put extra time and efforts
- Motivated to do research to solve programming problems

Software

Microsoft Visual Studio Professional 2017 or latest version

Available to students through the Microsoft Alliance (Dream Spark, Imagine, e5onthehub etc.)

Topic 1
Introduction to the .NET Framework
C#
Objects and classes

Software Development Frameworks

- **a collection of**
  - programming languages,
  - re-usable software components
  - a set of software tools
- **allows its users to create**
  - high quality application
    - quickly and efficiently
- **Many software development frameworks**,
  - general purpose and
  - for developing special types of applications

In this unit, we will focus on one very important general purpose software development framework - Microsoft .NET Framework.

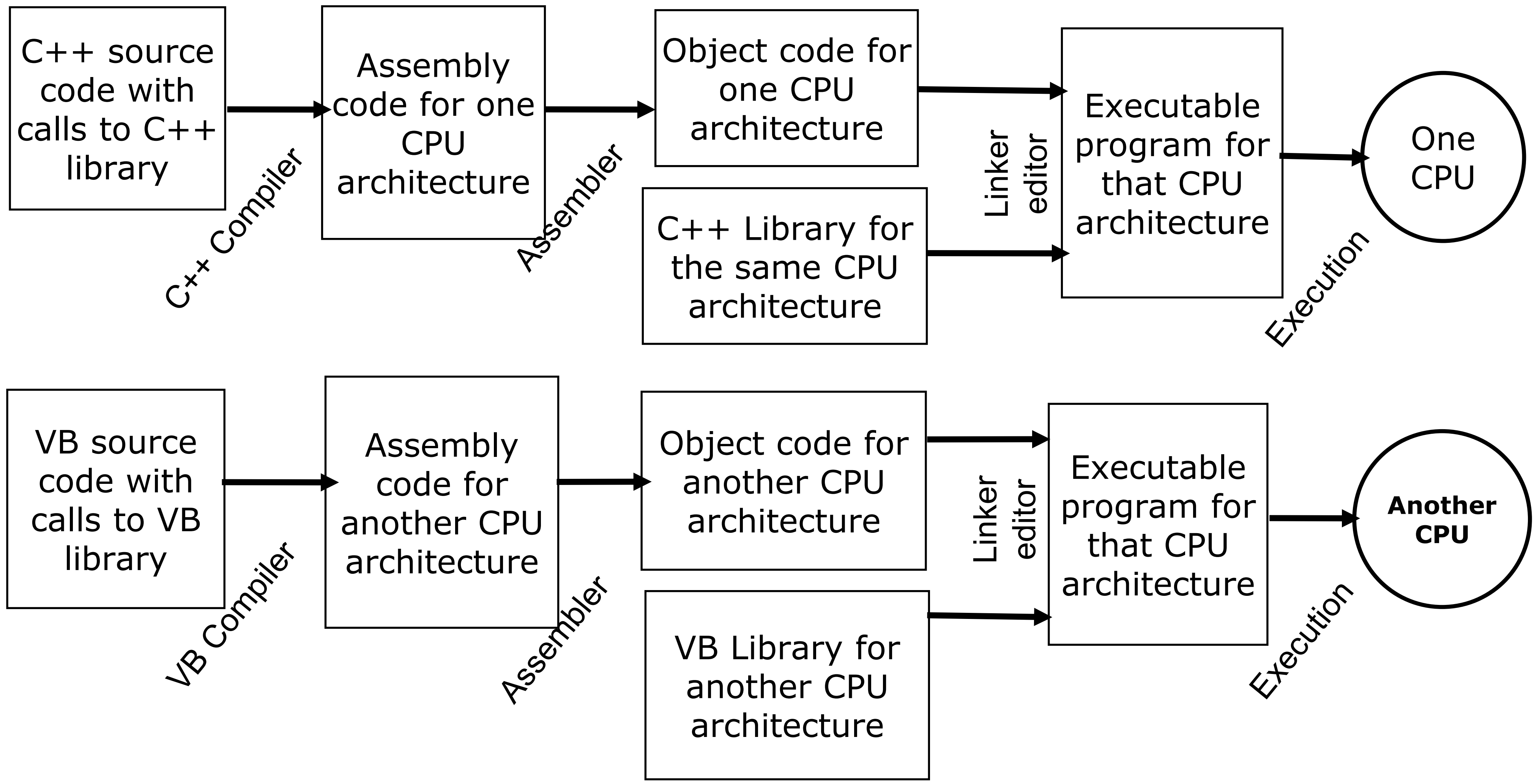
What is .NET Framework?

- .NET is the term Microsoft used to describe its (not so) new software development initiative.
- An important software development framework:
  - At the heart of it is a Common Language Runtime
  - And a set of pre-compiled class libraries
  - A set of programming languages
    - C#,
    - Visual Basic.NET,
    - C++ and
    - Visual Studio.

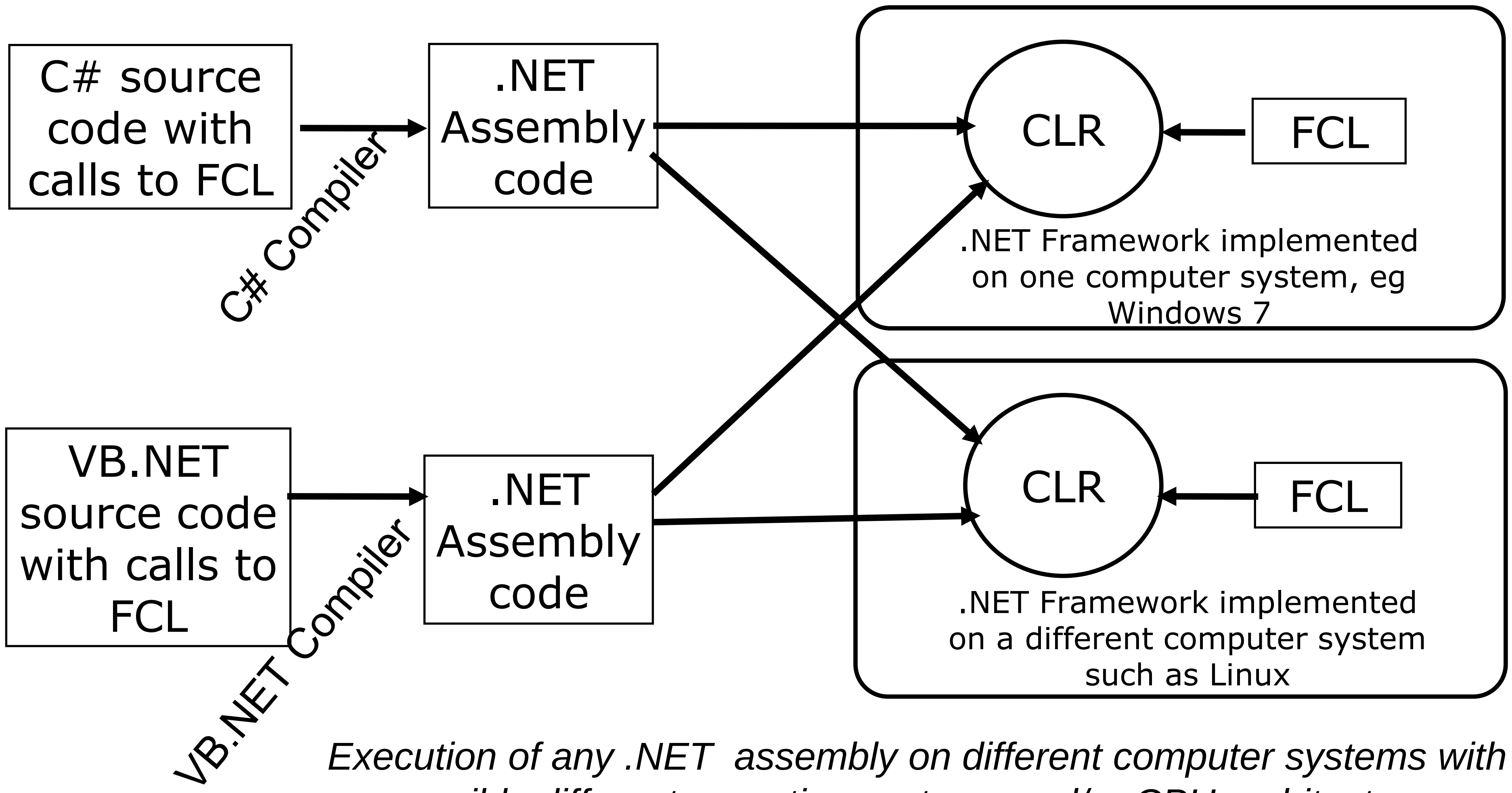
CLR

- Common Language Runtime
  - loads the .NET assembly code,
  - compiles it on the fly into the native machine code via its Just-in-Time compiler (JIT) executes it
  - also manages the memory of the running program.
- Any computer system with a CLR could run the .NET assembly program
- CLR is also called a Virtual Execution System (VES).
  - Similar to Virtual Machine in Java.

Traditional Programming Model


C++ source code with calls to C++ library
C++ Compiler
Assembly code for one CPU architecture
Assembler
Object code for one CPU architecture
C++ Library for the same CPU architecture
Linker editor
Executable program for that CPU architecture
Execution
One CPU
VB source code with calls to VB library
VB Compiler
Assembly code for another CPU architecture
Assembler
Object code for another CPU architecture
VB Library for another CPU architecture
Linker editor
Executable program for that CPU architecture
Execution
Another CPU

.NET Programming Model



C# source code with calls to FCL
C# Compiler
.NET Assembly code
VB.NET source code with calls to FCL
VB.NET Compiler
.NET Assembly code
CLR
FCL
.NET Framework implemented on one computer system, eg Windows 7
CLR
FCL
.NET Framework implemented on a different computer system such as Linux

Execution of any .NET assembly on different computer systems with possibly different operating systems and/or CPU architectures

Platform Independence

- A .NET program is compiled and deployed as an architecture independent assembly code.
- Format of the assembly - CLI standard.
- The assembly code would run on any system (Microsoft Windows, Linux, Mac OS X etc)
- At least in theory, .NET programs are platform independent.
- In reality???
- There are attempts to implement CLI and FCL on other platforms (eg, Linux and Mac OS X) such as mono project (www.mono-project.org).

First .NET Program

Here is our first .NET program using C# - the famous "Hello, world!" program:

```
// Hello.cs
//
// this is our first C# program

public class Hello
{
    public static void Main(string[] args)
    {
      System.Console.Out.WriteLine("Hello, world!");
    }
}
```

Compile and Run Program

Use a text editor such as Notepad++ to create the source code.

Save the source code into a file with .cs extension name, such as "Hello.cs".

Compile it with C# compiler csc from Command Prompt:

```
csc  Hello.cs
```

Execute the program Hello.exe by typing Hello in Command Prompt or double clicking it.

Note that csc is usually under directory:

```
c:\WINDOWS\microsoft.NET\Framework64\v4.0.30319\
```

Simple Types

Reserved word	***Aliased type***
sbyte	System.SByte
byte	System.Byte
short	System.Int16
ushort	System.UInt16
int	System.Int32
uint	System.UInt32
long	System.Int64
ulong	System.UInt64
char	System.Char
float	System.Single
double	System.Double
bool	System.Boolean
decimal	System.Decimal

Control Structures

C# provides a rich set of control structures:

- if statements
- switch statements
- while loop
- for loop
- foreach loop
- break
- continue

“if” Statements

```
if (x>0 && x<=10) {
    case = 1;
}
else if (x<=0) {
    case = 2;
}
else {
    case = 3;
}
```

“switch” Statements

```
string grade = Console.ReadLine();
switch ( grade.ToUpper() )
{
    case “P”:
        System.Console.Out.WriteLine(“Pass”);
        break;
    case “N”:
        System.Console. Out.WriteLine(“Fail”);
        break;
    default:
         System.Console. Out.WriteLine(“Supp!”);
}
```

“while” Statements

```
int n=0;
while (n<10)
{
    System.Console.Out.WriteLine(“{0}  “, n*2);
    ++n;
}
```

“for” Statements

```
for ( int i=0;  i<10;   ++i )
{
 System.Console.Out.WriteLine(“{0}   “, i*i);
}
```

“foreach” Statements

Used to iterate through an array. Syntax:

foreach (*element_type id* in *array*) { . . . }

Eg

```
string[] names = {“Peter”, “Emily”, “David”, “Nik”};

int i=0;
foreach ( string n in names )
{
 System.Console.Out.WriteLine(“Name{0}:  {1} “,   ++i,   n);
}
```

Should output:

```
Name1:  Peter
Name2:  Emily
Name3:  David
Name4:  Nik
```

“break” and “continue” Statements

`break` and `continue` statements change the control flow of loop statements.

`break` statement will jump out of the enclosing loop statement.

`continue` statement jump to the begin of the enclosing loop statement.

Classes

A C# program consists of mainly a collection of classes

A class encapsulates a set of ***members*** such as

- constants
- fields
- methods
- constructors

A class provides a blueprint for one type of objects

Fields

A field in a class defines a piece of data (can be as simple as a single integer or as complex as an object requiring mega bytes of memory)

Example:

```
string name;
uint number;
```

Methods

A method in a class defines an action. A method has a return type (or void), a signature, and a body.

A method usually acts on the data stored in its containing object.

The following methods have different signatures

```
int Add(int x, short y) { ... }
int Add(float x, int y){ ... }
int Add(int x, short y, short z) { ... }
```

Other Class Members

Constants:

Constructors:

- For use to initialise an object during object creation.
- Every class has a constructor.

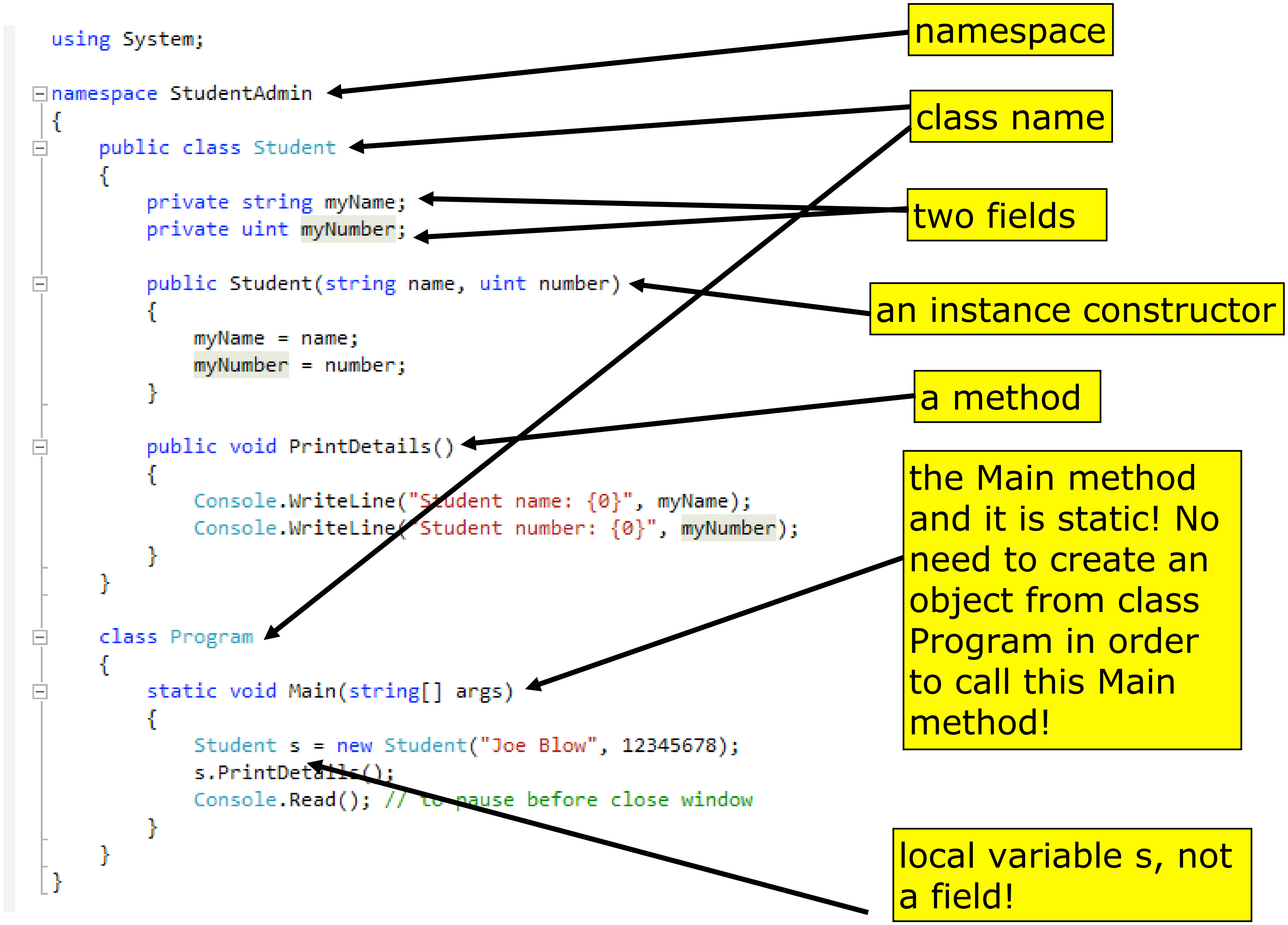
Class Example

The following program consists of two classes, Student and Program.

Assume that the class Student models a student

The class Student contains the following members:

- two fields: myName and myNumber representing a student’s name and his/her student number.
- one instance constructor which has the same name as its class and is used during the creation of a student object from the Student class
- a method PrintDetails that is used to print the student’s details.



```
using System;

namespace StudentAdmin
{
    public class Student
    {
        private string myName;
        private uint myNumber;

        public Student(string name, uint number)
        {
            myName = name;
            myNumber = number;
        }

        public void PrintDetails()
        {
            Console.WriteLine("Student name: {0}", myName);
            Console.WriteLine("Student number: {0}", myNumber);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Student s = new Student("Joe Blow", 12345678);
            s.PrintDetails();
            Console.Read(); // to pause before close window
        }
    }
}
```

namespace

class name

two fields

an instance constructor

a method

the Main method and it is static! No need to create an object from class Program in order to call this Main method!

local variable s, not a field!

A Class and its Objects

- Object oriented programming focuses on
  - modelling objects and
  - their interact with each other.
- A C# object
  - computer representation of a real-world object
- A class describes the common properties of a type of objects

Object Instantiation

- Object created from class
- The creation of an object from a class is called *instantiation* of the class
- In the previous example, we create a Student object using the “new” operator.

Student s = new Student(“Joe Blow”, 12345678);

- Invoked the method PrintDetails via object reference

s.PrintDetails();

Instance Constructor

- Used to initialise the object during the object creation.
- In class Student, the member

```
public Student(string name, uint number)
{
  myName = name;
  myNumber = number;
}
```

is an instance constructor.

- The constructor looks like a method
  - no return type
  - its name is the same as the class name.
- The constructor is used in the class instantiation:
- `s = new Student(“Joe Blow”, 12345678);`

Static Members

A class may contain static members such as static fields and static methods.

A static member can be used directly from the class without creating an object first.

A static member is qualified by the reserved word static.

A member that is not static is known as instance members

Static Field vs Instance Field

- A static field is shared by all objects of the class,
- Contrary to this, one copy of instance field is created for each object
- If a class declared one instance field and one static field,
  - you have created 10 objects from the class,
  - 10 distinct copies of the instance field, one in each object, in the memory.
  - only one copy of the static field in the memory.
  - All 10 objects will share this copy of static field.

Static Member Example

Assume all students are from the same university. We can represent the student’s university by using a static field.

We can declare a static method that will set the university value to the static field.

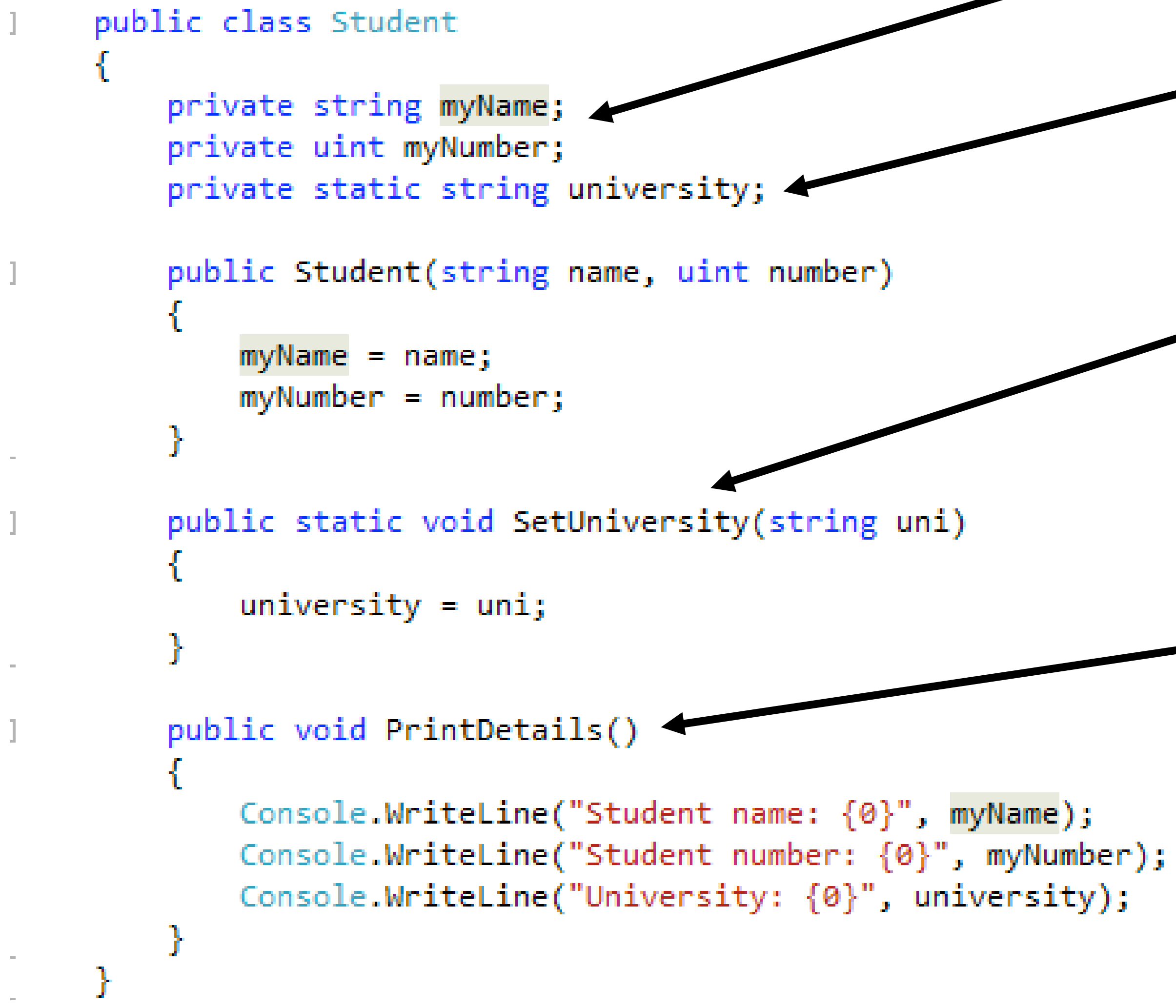
Note that a static method can only access static fields. It cannot access any instance fields (think why).

The modified code is listed in the next two slides two instance fields

a static field

a static method, note this method can only access static fields, not instance fields

an instance method, note this method can access static as well as instance fields

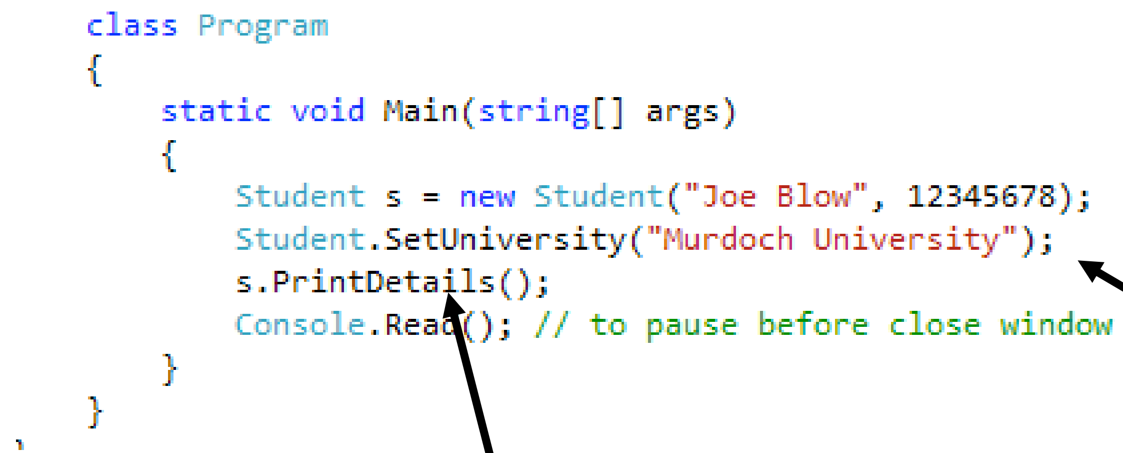


```
public class Student
{
    private string myName;
    private uint myNumber;
    private static string university;

    public Student(string name, uint number)
    {
        myName = name;
        myNumber = number;
    }

    public static void SetUniversity(string uni)
    {
        university = uni;
    }

    public void PrintDetails()
    {
        Console.WriteLine("Student name: {0}", myName);
        Console.WriteLine("Student number: {0}", myNumber);
        Console.WriteLine("University: {0}", university);
    }
}
```



```
class Program
{
    static void Main(string[] args)
    {
        Student s = new Student("Joe Blow", 12345678);
        Student.SetUniversity("Murdoch University");
        s.PrintDetails();
        Console.Read(); // to pause before close window
    }
}
```

We access the static method SetUniversity directly using the class name, not object reference!

But we cannot access this instance method using the class name. We use object reference instead.

Further Readings

Familiarise yourself with the recommended texts from the library.

Download each of them using the appropriate “reader” software.

(no need to buy them!)

What to do next week?

Look at the C# language

Object orientation