

C# For Beginners

Introduction to .NET and the .NET Framework

Contents:

- The .NET Platform
- The .NET Framework and SDK
- The .NET Building Model
- Native and Intermediate Code
- The Integrated Development Environment (IDE)

Farid Naisan, farid.naisan@mah.se

From Problem to Program

- Why program? Well, we have a problem to solve!
 - Computers need instructions.
 - Instructions are given in form of programs.
 - Programs are written in a programming language – source code.
 - Easier for programmers to write, but
 - not understandable by the computers.
 - The source code needs to be interpreted to code understandable by computers.
 - The interpreter is called “Compiler”.

Farid Naisan, Malmö University

2

Where to begin?

- Prepare your development environment.
 - Before you can start coding, you need the following tools:
 - .NET Framework SDK (Software Development Kit).
 - A text editor.
- Visual Studio (VS) integrates .NET SDK with a feature-rich code editor and many other facilities that make programming much easier and fun.
- If you install VS, that is all you need for creating C# applications.
- VS installs the SDK and sets up all environmental parameters.

Farid Naisan, Malmö University

3

The .NET Platform

- .NET (dot NET) is the name of a set of technologies developed by Microsoft for developing, deploying and executing software applications and services that use the .NET technologies.
- .NET Framework is a software framework that includes a number of programming languages, compilers, tools and all other components that are required to run a .NET application.
- .NET was introduced as single platform, the .NET Framework in 2002 and since then, it has undergone huge updates. The latest version is .NET 4.7.2.

Farid Naisan, Malmö University

4

The .NET Framework

- The .NET Framework is a software framework that helps you create desktop, web and mobile applications that run on Windows.
- The Framework is necessary to develop and run a .NET application, such as applications written in C#.
- The Framework has two major parts:
 - FCL (Framework Class Library) - a large library of classes
 - CLR (Common Language Run-time) – the run-time engine
- C# and Visual Basic are two of the several languages that come with .NET.

Visual Studio and Other IDE's

- **Visual Studio** is Microsoft's integrated development environment (IDE).
- VS is a comprehensive collection of tools and services for developing, testing, running and distributing a wide variety of applications.
- Visual Studio runs under Microsoft Windows.
- **SharpDevelop** (#Develop) is another IDE that can be used for developing C# applications for Windows.
- **MonoDevelop** is an IDE that can be used for developing C# applications on Mac OS and Linux.

Compilation and Linking

- Developing a C# program begins with writing source code and ends with producing an executable file, that is usually an EXE file or a DLL (Dynamic Link Library).
- DLL is a library that cannot run by itself but is used by another EXE or another library.
- Source code is written in files using the C# language's syntax and semantics.
- Source code needs to be compiled (converted) to code which the computer understands (0's and 1's), known as "machine code".

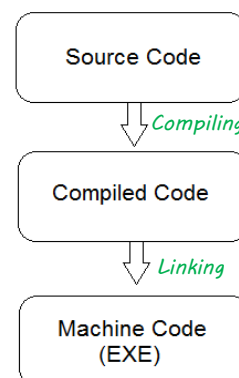


Farid Naisan, Malmö University

7

Build

- There are SDK tools that carry out compilation and linking in one step (as in Visual Studio).
- The process of Compiling and Linking is called **Build**.
- Building an application imply writing source code and producing the final product which is an executable, an EXE or a DLL.



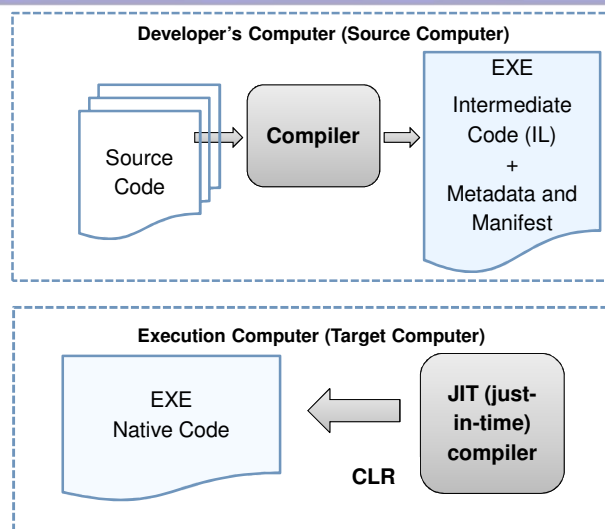
Farid Naisan, Malmö University

8

Native and Intermediate Code

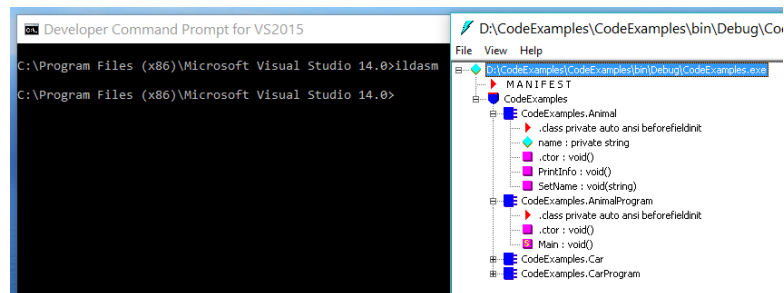
- When a compiler generates code to run on a target machine (with a certain Central Processing Unit, CPU, and a certain Operating System, OS), the code is referred to as **native** code.
- Intermediate code does not target any specific CPU or OS. It generates code targeting a “virtual machine”.
- Source code that is compiled to intermediate code uses a format that is known as **Intermediate Language (IL)**.
- IL code is CPU-independent partially compiled code.
- IL is compiled to machine code by the CLR’s JIT (Just-In-Time) compiler.

The .NET Build Model



ILDASM.EXE

- With the .NET SDK follows a tool called ildasm.exe.
- It can be run both from the command prompt and Visual Studio to get an insight look at the contents of the IL.
- ILDASM stands for IL-Disassembler and views the assembly contents for all the code components in text form.

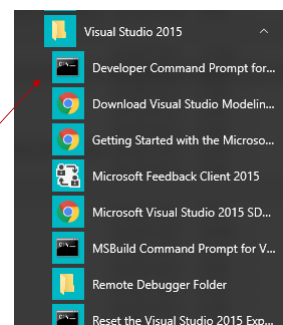


Farid Naisan, Malmö University

11

The C# Compiler

- The C# compiler is an EXE executable file, csc.exe that is a part of the SDK.
- This tool can be invoked from the Command Prompt, provided the path is adjusted.
- If you have installed Visual Studio, you can open the Visual Studio Developer Command Prompt and this has all the environmental parameters set up.
 - The Command Prompt can be loaded from the Windows Start, Visual Studio group.



Farid Naisan, Malmö University

12

Example of Compiling from Command Prompt

- To compile from Command Prompt, use CSC.exe.
- CSC has many options.
- To run the application, just write the name of the exe-file and press Enter.

```

Hello people!

C:\Program Files (x86)\Microsoft Visual Studio 14.0>d:
D:\>cd D:\CodeExamples\CodeExamples
D:\CodeExamples\CodeExamples>csc *.cs
Microsoft (R) Visual C# Compiler version 1.3.1.60616
Copyright (C) Microsoft Corporation. All rights reserved.

D:\CodeExamples\CodeExamples>program.exe
My name is: DILY!
```

Summary

- To write a C# application, you need the .NET SDK and an editor.
- Visual Studio is a very advanced IDE, Integrated Development Environment.
- You can use the Community Edition which is free for individual developers.
- C# applications use the .NET technologies included in the .NET Framework.
- The Framework contains all the software components needed to develop, test and distribute all types of applications.
- C# is part of the .NET Framework.

Programming in C# and .NET

.NET Core, .NET 5 and the future of the .NET Platform

Agenda:

- What happens to .NET Framework
- .NET Core, .NET 5, .NET 6
- Cross Platform development
- A unified development platform
- Install .NET 5

Farid Naisan, Senior Lecturer, Malmö University, farid.naisan@mau.se

.NET Framework

- .NET is a developer platform made up of tools, programming languages, and libraries for building many different types of desktop applications, websites, services and more, running on Windows.
- .NET Framework consists of the common language runtime (CLR) and the .NET Framework Class Library (FCL).
- Besides FCL, the Framework covers a broad set of Windows technologies including ASP.NET Web Forms, WCF, and more. Support for .NET Framework follows the Lifecycle Policy of the parent Windows operating system.
- The first version of .NET Framework was released on 13 February 2002. Since then many new versions have new features and usually with a new release of Visual Studio.
- The latest and last version of the .NET Framework is 4.8 which was released in April 2019. There will not be a new version of the .NET Framework. However, it will be supported as long as Windows is supported.

Farid Naisan, Senior Lecturer, Malmö University

2

.NET Framework's future

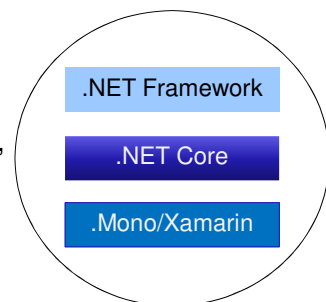


- Beginning with version 4.5.2 and later, .NET Framework is defined as a component of the Windows operating system (OS).
- .NET Framework 4.8 is the latest and the last version of .NET Framework, and no further versions will be released.
- However, .NET Framework will continue to be serviced with monthly security and reliability bug fixes.
- .NET Framework 4.5.2 and later follows the lifecycle policy of the underlying Windows OS on which it is installed.
- It will continue to be included with Windows, with no plans to remove it. You don't need to migrate your existing .NET Framework apps, but for new development, use .NET 5.0 or later.

Implementations of .NET currently in use



- **.NET Framework:** is built for Windows and is the original implementation of .NET.
- **.NET Core** is a cross-platform implementation of .NET for running websites, services, and console apps on Windows, Linux, and macOS. .NET Core is open source on GitHub.
- **Xamarin/Mono** is a .NET implementation for running apps on all the major mobile operating systems, including iOS and Android.
- .NET Framework is not cross platform as it was built for Windows and it cannot be used to develop applications that work on other platforms such as macOS and Linux.
- Microsoft has always wanted to create a unified platform that includes all the above and is cross-platform. .NET Core became the solution!



.NET Core



- As the .NET Framework was built for Windows operating system, it cannot be used to develop application work on other platforms such as macOS and Linux, i.e. .NET Framework is not cross-platform.
- Microsoft had to build a new development platform that is open source and cross-platform.
 - Most of the VMs (Virtual Machines) created on Microsoft cloud, Azure, were Linux machines.
 - Creation of .NET core became the solution.
- .NET Core is cross-platform and open source.
- .NET Core was announced in 2014 by Microsoft and the first version was release in 2016. It was developed along with .NET Framework.
- Since the first release many minor and major versions of .NET Core were release. The last version was .NET Core 3.1.
- Alongside .NET Core, .NET Framework was also improved until the last version 4.8.

Farid Naisan, Senior Lecturer, Malmö University

5

.NET Core



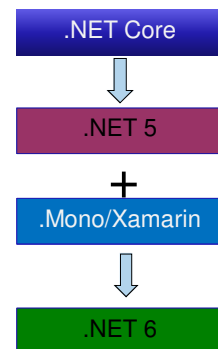
- .NET Core is a cross-platform framework to build applications for all operating systems including Windows, macOS and Linux.
- .NET Core is open-source and as of version 3 supports WPF and Windows Forms.
- The two major focus areas when .NET Core was developed were **Performance** and **Scalability**. Both .NET Core and ASP.NET Core are among the top-performing web frameworks.
- The last release of .NET Core is version 3.1 released in December 2019.
- The next major release of .NET Core 3.1 was named .NET Core 5 and it was released in November 2020. This version was announced as .NET 5 (without the word Core).
 - The main reason is to imply that .NET 5 is the one implementation of .NET that is going forward.

Farid Naisan, Senior Lecturer, Malmö University

6

.NET 5

- .NET 5 refer to several technologies including the runtime, ASP.NET Core, Entity Framework Core, and more.
- .NET 5 supports many new application types and many more platforms.
- Soon after the release of .NET 5, .NET 6 Preview 3 was released in April 2021 which integrates capabilities for Android, iOS, and MacOS that currently reside in the Xamarin open source mobile .
- As far as web development and data access platforms are concerned, the word "Core" is maintained:
 - **ASP.NET Core 5.0** is released to avoid confusion with ASP.NET MVC 5.0.
 - **Entity Framework Core 5.0** is released to avoid confusion with Entity Framework 5.0
- Both of the above are contained in .NET 5.



Farid Naisan, Senior Lecturer, Malmö University

7

.NET 5 – open source and cross-platform

- The .NET Foundation is an independent, non-profit organization established to support an innovative, commercially friendly, open-source ecosystem around the .NET platform.
- The development of .NET 5.0 and the future versions are done within the .NET Foundation.
- .NET 5 includes improvements and new features compared .NET Core 3.1 containing C#, F# and Visual Basic updates, improved APIs and many technical improvements.
- C# 9 and F# 5 are the default languages for .NET 5.
- New versions of .NET (.NET 6, 7, 8, etc.) are planned for every November.
- .NET 6 will be released in November 2021, and will be supported for three years, as a Long Term Support (LTS) release.

Farid Naisan, Senior Lecturer, Malmö University

8

.NET Unification

- .NET 5.0 was released on November 10, 2020 as another big release of the .NET developer platform.
- Microsoft has had a vision to create a unified platform that includes all frameworks and can be used across platforms and devices.
- The goal is to bring .NET Core and Mono/Xamarin together in one base class Library (BCL), integrating the capabilities for Android, iOS and MacOS
- .NET 5 is the first part of the unification and the next immediate step is .NET 6.
- .NET 6 will have improvements for cloud, desktop and mobile apps. Some features:
 - Single SDK and BCL
 - Cross platform native UI
 - Cross platform web UI
 - Cloud native

Farid Naisan, Senior Lecturer, Malmö University

9

.NET – A unified platform

- Microsoft's goal is to create one.NET that includes all the .NET platforms.
- .NET 5 is the first step towards this goal.



Source: Microsoft.com

Farid Naisan, Senior Lecturer, Malmö University

10

.NET Core 3.1 and .NET Support

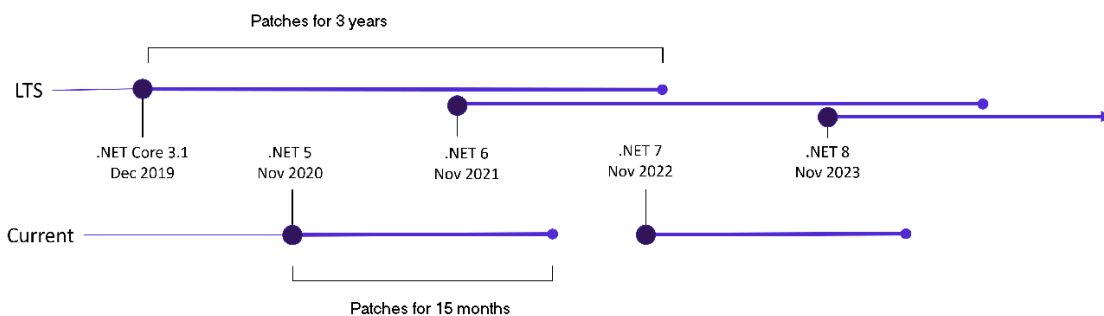
- .NET Core and future versions of .NET provide Long Term Support (LTS) releases that get three years of patches and free support.
- With the release of .NET 5, .NET Core will be outdated and when .NET 6 is released, .NET 5 will be outdated. Microsoft's support policies regarding .NET 5 and .NET core is summarized in the table below:

Version	Release Date	End of Support
.NET 5	November 10, 2020	3 months after .NET 6 release (around February 2022)
.NET Core 3.1	December 3, 2019	December 3, 2022

- .NET 5 has better memory and performance enhancements over .NET Core 3.1 - good reason for an upgrade. It also has support for WPF-

Release schedule

- Beginning in November 2020 with the release of .NET 5, major versions release every year in November. The quality of all releases are exactly the same, the only difference is the length of support. LTS (Long-term Support) releases get free support and patches for 3 years. Current releases get free support and patches for 15 months.



C# - Latest Versions



- C# is a fast developing language and is one of the most popular programming languages.
- C# together with Visual Studio provides capabilities that allow programmers to be productive writing code that is clean, fast and rich with modern features.
- C# 8.0 is the first major C# release that specifically targets .NET Core. Some features rely on new CLR capabilities, others on library types added only in .NET Core.
- With the release of .NET 5, C# 9 with a broad set of new features was also released.
- .NET 5 brought also new features in other .NET technologies including ASP.NET Core, Blazer (used for web), etc.

What disappears in .NET 5



- Blazor is a framework for building interactive client-side web UI with .NET, using C# instead of JavaScript.
- ASP.NET WebForms are used to build server-side based web applications using .NET Framework.
- Using ASP.NET Core, Blazor, Razor Pages or MVC, SPA with .NET Core backend
- WCF -> gRPC, ASP.NET Core API
- .NET Standard is also replaced by .NET 5 for most part. Microsoft recommends:
 - Use net standard 2.0 to share code between .NET Framework and all other platforms.
 - Use net standard 2.1 to share code between Mono, Xamarin and .NET Core 3.x.
 - Use net 5.0 for code sharing moving forward.

Install .NET 5

- Go to url: <https://dotnet.microsoft.com/>
- Click on the Download button, you will get to the download page as in the image below.
- Select .NET 5.0 (the first option). You will also see a message on this page that says that this release is compatible with Visual Studio 2019 v.16.9.
- Make sure to update VS to v 16.9.
- Go to Help menu in VS and select the option "Check for Updates"!



Farid Naisan, Senior Lecturer, Malmö University

15

.NET project SDKs

- .NET Core and .NET 5.0 and later projects are associated with a software development kit (SDK).
- The .NET SDK is the base SDK and is available for other SDKs that are used in a project.
- The projects do not have a part References any more and is substituted by Dependencies.
- Dependencies is a better way of managing different types of references in a more structured way showing where the references come from, SDK, NuGet or other SDKs.
- Each *project SDK* is a set of tasks that are grouped in a particular order and is responsible for compiling, packing, and publishing code.

Farid Naisan, Senior Lecturer, Malmö University

16

Summary

- .NET Framework was released 2002, supported only Windows platform.
- .NET Core was released 2016, is Cross-Platform, supports Windows, macOS and Linux.
- .NET 5 was released 2020, is cross-Platform and support more platforms, Windows macOS, Linus, iOS, Android, tvOS, watchOS, WebAssembly and more.
- .NET 5 supports several application types, Desktop, Web, Cloud, Mobile, Games, IoT, ML and AI.
- To install .NET 5, go to url: [dotnet.Microsoft.com](https://dotnet.microsoft.com).

References.

- A review of .NET towards a unified platform:
https://www.youtube.com/watch?v=t3zj0c244UE&ab_channel=kudvenkat
- .NET project SDKs:
<https://docs.microsoft.com/en-us/dotnet/core/project-sdk/overview>
- MSBuild Targets:
<https://docs.microsoft.com/en-us/visualstudio/msbuild/msbuild-targets?view=vs-2019>
- MSbuild Tasks:
<https://docs.microsoft.com/en-us/visualstudio/msbuild/msbuild-tasks?view=vs-2019>
- .NET Support Policy:
<https://dotnet.microsoft.com/platform/support/policy>

C# For Beginners

Introduction to Object-Oriented Programming (OOP)

Contents:

- Object-Orientation and Object-oriented Programming
- Classes and Objects
- Encapsulation, Inheritance and Polymorphism
- Desktop and GUI-Based Applications.
- Types of Desktop Applications
- Console, Windows Forms and WPF

Farid Naisan, farid.naisan@mah.se

Object-Oriented Programming (OOP)

- Object-oriented programming (OOP) is based on the concept of modularity and reusability.
- The basic unit is an object which is an entity made of related data and operations.
 - A car, a TV, and a child are examples.
- The data describes the status of the object at a certain point of time:
 - Color = Red, Horse Power = 178
- The operations describe their behavior, by performing tasks on the object itself using its data.
 - ChangeColorToGreen.

Farid Naisan, Malmö University

2

Class

- A class is like a form from which you can make objects.
- It is a template for objects that are similar from a perspective.
 - A car dealer registers cars with their names, plate numbers, models and other such attributes.
 - The dealer would like to be able to change the price, calculate taxes, insurance costs and sell the cars.
- An auto-repair shop would want to store a car's plate number, the owner info, the services required, and perform an engine check.
- A class defines attributes and operations for a group of objects.

Farid Naisan, Malmö University

3

Classes and objects

- When we write a C# application, we write classes to define a group of objects.
- Objects of different classes need to collaborate with each other to make a job done.
- Objects must be created before they can be used.
 - There are exceptions, for example classes with static method, abstract classes, etc..
- In C#, there is a keyword called **new** that is used to create an object of a class. Then we use the dot-notation to call its methods to perform operations.

```
Car carObject = new Car( );  
carObject.Start();
```

Farid Naisan, Malmö University

4

Object-Orientation

- Object-Orientation is an ideology, a paradigm, which can be used to model different systems.
- Object-orientation is based on three key concepts known as the pillars of object-orientation.
 - Encapsulation
 - Inheritance
 - Polymorphism
- Object-oriented programming, OOP, refers to the implementation of object-orientation in programming.

Encapsulation

- Encapsulation is to group related data and operations into an entity.
 - The entity is known as a “class” in the programming terminology.
- Encapsulation controls how the data and operations can be accessed from outside a class.
- The main purpose of encapsulation is to hide an object’s data and its operations in order to:
 - protect data from intentional and unintentional corruption,
 - hide the object’s internal implementation.

Inheritance

- The purpose of inheritance is to reuse code.
- Inheritance allows you to extend an existing class to build new classes without changing the existing code.
 - Extend an Employee class to build a Chief class.
 - The Employee class is called the “base-class” and the Chief class is called the “subclass”.
- In C#, a subclass can have only one base-class.
- All classes that you write in your applications, are automatically derived from a class called System.Object which is a part of the FCL.

Polymorphism

- Polymorphism means “one name many forms” and is, in simple terms, about making a family of related objects work in a similar way.
- Another way of describing polymorphism is to say that one object behaving in multiple forms.
 - Animal is an object that can have many forms: cat, dog, bird or butterfly.
- Polymorphism is a complicated phenomenon but is the most powerful feature of object-orientation.
 - It has been the basis of many object-oriented design patterns solving complicated but recurrent problems.

OOP's Advantages

- Model of real world
- Better communication
- Better planning and better progress measuring
- Better Budgeting
- Breaking complex problems into smaller, more controllable parts
- Code re-use
- Team work
- Effective Maintenance Management

Farid Naisan, Malmö University

9

Program, Application, App – what is the difference

- The three words are used interchangeably.
- The terms **program** and **application** are almost synonyms but **program** is perhaps a more general term.
- Programmers use the term “application” while users usually use the word “program”.
 - A BIOS program and the operating system (OS) are programs, because they are not system-dependent.
 - MS Word and Excel are applications, because they are dependent on the OS.
- An App is usually related to mobile and tablet devices, although there are apps for desktop computers too.
 - An app is usually limited to perform one or only a few tasks.

Farid Naisan, Malmö University

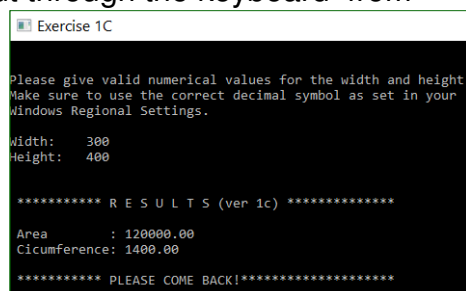
10

Types of applications

- C# can be used to create any type of applications, desktop, web and mobile applications.
- In this course, we will be working with desktop applications which are stand-alone programs running on a desktop or laptop computer.
- There are at least three ways of creating a desktop application:
 - Console application
 - Windows Forms application
 - Windows Presentation Foundation (WPF) application.

Console Applications

- A console application is a simple form of application that runs on a Prompt Window.
- It has a character-based user interface and a sequential execution flow.
- The application takes input through the keyboard from the command line and displays output by writing to the command line.



```
Exercise 1C

Please give valid numerical values for the width and height.
Make sure to use the correct decimal symbol as set in your
Windows Regional Settings.

Width: 300
Height: 400

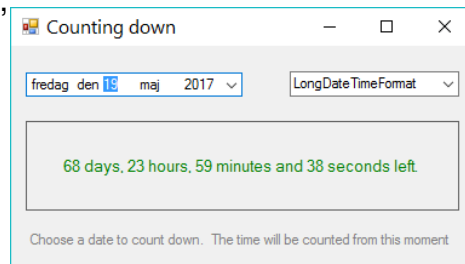
***** R E S U L T S (ver 1c) *****

Area      : 120000.00
Circumference: 1400.00

***** PLEASE COME BACK!*****
```

Windows Forms Application

- Windows Forms is a part of the .NET Framework's class library.
- It is used for creating applications with graphical user interface (GUI) for Windows.
- User interactions can take place with a large number of ready to use components, such as Textboxes, Labels, Buttons, Menus and Listboxes.

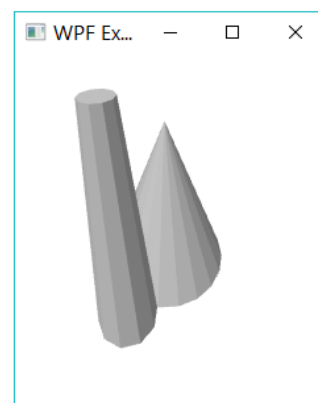


Farid Naisan, Malmö University

13

Windows Presentation Foundation (WPF)

- WPF is Microsoft's latest GUI Framework for creating desktop applications.
- It contains more advanced features, graphics, multimedia and animations.
- The main idea is to separate the design of the GUI from the logics (code-behind).
- WPF also integrates different Microsoft's technologies.



Farid Naisan, Malmö University

14

Managed Code

- Managed code is source code that requires and executes under the full management of the CLR.
- IL code is managed code. It supplies all information in form of metadata required by the CLR to manage the memory, to load and run the application correctly and safely.
- Unmanaged code usually machine-specific native code that runs directly by OS.
 - Code written in most of the old languages like C is unmanaged code.

Summary

- OOP is a methodology and is defined by three basic concepts: Encapsulation, Inheritance and Polymorphism.
- OOP is based on objects which are instances of classes.
- In this course we will be mostly working with Encapsulation but will have OOP in mind already from the beginning.
- We are going to begin working with Console applications to learn the basic C# syntax and constructs, and then go over to Windows Forms.
- Are you ready?

Programming in C# I

Introduction to classes and objects

Contents:

- Classes and objects
- C# application
- Console.Write and WriteLine
- Console.ReadLine

Farid Naisan, Farid Naisan, Malmö University, farid.naisan@mau.se

C# is object-oriented

- An object-oriented application works with objects.
- Every object belong to a class in which the main data types and operations are defined.
- To write a C# application is to write one or more classes.
- Some applications have a few number of classes but most applications have many;
 - Some large applications can have thousands of classes.
- What is the difference between a class and an object?

Farid Naisan, Malmö University, farid.naisan@mau.se

2

Classes and object

- The terms “class” and “object” are used very interchangeably, often as a matter of carelessness.
- An object is a “thing”, and a thing in object-oriented terms can be a real object, virtual or a conceptual thing,
 - person, a car, an address, a solution to a mathematical equation are some examples.

Everything is object defined by a class

- Assume that your in a grocery store. There are huge numbers of objects.
- They are categorized in different groups and each group is broken down to more categories so at you finally see products like
 - TVs, Chairs, Tables, Washing Machines, Bread Loafs, Fruit, etc.
- Are these objects or classes? The rule of thumb:
 - Abstract items are classes – they define objects
 - Every specific item is an object

An object is universally unique!

- The name TV, Chair and so are abstract; they define object groups.
- But when point at one specific TV, or one specific chair, these are objects – each one of these are universally unique! Two TVs can be exactly the same, but they are still two different objects of the same type (both are TV).
- What makes the difference?
 - The value of their data – their status.
 - One TV has S/N that is not the same as the other one.
 - Each of them have a special current channel.

Farid Naisan, Malmö University, farid.naisan@mau.se

5

The Television class

- How do we write a class for TV in C#?
- Begin with words public class and then chose a suitable name, here **Television**.
- Define some data holders to make different objects. Think of these as boxes that keep values.

```
public class Television
{
    //Data holders(also called fields, instance variables or attributes)
    private string channel;    //The channel being played
    private double frequency;  //The frequency of the current channel
    private int volume;        //The current volume
}
```

Farid Naisan, Malmö University, farid.naisan@mau.se

6

What can a television do?

- Our virtual objects are much more intelligent than the natural ones.
- They take care of themselves.
 - They can put themselves on, off,
 - change channel, and
 - view information about current channel.
- A class has therefore two main parts:
 - Data – instance variables
 - Operations, methods

Farid Naisan, Malmö University, farid.naisan@mau.se

7

The TV class with correct C# syntax

```
public class Television
{
    //Data holders(also called fields, instance variables or attributes)
    private string channel;    //The channel being played
    private double frequency; //The frequency of the current channel
    private int volume;       //The current volume

    //Operations - called methods (or functions)
    public void ChangeChannel(string newChannel)
    {
        //spara nyKanal i instansvariabeln
        channel = newChannel;
    }

    public void ShowChannelInformation()
    {
        System.Console.WriteLine("The TV is play channel " + channel);
    }
}
```

Farid Naisan, Malmö University, farid.naisan@mau.se

8

We have a class what to do next?

- To use this in a program:
 1. Create a C# application
 2. Write the save this class as a C# file, Television.cs
 3. Write a class with the method Main.
 4. Create an **object** of this class with the C# keyword. **new**.
 5. Call a method of the object.
- Assuming that you have follow steps 1 to 3, the next slide show the code for steps 4 and 5.

Farid Naisan, Malmö University, farid.naisan@mau.se

9

The main class TVProgram

```
8 //This is the first class that will be run by
9 //CLR when the program is started by the user
10 public class TVProgram
11 {
12     //And this is the very first method that
13     //will automatically be run when the
14     //program starts
15     static void Main(string[] args)
16     {
17         //Create an object of the Television class
18         Television oneTvObj = new Television();
19
20         //Use the object by calling a method
21         oneTvObj.ChangeChannel("News on XYZ");
22         oneTvObj.ShowChannelInformation();
23
24         //Let the console window stay on the screen
25         Console.ReadKey();
26     }
27 }
```

Diagram illustrating the code for the main class TVProgram, with numbered steps 3, 4, and 5 pointing to specific lines:

- Step 3 points to line 10: `public class TVProgram`
- Step 4 points to line 18: `Television oneTvObj = new Television();`
- Step 5 points to line 21: `oneTvObj.ChangeChannel("News on XYZ");`

Farid Naisan, Malmö University, farid.naisan@mau.se

10

Keywords

- Keywords in the sample program are:
 - `public`
 - `void`
 - `class`
 - `string`
 - `static`
- Keywords are reserved words in C#. They cannot be used as names for classes, instance variables or methods.
- C# is case-sensitive – the words **main** and **Main** are two different words.

Semi-colons marks end of statement

- Semi-colons are used to end C# statements.
- However, not all lines of a C# program end a statement.
- An important part of learning C# is to learn where to properly use the punctuation.
- The C# compiler will generate error when your code does not follow the language syntax (grammar).
- It creates an EXE only when the compilation can be achieved without any error.
 - You cannot test your application if the compilation is not successful.

Lines vs Statements

- There is a difference between lines and statements when discussing source code.
- A statement is a complete C# instruction that causes the computer to perform an action. This is a statement in one line.

```
double value = value + 5;
```

- This is the same statement using two lines.

```
double value = value +  
5;
```

- This is an example of two statements using one line.

```
double value = value + 5; Console.WriteLine()
```

- **Rule of thumb:** Use only one statement per line!

Parts of a C# Program

• Comments

- To comment a line of code, use `/*` as below:

```
class Product  
{  
    //Variables for storing the user's input  
    private String name;  
    private double unitPrice; //unit price  
}
```

- Compiler will ignore text that comes after the two forward slashes. Comments are used for documentation of the code.

• Namespace

- A namespace contains classes.
- The System namespace contains the basic classes needed for all C# programs
- The `using` keyword allows use of the various class methods without having to use the namespace qualifier each time.

Commenting Code

- C# provides two methods for commenting code.

Comment Style	Description
//	Single line comment. Anything after the // on the line will be ignored by the compiler.
/* ... */	Block comment. Everything beginning with /* and ending with the first */ will be ignored by the compiler. This comment type cannot be nested.

Farid Naisan, Malmö University, farid.naisan@mau.se

15

Special Characters

//	double slash	Marks the beginning of a single line comment.
()	open and close parentheses	Used in a method header to mark the <i>parameter list</i> .
{ }	open and close curly braces	Encloses a group of statements, such as the contents of a class or a method.
" "	quotation marks	Encloses a string of characters, such as a message that is to be printed on the screen
;	semi-colon	Marks the end of a complete programming statement

Farid Naisan, Malmö University, farid.naisan@mau.se

16

Console Output

- The **WriteLine** method places a newline character at the end of whatever is being printed out.
- The following lines:

```
Console.WriteLine("This is being printed out");  
Console.WriteLine("on two separate lines.");
```

would be printed out on separate lines since the first statement sends a newline command to the screen.

Console Output

- The **Write** statement works very similarly to the **WriteLine** statement. The cursor moves to the beginning of the next line.
- The difference is that Write statement does not put a newline character at the end of the output. The cursor remains at the end of the output.
- The lines:

```
Console.Write("These lines will be ");  
Console.Write("printed on ");  
Console.WriteLine("the same line.");
```

Will output:

These lines will be printed on the same line.

Console - input

- The user gives input to our console application by using the keyboard. Everything written becomes characters even numbers.
- **Console.ReadLine** is a method that can be used to read a line of text from the Console Window.
- The results are to be saved in a variable:
 - `string inputText = Console.ReadLine();`
- If the input is supposed to be a number, the input must be changed from text (characters) to a number like **int** or **double**.

Read input example

```
public void ReadNewVolumeValue()
{
    //Ask the use for new volume to put up to
    string volumeString = Console.ReadLine();

    //Use input is text such "6", "12", etc
    //must be changed to int in order to save in the
    //variable volue (which is an int)
    volume = int.Parse(volumeString);
}
```

Temporary variable holding the input string →

↖ *Convert vlumeString (ex "12") to corresponding **int** number (i.e. 12).*

Summary

.NET

- Every C# application consists mainly of classes, at least one class, but often many classes.
- A class is a definition of a group of objects.
- An object is a unique instance of a class. Every object belong to a class.
- One of the classes serve as an entry to the program.
- This class must have a method called **Main**.
- An object of a class is created by using the C# keyword **new**.

Structure of a C# Application

Contents:

- The basic structure of a C# application
- Namespace
- Members of a class, fields and methods
- The Main method
- Statements and expressions
- Access modifiers

A simple structure of a C# application

- A C# application consists at a minimum the following parts:
 - A namespace declaration
 - One or more classes
 - One class with the method Main
- A C# class has members:
 - Fields
 - Methods
 - Other members

What is a namespace?

- A namespace is a group name for a set of classes.
 - Defines a scope that contains a set of related objects.
- You can use a namespace to organize code elements and to create globally unique types.
- All code files in your application can use the same namespace.
- The `namespace` keyword followed by an identifier (ex. **CarPark**) is used to declare a namespace.
- The curly braces are a part of the syntax.

```
namespace CarPark
{
    //Class definition
}
```

A C# application with only class

- This is a complete program!
- Lines 1 and 3 are empty lines.
- Lines 4 to 8 are other namespaces that this class is referring to.
- Line 10 is the name of a namespace to which this class belongs to.

```
1
2 //AnimalProgram.cs
3
4 using System;
5 using System.Collections.Generic;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9
10 namespace AnimalMotel
11 {
12     0 references
13     class AnimalProgram
14     {
15         0 references
16         public static void Main ( )
17         {
18             Console.BackgroundColor = ConsoleColor.Gray;
19             Console.Clear ( );
20             Console.ForegroundColor = ConsoleColor.DarkBlue;
21
22             Console.Title = "My First Program";
23             Console.WriteLine ( "Hello people! I am learning C#." );
24
25             Console.WriteLine ( "Press any key to exit" );
26
27             Console.ReadLine ( ); //Wait for a key to be pressed.
28         }
29     }
```

The "using" directive

- By using the "using" directive on top of a code file, you are specifying the namespace within which the classes whose methods you are calling, reside.
- The using directive is used to access classes that are outside the namespace of your application.
- If you do not provide a using directive, you must specify a fully qualified path to the method of a class you are calling.
 - System.Console.WriteLine();

Class definition

- The definition of a class begins with the keyword `class`, followed by the class name and the class body enclosed by a pair of curly braces.
- An access specifier such as `public`, `private`, and `internal` can be used to specify the level of the class' accessibility.
- Contrary to variables and methods, classes should be either `public` or `internal`.
- The default specifier is `internal`.

```
class Car
{
    //body
}
```

Access specifiers

- Access specifiers set the rules for how a class and its members can be accessed.
- There are several specifier types but let's begin learning the following three modifiers, all keywords:
 - `public`, `private` and `internal`.
- The `public` modifier makes a class or a member accessible for all assemblies.
- The `internal` modifier makes a class or a member only accessible within the assembly to which they belongs.
- The `private` modifier makes the class or the member unavailable for code outside the class.

C# Types

- The C# language is made of only five types.
 - `class`
 - `interface`
 - `delegate`
 - `struct`
 - `enum`
- All code you write in C# make use of these types.
- The most common type used in applications is of course the class type.

Members of a class

- Classes can have several types of members.
- The most important members are:
 - Fields, that represent their data
 - Methods that represent their behavior.
- Fields are variables that store the data.
- Methods perform operations.
- All members of a class must be declared.
 - Fields and Methods both need to be declared following C#'s declaration rules.

Declaration of fields

- The names **model** and **color** are two variables.
- A variable has a scope, data type a value, and a name.
- The variable **model** is of the type **string** which only can save text. The variable **isSold** only can save one of the values **true** or **false**, because it has a **bool** data type.

```
class Car
{
    //Fields (instance variables, attributes)
    private string model;
    private int numOfDoors;
    private double tankVolume; //liters
    private decimal price;
    private bool isSold;
```


Variable initiation

- The C# compiler automatically initiates all fields but not the local variables.
- The compiler initiates a variable to the data type's zero value.
 - An int variable is initiated to 0.
 - A double variable is initiated to 0.0.
 - A bool variable is initiated to false.
 - A string is initiated to null which means "nothing".
- A string is an object in C# and all objects are initiated to null.

Methods

- Methods have a variety of types but we begin with simplest type called `void`.
- A `void` method performs a task which is one or more statements.

```
public void SetDefaultValues ( )  
{  
    model = "Unknown";  
    numOfDoors = 4;  
    isSold = false;  
}
```

Statement and expression

- A statement is the smallest but complete unit of execution that directs the program to perform an action.

```
Console.WriteLine ( "Good morning!" );
```

- An expression is a sequence of one or more operands and zero or more operators that can be evaluated to a single value.
- Which ones are statements and which ones are expressions in the code below?

```
result = number1 + number2;  
  
if (DateTime.Now.Hour < 12)  
    Console.WriteLine ( "Good morning!" );  
else  
    Console.WriteLine ( "Good afternoon!" );
```

Our class so far

More methods can be added to perform more tasks.

Assignment statement

```
namespace CarPark  
{  
    0 references  
    class Car  
    {  
        //Fields (instance variables, attributes)  
        private string model;  
        private int numOfDoors;  
        private double tankVolume; //liters  
        private decimal price;  
        private bool isSold;  
  
        //Methods  
        0 references  
        public void SetDefaultValues ( )  
        {  
            model = "Unknown";  
            numOfDoors = 4;  
            isSold = false;  
        } //end of method  
    } //end of class  
} //end of namespace
```

Operators

- Operators are symbols that are applied to one or more operands in an expression or a statement.
- Some examples of operands are:
 - +, -, *, /, ++, <, >, >=, <=, &&, || and many more
- The statement:
 - `result = number1 + number2;`
Uses the operators '=' and '+' to carry out the addition.

Using classes

- A class definition is like a blueprint that specifies what its objects can do.
- An object is a space in memory that has been allocated and configured using the blueprint.
- Before you can use an object, you need to write code to allocate a memory block for it.
- The memory allocation takes place when the object is created at run-time. It then will have an address.
 - The variable we use for an object holds this address.

Creating an object

- The primitive variables such as int, double and bool need not to be created.

- They can directly be assigned a value:

```
int number = 5;
```

- Objects need to be created after their declarations. The keyword **new** is used to create an object.

```
//Declare a car object (a variable)
Car onCarObject;
//Create the object
onCarObject = new Car ( );

//call a method of the object to get info
onCarObject.DisplayInfo ( );
```

A guideline – use a minimum of two classes

- Write one class for each type of a problem.
- Save the class in its own file *.cs.
- Write a start-up class including the method Main.
- Inside the Main method, declare and create an object of the other class.
- Call the created object's methods to make things happen in your program.
- If you need to write more classes, write the classes and use objects of them in those client classes that their services.

An example

```
class CarParkProgram
{
    public static void Main ( )
    {
        other code

        //Declare a car object (a variable)
        Car carObject;
        //Create the object
        carObject = new Car ( );

        //call a method of the object to get info
        carObject.SetDefaultValues ( );
        carObject.DisplayInfo ( );

        Console.WriteLine ( "Press Enter to exit" );
        Console.ReadLine ( ); //Wait for a key to be pressed.
    }
}
```

The Car Park Application

```
*** Model: Unknown Sold: False
Press Enter to exit
```

Output from the carObject

Keep this class and the Main method as small as possible

Summary

- A C# class consists roughly of a fields and methods. These are members of a class.
- Every application must have one class with the method Main through which the program execution takes place.
- Fields are to be declared. A declaration statement consists of an access modifier, a data type and a variable name.
- A simple void method consists of access specifier, the keyword void, a method name, a pair of parentheses and a body enclosed by curly braces..
- Objects need to be created before they can be used.
- An object is an instance of a class.

Programming using C#

Variables, Data types and Declaration Part 1

Contents:

- Data representation
- Basic data types in C#
- Variables and declarations

Farid Naisan, farid.naisan@mah.se

Types of data

- Computer programs deal always with data that needs to be stored as input, output or for using between calculations.
- Data occurs in different forms. Three of the basic types are:
 - Numerical values,
 - Text,
 - Logical values,
- Other and more complex types are objects which can be derived using these primitive types.
- Numerical values have in turn different types, whole numbers, numbers with decimals.
- Computers understand only the binary language in which all data is represented by 0s and 1s, but values in the programming languages are represented in different formats, binary, decimal, hexadecimal, etc.

Farid Naisan

2

Binary Notation

- Data values are stored in the computer's memory, and because a computer understands only the binary language, the value is stored as series of 0's and 1's, called **bits**.
- Considering one **byte** (8 bits), the relation between a binary notation with base 2 (using digits 0, 1) and our natural decimal system using base 10 (using digits 0..9) can be written as follows:

$$(11111111)_2 = 2^8 - 1 = (255)_{10}$$

- Keeping the minimum and maximum limits of one bit, one byte consisting of 8 bits can represent 256 values:

$$1 \text{ to } 256, 0 \text{ to } 255, \text{ or } -128 \text{ to } +127$$

- The ANSI standard, for example, uses one byte to represent 256 alpha-numeric characters, e.g. 'A' = 65 and 'B' = 66

Using multiple bytes

- For values that do not fit in a byte, more bytes are combined.
- A two-byte memory space can store values 0 to 65535:

$$(1111111111111111)_2 = 2^{16} - 1 = (65535)_{10}$$

and so on.

- One Kilobyte (Kb) is equal to 1024 Bytes.
- One Megabyte (Mb) is equal to 1024 Kilobytes.
- One Gigabyte (Gb) is 1024 megabytes or 1,073,741,824 bytes, usually said as one billion bytes.
- One Terabyte (Tb) is 1024 gigabytes or 1,099,511,627,776 bytes usually said one trillion bytes.

Other numeral systems

- You may also encounter numbers using other bases when programming.
- The most popular ones, in addition to the binary and decimal systems are:
 - Octal, base 8, using 0..7 as the only digits,
 - Hexadecimal, base 16, using 0..9 and A..E (or a ..e).
- The number 10 represents the value 2 in binary system, 10 in decimal system, 8 in Octal system and 16 in Hexadecimal system.

Farid Naisan

5

The Unicode Standard

- Working and manipulating strings is usually a difficult task in programming.
- .NET has extensive support that enables programmers to efficiently create, compare, and modify strings as well as rapidly parse large amounts of text and data to search for, remove, and replace text patterns.
- A string is a sequence of characters. Characters in C# are saved using the Unicode standard.
- Unicode is an international standard for representing the characters found in most modern languages.
- The .NET Framework provides an extensive support encoding and decoding of characters according to Unicode

6

Data types

- The numerical, text and logical value types are basic types that most programming languages have built-in definitions for them.
 - `int`, `double`, `bool`, `byte`, etc.
- Objects are data types that we need to define them by ourselves.
 - Chair, Car, Child, Address,
 - Child: name, age, number of teeth
 - Address: street, zip code, city and country
- C# defines some other common data types too.
 - `DateTime`
 - `Color`,
 - `Font`, etc.

Farid Naisan

7

Data Type vs Type

- A computer program works with data and data has a type.
- The whole C# language is based only on **five types** which are:
 - **class**, **interface**, **delegate**, **struct** and **enum**
- In the .NET World, using the word "Type" is a preferred way of meaning a data type.
- Even basic data types such `int`, `bool` and `double` are types, these belong to the type `struct` (structure).

Farid Naisan

8

Variables

- Data values need to be saved for manipulation and later use. Values are saved in variables.
- A variable is a temporary storage location in the computer's memory, reserved for holding information while the program is running
- Variables are constantly being created and destroyed during the program execution.
- The value that is stored in a variable may change, and therefore it has the name "variable".
- To specify the type of data, the amount of memory space to reserve and to assign an initial value to the variable, we need to **declare** a variable.

Farid Naisan

9

Declaration of a variable

- The process of specifying the two main components, namely the type and name, of a variable, is called "declaration".

```
string message;
```
- `string` is the type of the variable and `message` is the name of the
- `String` gives both range of values that the variable can store, and also the type of value which is text in this declaration.
- Usually there could be a third part in the declaration and that is the initiation part, as in the statement below.

```
string message = "High there!";
```

Farid Naisan

10

Variables declaration

- The word `string` is a definition for a built in data type which can hold a sequence of Unicode characters, each of which receives a 2 Byte memory space.
- There are more such built in data types that will be discussed shortly.
- A variable has three characteristics:
 - A name so it can be referenced.
 - A range, min and max limits (for memory allocation) – the type of the variable
 - A value, the current value stored in the variable
- The range is determined by the type of the variable.
- A value is assigned a variable using the “=” sign.
 - `variableName = newValue;`

Farid Naisan

11

Variable names

- Rule for selecting variable name:
 - Must use a letter as the first character or an ‘_’
 - Can't use a space, period, comma, or other characters like ! : @, &, \$, #.
 - Cannot be a keyword (like `if`, `do`, `while`).
- Avoid name conflicts within a scope. Names at class level, e.g. variable names and method names, should not occur as duplicates.
- Variables inside a method should not have same name and they should not have same names as the names of the method parameters.
- Select names such that they imply their intent. Short names like `x`, `y`, `n` are not a part of a good programming style. Using “**numOfProductsInStock**” is a much better variable name than “**n**”!

Farid Naisan, Malmö University

12

Constants

- Constants are variables that can be assigned a value only once and by initialization.
- They are used to declare and hold values that are constant and cannot change again.
- Syntax:

- In a Class:

```
//maxNumOfYears and errMsg cannot change again
private const int maxNumOfYears = 100;
private const string errMsg = "Wrong input! Try again!";
```

- In a method

```
const double interestRate = 0.035; //local const
```

Short and long names for basic types

- Basic types are defined by a struct and are placed under the System namespace.
- To make it easier for programmers, the short names as aliases to their struct-names (the long names) are defined in C#. For example:
 - `int` is the short name and alias for `System.Int32`.
 - `bool` is alias for `System.Boolean`
 - `string` is alias for `System.String`
- Note that `string` is an object and not a primitive type although it is used much in the same as primitive types (like `int`, `bool`, etc).
- All the basic types, `String` (or `string`), `Double` (or `double`), `Int32` (or `int`) all provide useful data manipulations.

```
string resultString = String.Format("The result is {0,10:f2}", dblResult);
```

Built in data types

- C# provides many built-in types that programmers can directly use.
- The range of values a certain type can store as well as the type of the data, numbers, text, char, etc., are defined and the memory is allocated automatically.
- The tables that follow summarize the C# data types but you will mostly need to use only a few of these.
- The types beginning with the letter 's' (signed) are those that can accept both positive and negative types.
- The ones beginning with the letter 'u' (unsigned) can only take positive values (and therefore a larger max range).

C# data types

C# Type	Name in the .NET Framework	Size	Range of allowed values
sbyte	System.Sbyte	1 byte	Integers in the range of -128 to +127
byte	System.Byte	1 byte	Integers in the range of 0 to 255
short	System.Int16	2 bytes	Integers in the range of -32,768 to +32,767
ushort	System.UInt16	2 bytes	Integers in the range of 0 to 65535
int	System.Int32	4 bytes	Integers in the range of -2,147,483,648 to +2,147,483,647
uint	System.UInt32	4 bytes	Integers in the range of 0 to 4294967295
long	System.Int64	8 bytes	Integers in the range of -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807

C# data types – cont.

bool	System.Boolean	1 byte	Values true or false only
ulong	System.UInt64	8 bytes	Integers in the range of 0 to 18446744073709551615
float	System.Single	4 bytes	Floating-point numbers in the range of $\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$, with seven digits of accuracy
double	System.Double	8 bytes	Floating-point numbers in the range of $\pm 5.0 \times 10^{-324}$ to $\pm 3.4 \times 10^{30}$, with 15 digits of accuracy
decimal	System.Decimal	16 bytes	Floating-point numbers in the range of $\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$
char	System.Char	2 bytes	Unicode

Farid Naisan, Malmö University

17

Which types to use?

- The tables show several types that can be used with integers. Unless there are other reasons:
 - use int type for integer values.
- For number having a fractional part (floating point numbers), there float, double and decimal types.
 - Use double for floating point numbers (numbers with a decimal parts).
 - Use decimal when working with currency.
- For logical values such as yes or no, on or off, true or false, use bool.
- For textual data, use string. Unless, you are working strictly with chars, you can use string even with having one character.
- For individual characters, use char.

Farid Naisan, Malmö University

18

The decimal type

- The decimal type is actually neither a floating type nor a integer type.
- The keyword decimal denotes a 128-bit data type and it can store either a whole number or a decimal number up to 28-29 significant digits.
- It holds a binary integer value, together with a sign bit and an integer scaling factor that specifies what portion of the value is a decimal fraction.
- A decimal numbers have a more precise representation in memory than floating-point types (float and double), but it is slower in performance.
- This type is suitable for financial calculations or other calculations that require a large number of digits but cannot tolerate rounding errors.
- When using decimal with other numerical values in the same expression, you must explicitly do a type casting (conversion) to avoid a compilation error.

Integer Data Types

- byte, short, int, and long as well as the corresponding unsigned types numbers (ubyte, ushort, uint, and ulong) are all integer data types.
- They can hold whole numbers such as 5, 10, 23, 89, etc.
- Integer data types cannot hold numbers that have a decimal point in them.
- Integer values that you embed into C# source code are called *integer literals*.

Example of valid integers:

-5 12456 1077 -324567 +77L

Example of invalid integer literals:

-5. +55.0 1245,8 SEK1500.

- The default type for a integer literal is int.

Floating-Point Data Types

- Data types that allow fractional values are called *floating-point* numbers.
 - 1.7 and -45.316 are floating-point numbers.
- In C# there are three data types that can represent floating-point numbers.
 - float - also called *single precision* (7 decimal points).
 - double - also called *double precision* (15 decimal points).
 - decimal - smaller range and greater precision than float and double.

Floating Point Literals

- When floating point numbers are embedded into C# source code they are called *floating point literals*.
- The default type for floating point literals is double.
 - 29.75, 1.76, and 31.51 are double data types.
- Numbers with exponent belong also to this category.
- Example of valid floating point numbers:
 - -5. 3F 77.35 1.0e-3 21.0E4 2500M
 - **E** or **e** denotes "10 to the power of".

Floating-Point Literals

- A double value is not compatible with a float variable because of its size and precision.
 - `float number;`
 - `number = 23.5; // Error!`
- Doubles can be forced into a float by appending the letter F or f to the literal.
 - `float number;`
 - `number = 23.5F; // This will work.`
- Doubles can be forced into a decimal by appending the letter M or m to the literal.
 - `float number;`
 - `number = 23.5M; // This will work.`

Floating-Point Literals

- Literals cannot contain embedded currency symbols or commas.
 - `grossPay = $1,257.00; // ERROR!`
 - `grossPay = 1257.00; // Correct.`
- Floating-point literals can be represented in *scientific notation*.
 - `47,281.97 == 4.728197 x 104.`
- C# uses *E notation* to represent values in scientific notation.
 - `4.728197 x 104 == 4.728197E4.`

Scientific and E Notation

Decimal Notation	Scientific Notation	E Notation
247.91	2.4791×10^2	2.4791E2
0.00072	7.2×10^{-4}	7.2E-4
2,900,000	2.9×10^6	2.9E6

- A number can be written inside the code as either the decimal notation or the E-notation.

The decimal sign

- The decimal sign become an important issue when working with the user-interface.
- When inputting a floating point value like 4.56, the user is responsible for using the correct sign according to the regional settings in Windows.
 - Windows in Swedish uses the comma character as the decimal sign by default, for instance 4,56
 - Windows in English uses the dot notation by default, for instance 4.56.
- Using the C# objects, this will automatically correct and you should not need to worry about that when receiving input or showing output.
- However, when you write literal floating point numbers (constant numbers), you must use the dot notation or the exponential format as exemplified before. The code 4,56 will not compile but 4.56 will.

The Boolean Data Type

- The C# Boolean data type- `bool` can have two possible values.
 - `true`
 - `false`
- The value of a Boolean variable may only be copied into a Boolean variable.
- Number can be used instead of true or false.

The char Data Type

- The C# char data type provides access to single characters.
- char literals are enclosed in single quotation marks.
 - `'a'`, `'Z'`, `'\n'`, `'1'`
- Don't confuse char literals with string literals.
 - char literals are enclosed in single quotes.
 - string literals are enclosed in double quotes

Escape sequences

- There are characters in c# that you cannot write like a normal character like 'a'.
- These characters denote end of line, tab and so on.
- The set of these characters are known as Escape-sequences”.
- The characters are denotate by a combination of back-wards slash and a letter;
- - '\n' new line
 - '\t' new tab

The complete list can be found at: <http://blogs.msdn.com/b/csharpfaq/archive/2004/03/12/88415.aspx>

Escape sequences

- Because of the Escape sequences, whenever you have an apostrophe or a backslash character in code, the c# compiler will give an error if it does not recognize the sequence.
- To fix this problem, you can simply use an extra slash before the apostrophe or the backslash character, as in the examples below.

```
char c1 = '\''; //c1 contains the char '  
char c2 = '\"'; //c2 contains the char "  
char c3 = '\\'; //c3 contains the char \
```

Alternative to escape sequences

- An alternative to using escape sequences is to use the helper objects from the .NET Framework that represent these characters in other ways.
- The object `Keys` defines all keys on the keyboard.
- The object **Environment** has also some useful properties that can be used. **Environment.NewLine**, for example causes a new line in your text.
- Using the above instead of Escape sequences is a better alternative as these are objects.

Strings and Unicode

- Internally, characters are stored as binary numbers.
- Character data in C# is stored as Unicode characters.
- The Unicode character set can consist of 65536 (2^{16}) individual characters.
- Each Unicode character takes up 2 bytes in memory.
- The first 256 characters in the Unicode character set are compatible with the ASCII* character set.
 - The char 'A' has a numerical value of 65

*American Standard Code for Information Interchange

String literals

- String literals are enclosed inside a pair of quotation marks:
 - "5" "Hi there"
- As mentioned before, due to escape sequences, the occurrence of a backslash or an apostrophe may provide a compilation problem. The following line will not compile:

```
string fileName = "D:\Projects\CSharp\ToDo.txt"; //Error
```

- The correct way to code the above is:

```
string fileName = "D:\\Projects\\CSharp\\ToDo.txt";
```
- An alternative way is to use the "@" character. This character tells the compiler that all characters are included in a string (notice the syntax):

```
string fileName = @"D:\Projects\CSharp\ToDo.txt";
```

More on variables

- Information in a C# program is stored in memory.
- Variable names represent a location in memory.
- Variables in C# are also called fields.
- Variables are created by the programmer who assigns it a programmer-defined identifier, ex:

```
int hours = 40;
```

- In this example, the variable *hours* is created as an integer and assigned the value of 40.
- A literal is a value that is written into the code of a program, like 5, and "hi".

Variables and Literals

```
int number; //This is a variable declaration

//To store a value in the number variable, we use
//an assignment statement.

//In an assignment, the value at the left side will be
//overwritten by the value at the right side.
number = 77; //This is an assignment statement
```

*The value 77 is stored in
Memory. The memory
address is referred to by the
variable name, number..*

0x000	
0x001	77
0x002	
0x003	

This is a string literal.

```
Console.Write("The current value is: ");
Console.WriteLine(number);
```

35

Variable Names

- Variable names should be descriptive.
- Descriptive names allow the code to be more readable; therefore, the code is more maintainable.
- Which of the following is more descriptive?

```
double str = 0.0725;
double salesTaxRate = 0.0725;
```
- Try to make your code *self-documenting* by using suitable names for variables and methods.
- Use as long a name as you may need.

36

C# Naming Conventions

- Variable names should begin with a lower case letter and mix case thereafter:
Ex: int `calculateTaxRate`
- Class names and method names should begin with an upper case letter. Ex:

```
public class Calculator
{
}
```
- Class and variable names are chosen as noun or noun phrases.
- Method names are usually verb phrases.

37

Summary

- *The topic continues in the next module!*
- Every program works with data. Data needs to be saved in an object for processing.
- To save data we use variables and these represent some space in memory where the data is saved.
- We must tell compiler to allocate correct amount of memory space for a variable. We use data types.
- C# has many built-in data types, but the rule of thumb is to use:

<code>double</code>	for floating point numbers (numbers with fraction). This is default floating point type in C#
<code>int</code>	for integer numbers. This is the default integer type in C#.
<code>bool</code>	for logical values
<code>char</code>	for single characters
<code>decimal</code>	for data requiring high precision such as money and financial calculations.

Farid Naisan, Malmö University

38

Programming in C# I

Getting Started

- Why are we here? Where are going?
- Tools and Development Environment
- Where to begin?

I must get a computer to help me

- An early morning, APU is trying to learn to count:
 - $3 + 5 = 8$
 - $2 + 3 = 5$
 - $2.05 + 1.7 = ???$
- Pleeez write a program that adds two numbers:
 - $C = A + B$
 - Ex: $A = 6$, $B = 9$,
 - $C = 6 + 9 = 15$



- What to do?
 - from problem to program!

Programming

.NET

- Why program?
 - Computers need instructions.
 - Instructions are given in form of programs
 - Programs are written in a programming language.
- Where are we going?
 - To learn programming,
 - in the OOP way,
 - why OOP? because it is a modern methodology,
 - with the programming language C#
 - why C#? because it is an easy language but very powerful!

To program...

.NET

- Where to begin?
 - Select your language (C#).
 - Install the required tools (programs) that help you:
 - write code using the language syntax and semantics(for the compiler to understand),
 - compile code - translate code to machine code, as a sequence of 0's and 1's (for the computer to understand),
 - run code (to test if the program works as it should).

Program or application

- Program or application are in essence the same thing.
 - Something that can run on a computer.
- Different types of applications exist:
 - Desktop applications
 - Web applications
 - Mobile applications
 - Web services, etc.
- There are different models (methodologies) for the development:
 - OOP – Object-oriented programming.
 - Procedural – programming with functions or procedures.

Farid Naisan, Malmö University

5

What and how are we going to do?

- We are going to develop C# desktop applications in this course.
 - Console applications
 - Windows Forms applications
- We are going to use OOP from the very beginning.
- We are going to use C# which is a part of the .NET platform.
- .NET is a collection of Microsoft's technologies for developing, testing, running and distributing different types of applications.

Farid Naisan, Malmö University

6

What is .NET Framework

- The .NET Framework is a software framework that has two main parts:
 - FCL (Framework Class Library) - a large library of classes
 - CLR (Common Language Run-time) – run-time engine
- The C# and Visual Basic are two of many languages that come with .NET.
- Every programming language has an SDK (Software Development Kit). For writing a C# application, we need:
 - .NET SDK that include .NET Framework,
 - A text editor.

Visual Studio

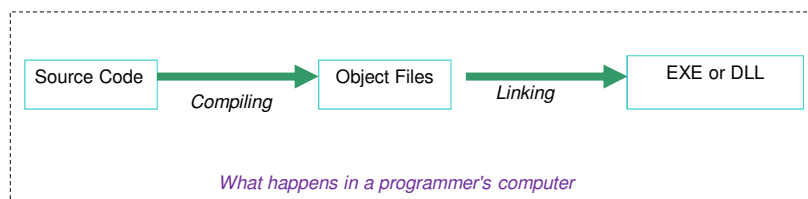
- Visual Studio (VS) is a comprehensive collection of tools and services for developing, testing, running and distributing a wide variety of applications.
- If you install VS, you don't need to install any other software for writing C# applications.
- VS installs all the SDK and sets up all environmental parameters.
- VS runs under Microsoft Windows.

To write a C# application

- A C# program consists mainly of **classes**.
- A class defines a group of similar **objects**.
- An **object** is an instance of a class.
- When our program is running, it is the objects that together make things work:
 - Every object has its own set of values (data) and communicates with other objects by messages.
 - Data is stored in the objects instance variables.
 - Messages are sent by calling other objects' methods.

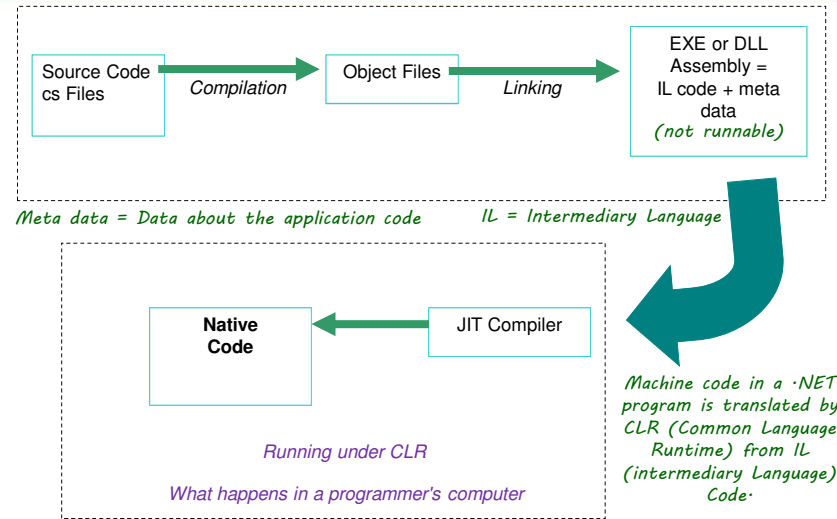
A general programming cycle

- General scenario



- There are SDK tools that carry out compilation and linking in one step.
- This is called **Build**.

Programming cycle – C# (.NET)



Farid Naisan, Malmö University

11

To Do? A round up

- We are going to write C# applications
- To do that, we need to install either:
 - .NET SDK and a text editor (Notepad, textpad, Notepad++, etc)., OR
 - Visual Studio, SharpDev, Mono, etc.
- Visual Studio is highly recommended.
- Follow the steps on the next slide.

Farid Naisan, Malmö University

12

Getting started - steps

- Install VS, if you have Windows as operating system
 - On Mac, install BootTrap, install Windows and then install VS.
- Creating a new project
- Write source code, classes. Follow precisely the languages rules (syntax).
- Build the project.
- Run and test.

Sammanfattning

- C# is a .NET language. We must have .NET Framework to develop and run a C# program.
- The .NET Framework is nowadays a part Windows and will therefore be automatically installed when installing windows.
- Visual Studio is a complete tool for developing, testing, running and distributing different types of applications.
- However, it is not necessary to have VS to write a C# application.
- There are other tools such as DevSharp which also can be used.
- It is fully possible to develop a C# application by using the .NET and a simple text editor such Windows Notepad, although it is quite tedious.