Lecture 1
Program memory structure. OOP basics

Programming II

School of Business Informatics
Autumn 2016

(: Programmer - n. [proh-gram-er]: a person who fixes problems that you don't know you have, in a way you don't understand :)

What is a computer program?

A **program** is a sequence of instructions represented in two main forms:

Human-readable source code

```
print "Hello world"
c = a + b;
...
```

Executable code

```
02 39 30 39 90 9A 8E F5
49 09 38 97 39 BC D5 38
A6 B7 00 00 00 40 35 34
34 55 98 39 C3 F5 24 E6
...
```

What is a computer program?

A **program** is a sequence of instructions represented in two main forms:

Human-readable source code

```
print "Hello world"
c = a + b;
...
```

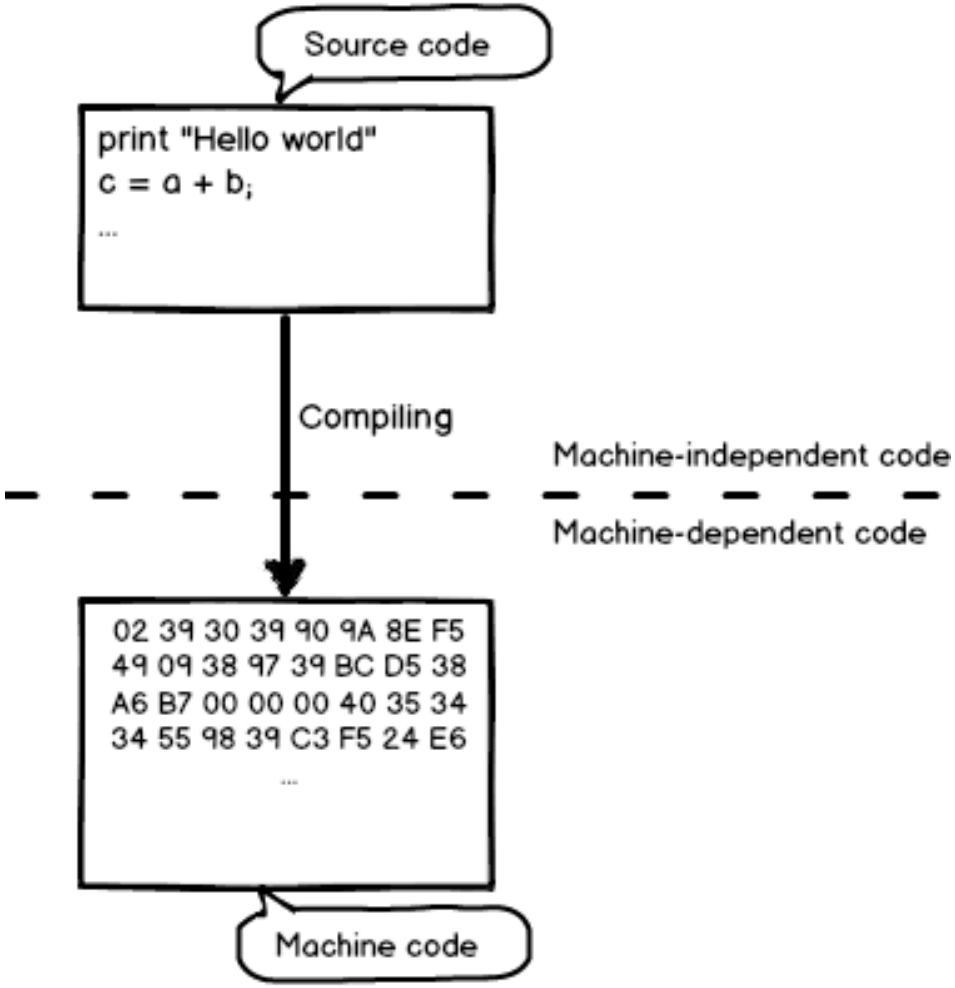
Executable code

```
02 39 30 39 90 9A 8E F5
49 09 38 97 39 BC D5 38
A6 B7 00 00 00 40 35 34
34 55 98 39 C3 F5 24 E6
...
```

How to convert from one form to another?

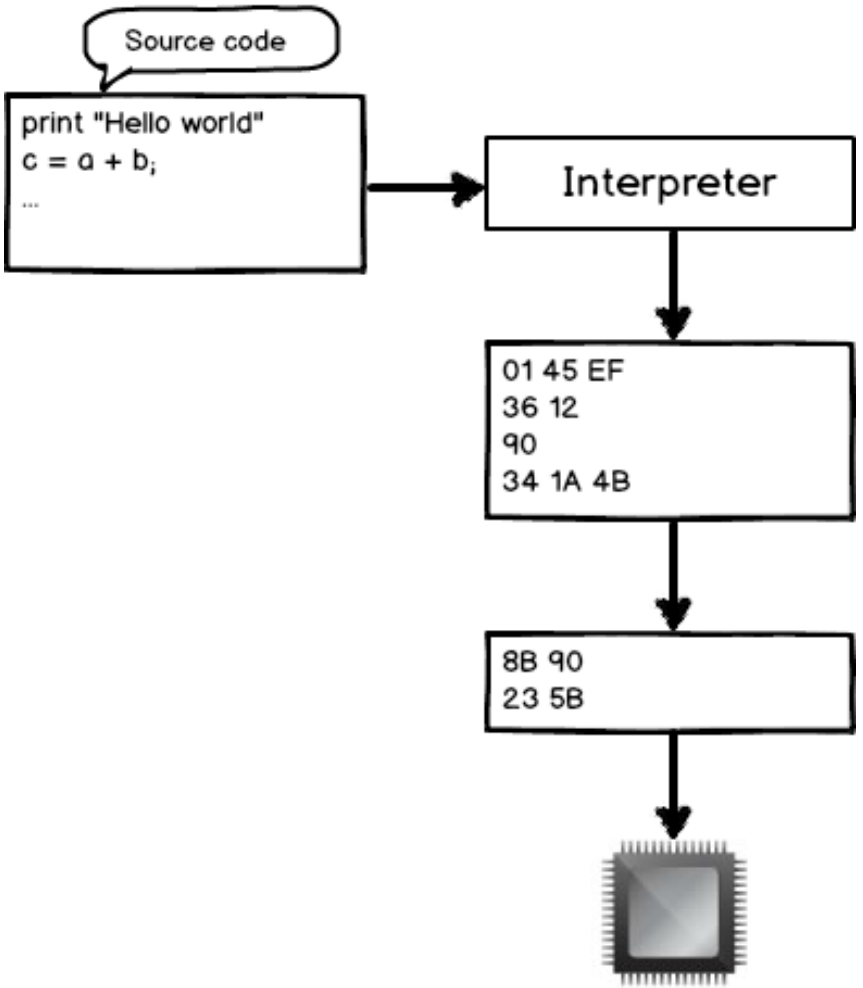
Conversion 1. Compilation

With languages, such as C, C++, Pascal, the whole source code is compiled into executable code **before running the program**



Conversion 2. Interpretation

When using script languages, such as PHP, Javascript the transformation from source code to executable code is done **on the fly**



Compilation vs Interpretation

Compilation:

- Fast execution of the program

Interpretation:

- Portable software

Compilation vs Interpretation

Compilation:

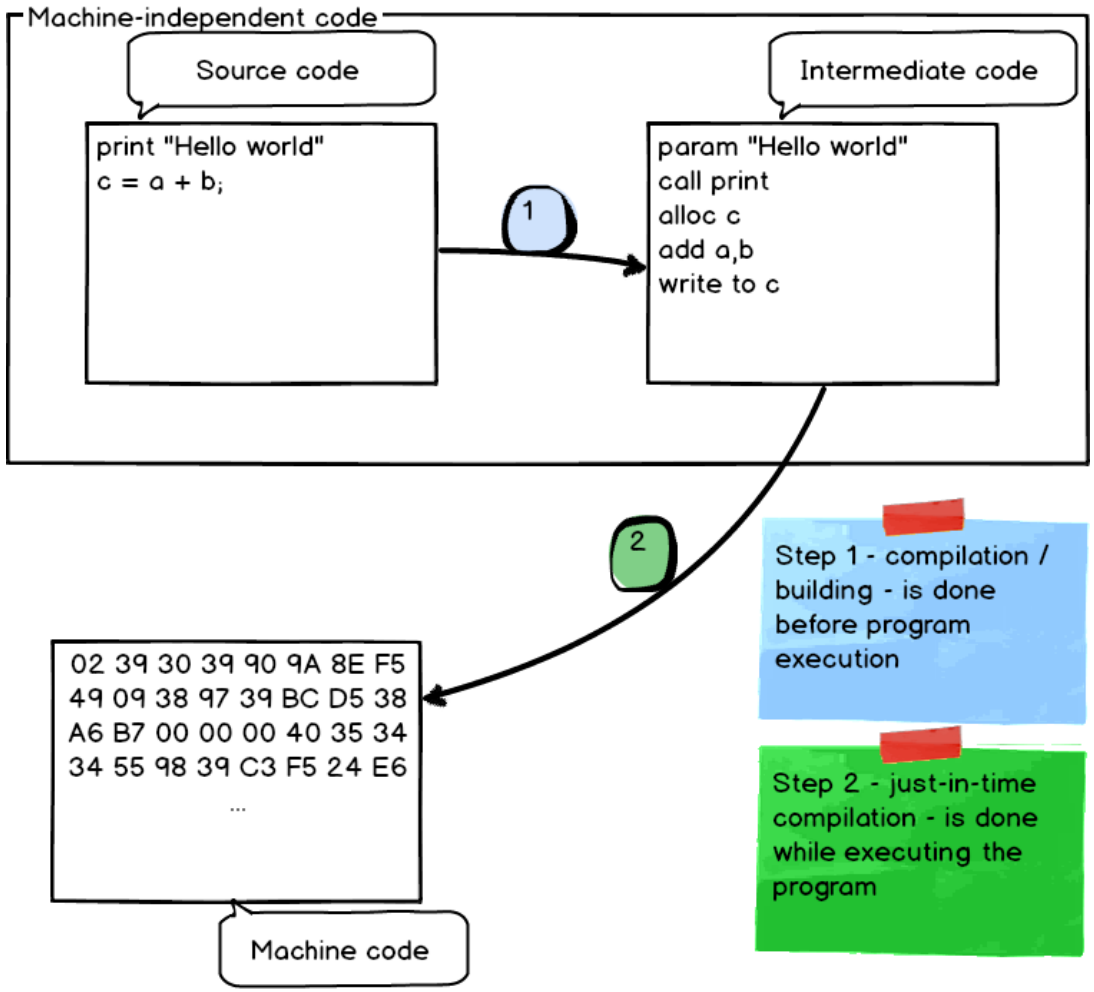
- Fast execution of the program

Interpretation:

- Portable software

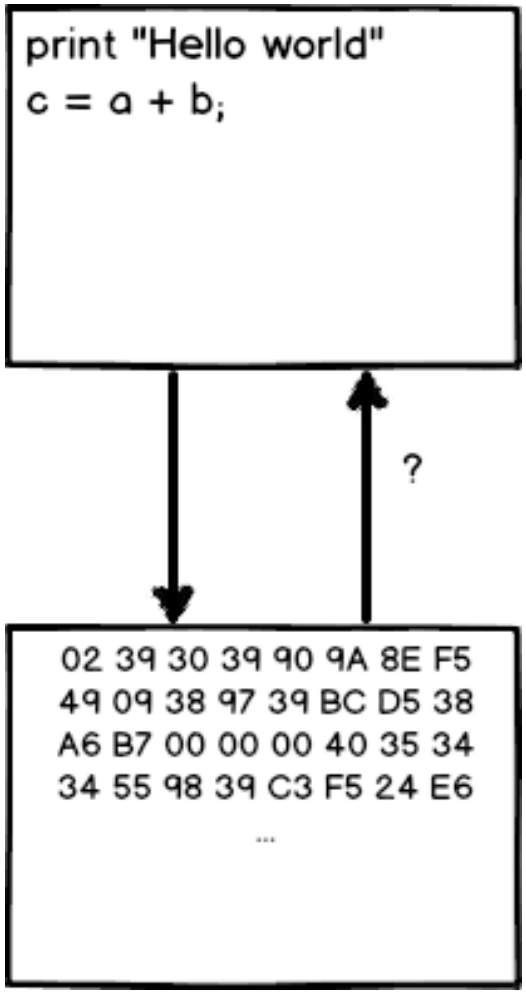
Can we combine advantages of both methods?

Intermediate code and just-in-time compilation 6



Two-way traffic?

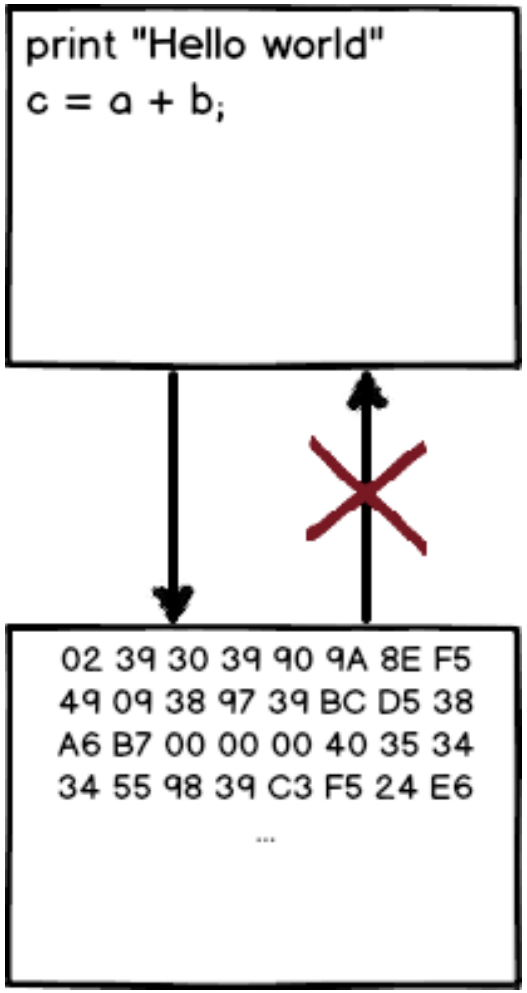
Is there a way to restore the complete source code from an executable?



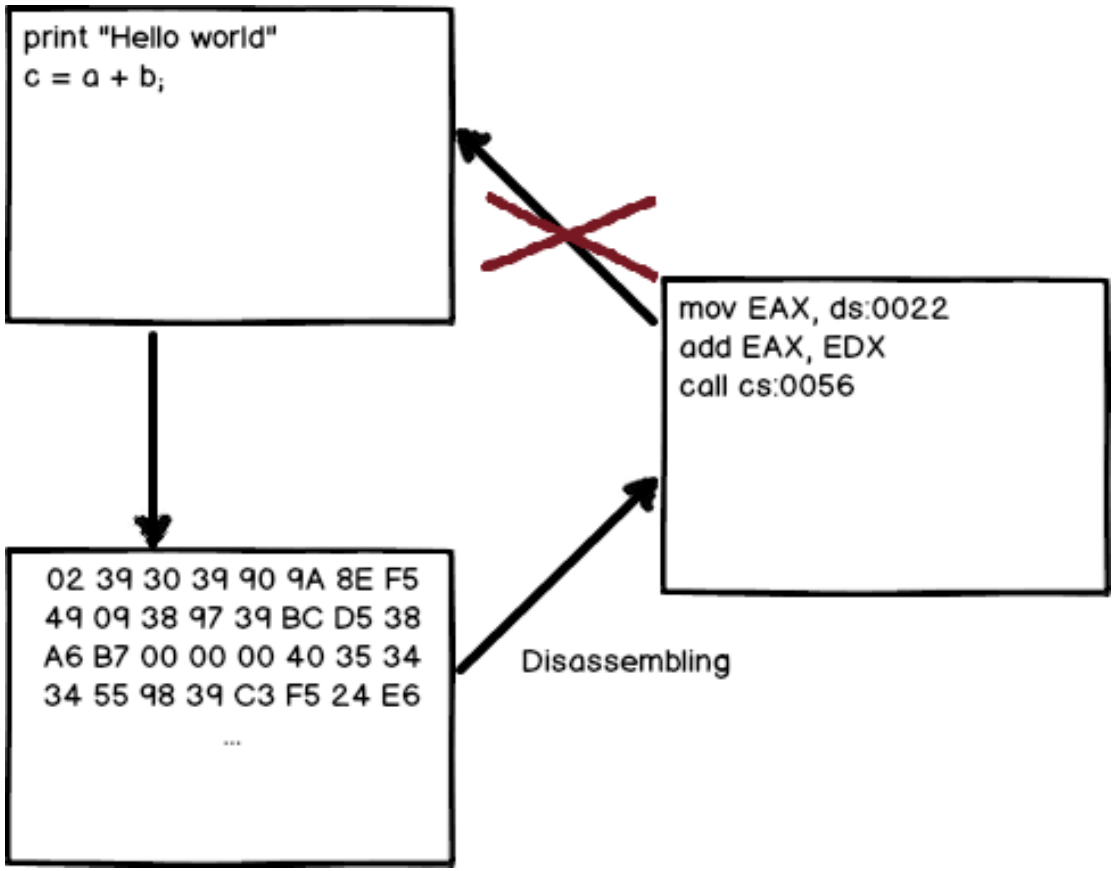
Two-way traffic?

Is there a way to restore the complete source code from an executable?

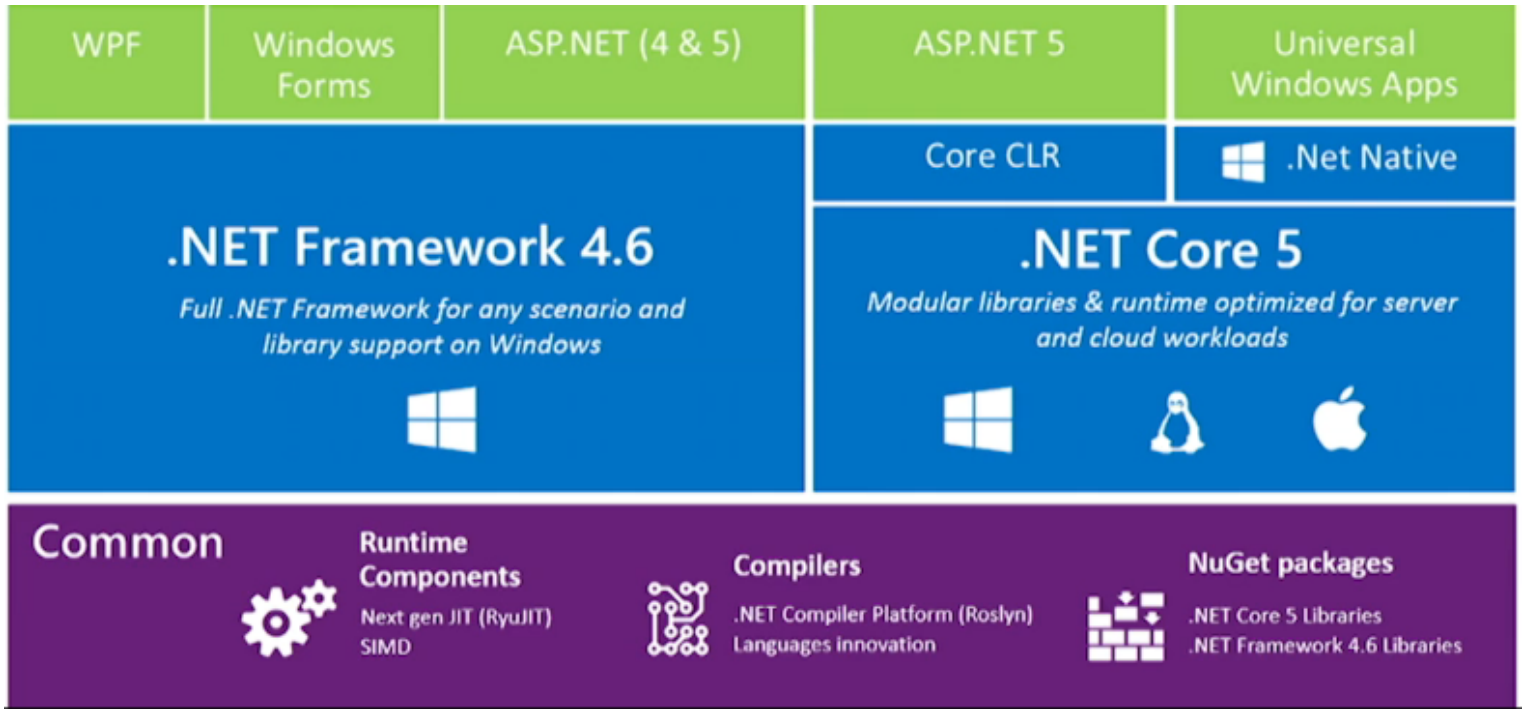
The answer is no. Most of high-level details (names of variables, functions, classes, etc.) are lost during the conversion.



Disassembly



.NET Framework state-of-the-art



Common Language Runtime (CLR)

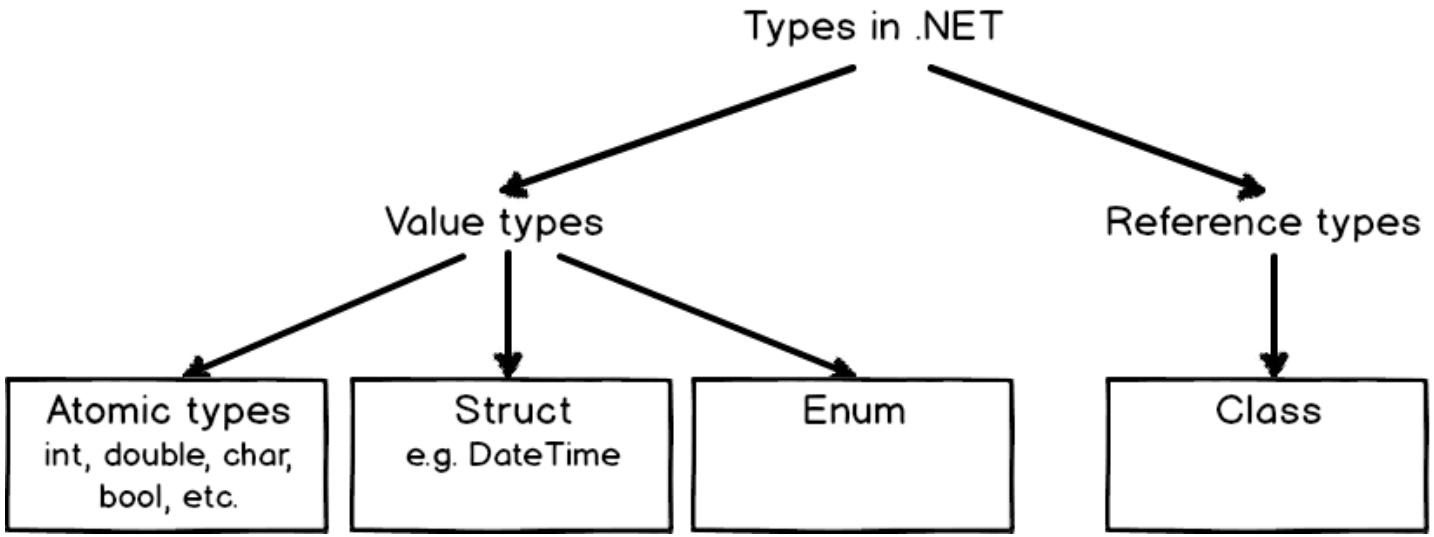
The key part of the .NET Framework is the CLR, which is responsible for:

- Just-in-time compilation
- Memory management
- Ensuring type safety
- Exception handling
- Security management

and some other important tasks

Values and references

- Variables of value types store data in the place where they are declared
- Reference variables store address of a block of memory where data resides
- A reference variable has a value of **null** when it does not point to any allocated block of memory



Heap and stack

.NET applications use two types of memory: a heap and a stack.

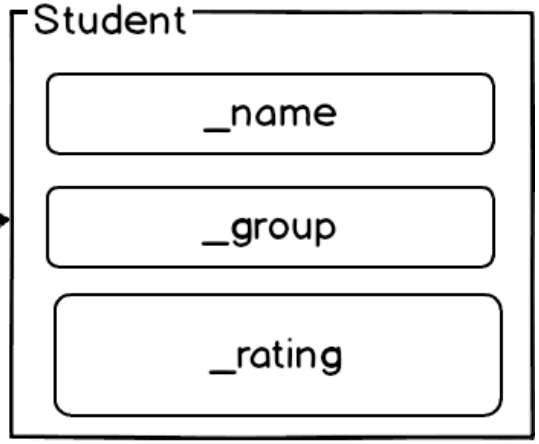
The **stack** stores:

- Variables declared inside methods
- Method parameters
- Return addresses forming a chain of method calls

The **heap** stores data of all objects (instances of classes)

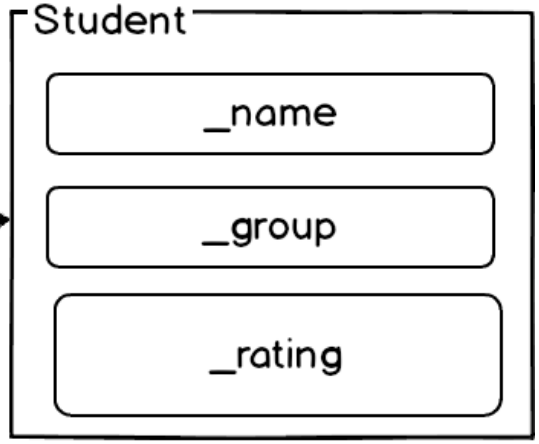
- Data definition and related code are placed in the same container - a class
- Classes serve as blueprints for objects (class instances)

```
class Student
{
    string _name;
    Group _group;
    double _rating;

    public void SetRating(double rating)
    {
        _rating = rating;
    }
}
```



- Data definition and related code are placed in the same container - a class
- Classes serve as blueprints for objects (class instances)

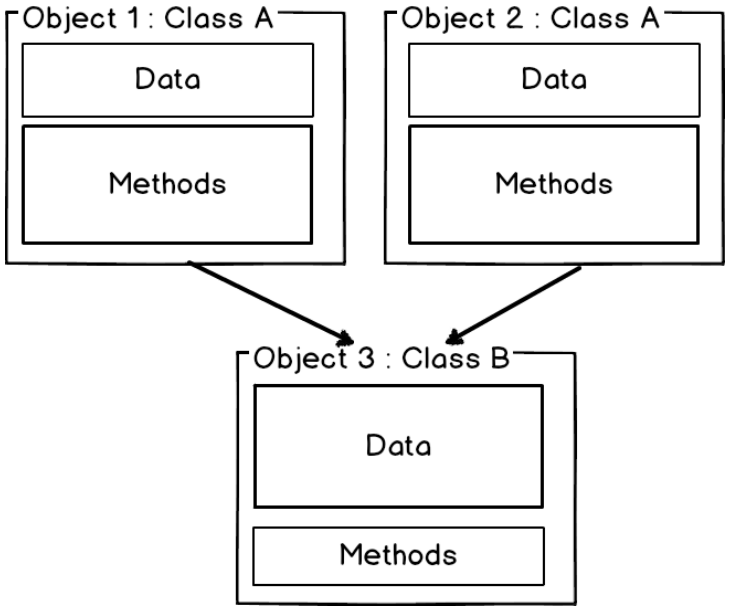
```
class Student
{
    string _name;
    Group _group;
    double _rating;

    public void SetRating(double rating)
    {
        _rating = rating;
    }
}
```



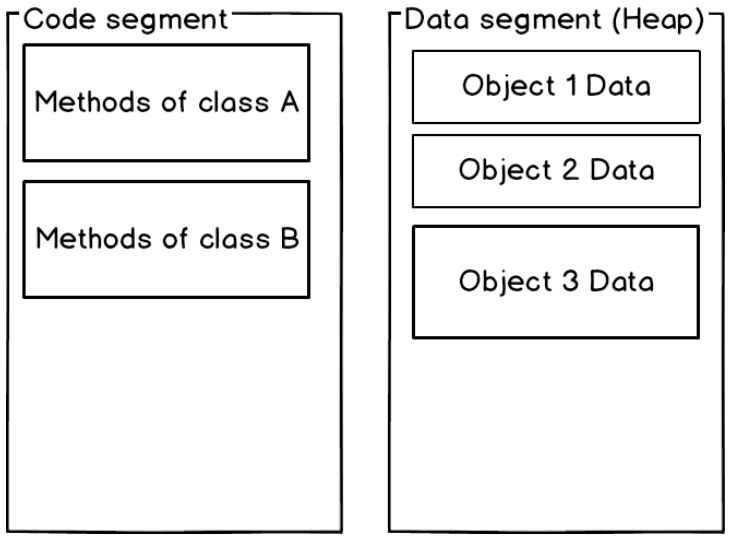
Where are methods placed?

Structure of an OO program

Logically a OO program is formed by a number of objects, each containing data and methods for working with it



Physically methods and data are placed in different areas of memory (code segment and data segment respectively)



Classes in C# 15

C# classes can contain:

- Fields / variables
- Methods
- Properties
- Constructors
- Events

Classes in C# 15

C# classes can contain:

- Fields / variables
- Methods
- Properties
- Constructors
- Events

Which of the items above relate to data and which to code?

Classes in C#

C# classes can contain:

- Fields / variables - data
- Methods - code
- Properties - code
- Constructors - code
- Events - data

Properties or methods?

Both properties and methods can be used to access private data inside the class.

- In general, properties represent data while methods represent actions
- Properties should not contain complicated calculations
- A class needs to be designed in a way that its properties can be set in any order

More advice on how to use properties and methods on MSDN

Object initialization

- A **constructor** is a special method inside a class that is responsible for initialization of an object's state
- A constructor is called only **once** when a new object is created
- A class can have more than one constructor. In this case all class constructors have to differ in signature
- In case no constructors are defined in a class, the compiler automatically inserts a default constructor
- Constructors can invoke each other - see example

Class or struct

- Both classes and structs in C#:
  - Are containers for structuring data
  - Group together data and related code
  - Can contain fields, methods, properties and constructors
- Structures do not support inheritance
- Structures are value types whereas classes are reference types

Most types in .NET are classes.

Immutable objects

An immutable object cannot change its state once created. Instead a new object is instantiated each time a source object is changed.

Immutable objects

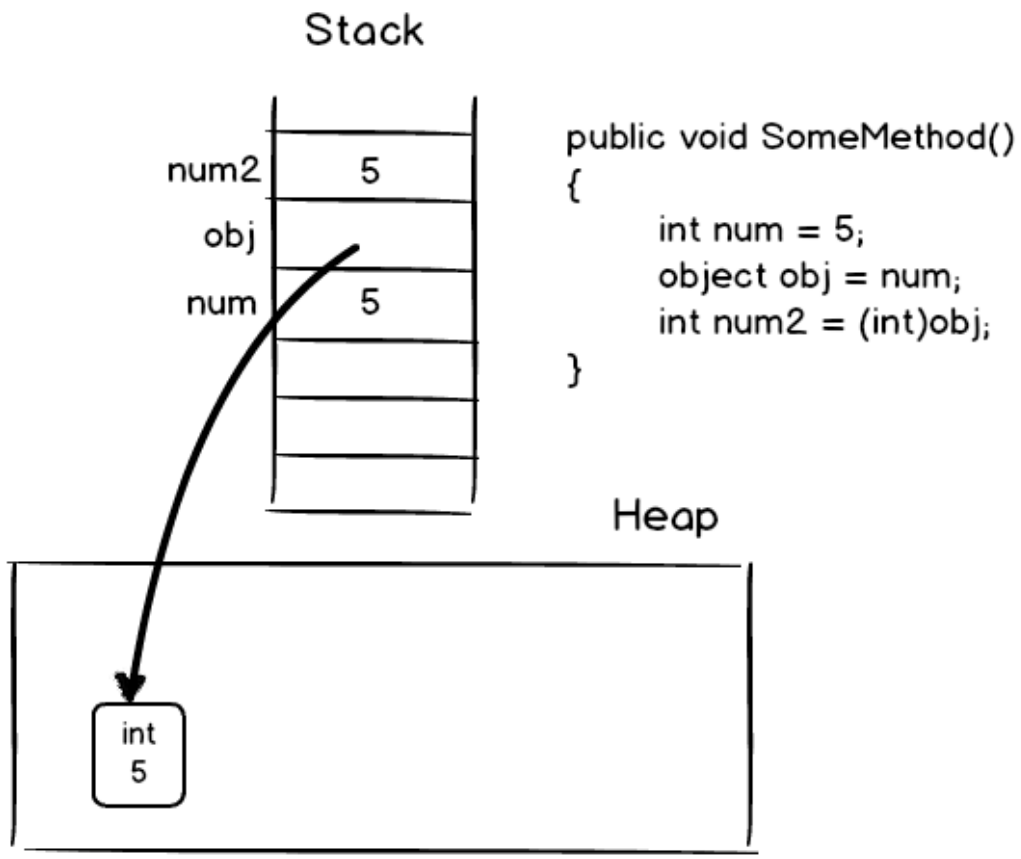
An immutable object cannot change its state once created. Instead a new object is instantiated each time a source object is changed.

- Immutability has nothing to do with value or reference types. It results from the way a class is designed
- There is however a strong recommendation to always make a struct immutable as the opposite may lead to errors (see MutableStruct project in the supplement)

Boxing and unboxing

- Any .NET type can be converted to object (System.Object)
- If the converted type is a value type, boxing occurs.
- When converting from a reference type to a value type the variable is unboxed.
- Often explicit boxing happens, e.g. when passing a value type to a method accepting object.

Boxing/unboxing example



Boxing costs: demo

"Boxing" project in the supplement:

- An Arraylist internally stores data as an array of objects. Each value type variable is boxed
- A List<T> stores values inside a single container