.NET Overview

(modified)

developmentor

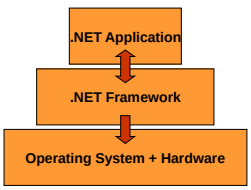
Objectives

- **Introduce .NET**
  - overview
  - languages
  - libraries
  - development and execution model
- **Examine simple C# program**

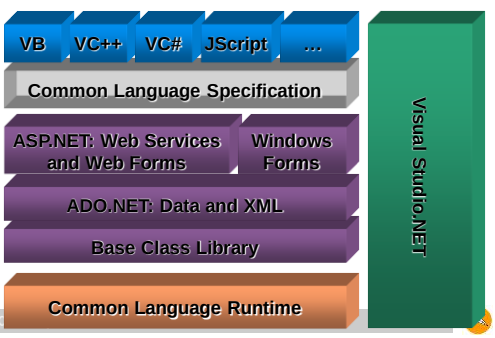
.NET Overview (released in 2002)

- **.NET is a sweeping marketing term for a family of products**
  - development tools and languages
  - platform
  - application management servers
  - value-added services (Skittles)

.NET – What Is It?



Framework, Languages, And Tools



Software development

- **.NET software development and execution has many actors**
  - languages
  - libraries
  - compilers
  - intermediate language
  - execution engine

Languages

- **Many .NET programming languages available**
  - C#
  - VB.NET
  - C++
  - etc.
- **Language choice typically based on many factors**
  - programmer background
  - problem domain
  - language features
  - corporate mandate

Language power

- **All languages can access .NET infrastructure**

C#

```
class Hello
{
  static void Main()
  {
    System.Console.WriteLine("hello");
  }
}
```

VB.NET

```
Class Goodbye

  Shared Sub Main()
    System.Console.WriteLine("goodbye")
  End Sub

End Class
```


Language interoperability

- **All .NET languages can interoperate**

C# calling VB.NET

```
class Hello
{
  static void Main()
  {
    System.Console.WriteLine(Greeting.Message());
  }
}
```

```
Class Greeting
  Shared Function Message() As String
    Return "hello"
  End Function
End Class
```


Language variability

- **Not all .NET languages have exactly the same capabilities**
  - differ in small but important ways

C#

```
class Hello
{
  static void Main()
  {
    int  i;
    uint u;
  }
}
```

signed integer

unsigned integer

VB.NET

```
Class Greeting
  Shared Sub Main()
    Dim i as Integer
  End Sub
End Class
```

signed integer only

Common Language Specification

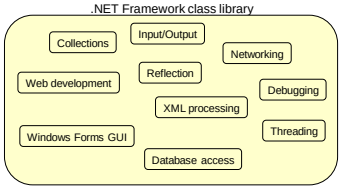
- **Common Language Specification (CLS) defines type subset**
  - required to be supported by all .NET languages
  - limiting code to CLS maximizes language interoperability
  - code limited to CLS called *CLS compliant*

not CLS compliant to use `uint` in public interface of public class

```
public class Calculator
{
  public uint Add(uint a, uint b)
  {
    return a + b;
  }
}
```

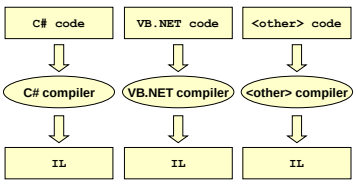

Library

- **Extensive set of standard libraries available**
  - for wide range of application types
  - called *.NET Framework class library*



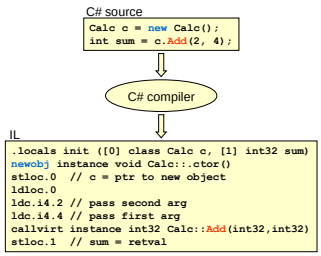
Compilation

- **Compilers produce *Intermediate Language* (IL)**
  - IL is not executable
  - similar to assembly language
  - processor independent



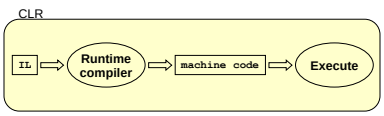
IL

- **C# compiler translates C# source code into IL**



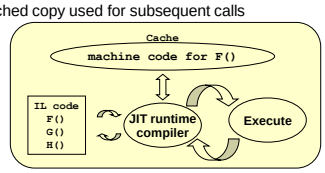
Execution engine

- ***Common Language Runtime* (CLR) is the execution engine**
  - loads IL
  - compiles IL
  - executes resulting machine code



JIT runtime compile

- **IL is compiled into machine code at runtime by the CLR**
  - compiles methods as needed
  - called *just in time* (JIT) compile
- **JIT compilation model:**
  - first time method is called the IL is compiled and optimized
  - compiled machine code is cached in transient memory
  - cached copy used for subsequent calls



C# program

- **C# program basics**
  - source file has `.cs` extension
  - **`namespace`** used to group related types
  - **`class`** defines new type
  - **`Main`** is application entry point
  - **`WriteLine`** writes output
  - `{` and `}` delimit code block

MyApp.cs

```
namespace MyNamespace
{
  class MyApp
  {
    static void Main()
    {
      System.Console.WriteLine("hello");
    }
  }
}
```