

C# Language & .NET Platform

6th Lecture:

C# Type System

<http://d3s.mff.cuni.cz/~jezek>

Department of
Distributed and
Dependable
Systems



Pavel Ježek

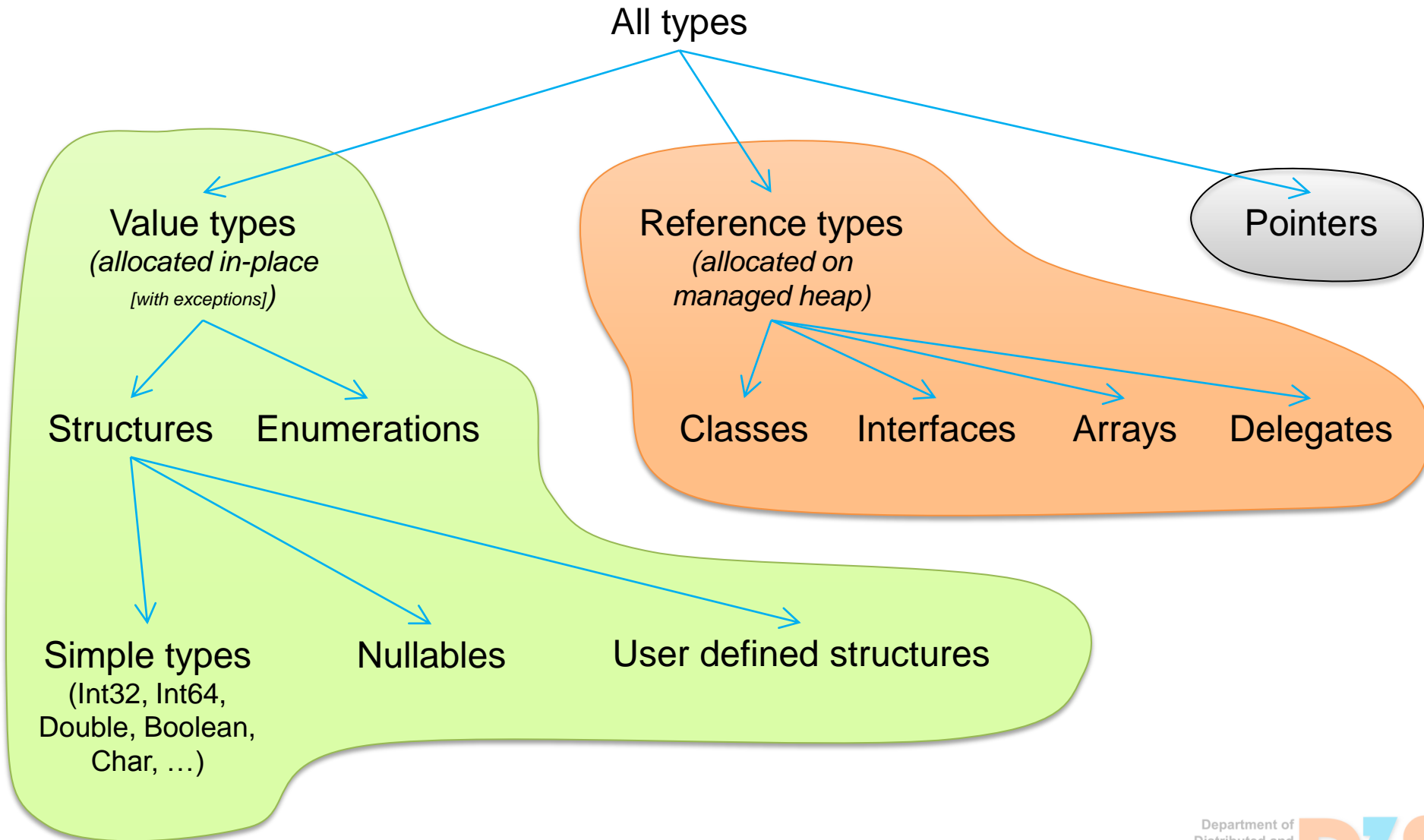
pavel.jezek@d3s.mff.cuni.cz



CHARLES UNIVERSITY IN PRAGUE
faculty of mathematics and physics

Some of the slides are based on University of Linz .NET presentations.
© University of Linz, Institute for System Software, 2004
published under the Microsoft Curriculum License
(http://www.msdnaa.net/curriculum/license_curriculum.aspx)

CLI Type System



Parameters

value parameters (input parameters)

```
void Inc(int x) {x = x + 1;}
void f() {
    int val = 3;
    Inc(val); // val == 3
}
```

ref parameters (transient parameters)

```
void Inc(ref int x) { x = x + 1; }
void f() {
    int val = 3;
    Inc(ref val); // val == 4
}
```

- "call by value"
- formal parameter is a copy of the actual parameter
- actual parameter is an expression

- "call by reference"
- formal parameter is an alias for the actual parameter
(address of actual parameter is passed)
- actual parameter must be a variable

Declaration of Local Variables

```
void foo(int a) {
    int b;
    if (...) {
        int b;           // error: b is already declared in the outer block
        int c;
        int d;
        ...
    } else {
        int a;           // error: a is already declared in the outer block (parameter)
        int d;           // ok: no conflict with d in the if block
    }
    for (int i = 0; ...) {...}
    for (int i = 0; ...) {...} // ok: no conflict with i from the previous loop
    int c;               // error: c is already declared in a nested block
}
```

```
int e = 1, f;
if (e == 1) {
    f = 2;
}
e = f; // error: f is not initialized in every possible execution path
```

Parameters

value parameters (input parameters)

```
void Inc(int x) {x = x + 1;}  
void f() {  
    int val = 3;  
    Inc(val); // val == 3  
}
```

ref parameters (transient parameters)

```
void Inc(ref int x) { x = x + 1; }  
void f() {  
    int val = 3;  
    Inc(ref val); // val == 4  
}
```

out parameters (output parameters)

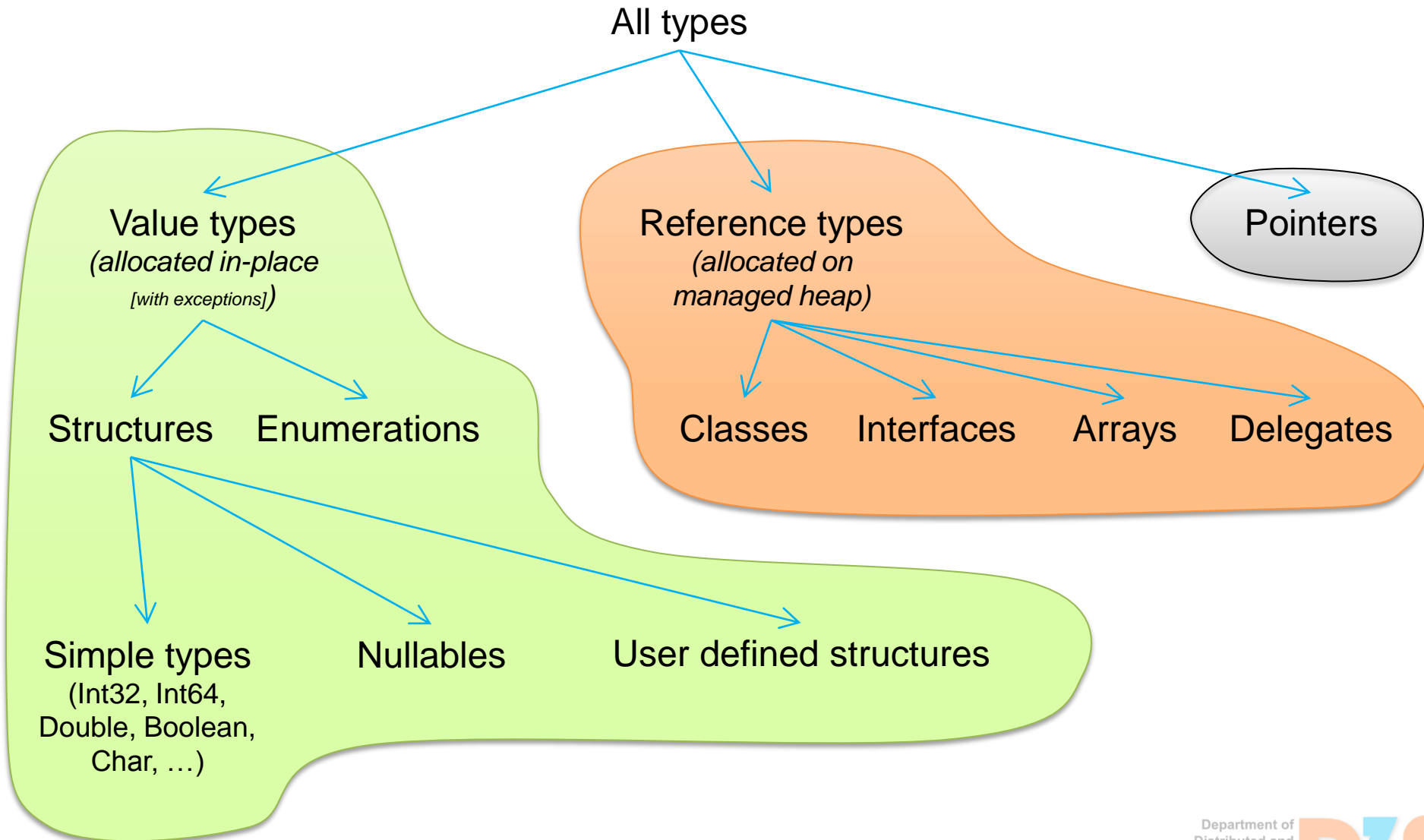
```
void Read (out int first, out int next) {  
    first = Console.Read();  
    next = Console.Read();  
}  
void f() {  
    int first, next;  
    Read(out first, out next);  
}
```

- "call by value"
- formal parameter is a copy of the actual parameter
- actual parameter is an expression

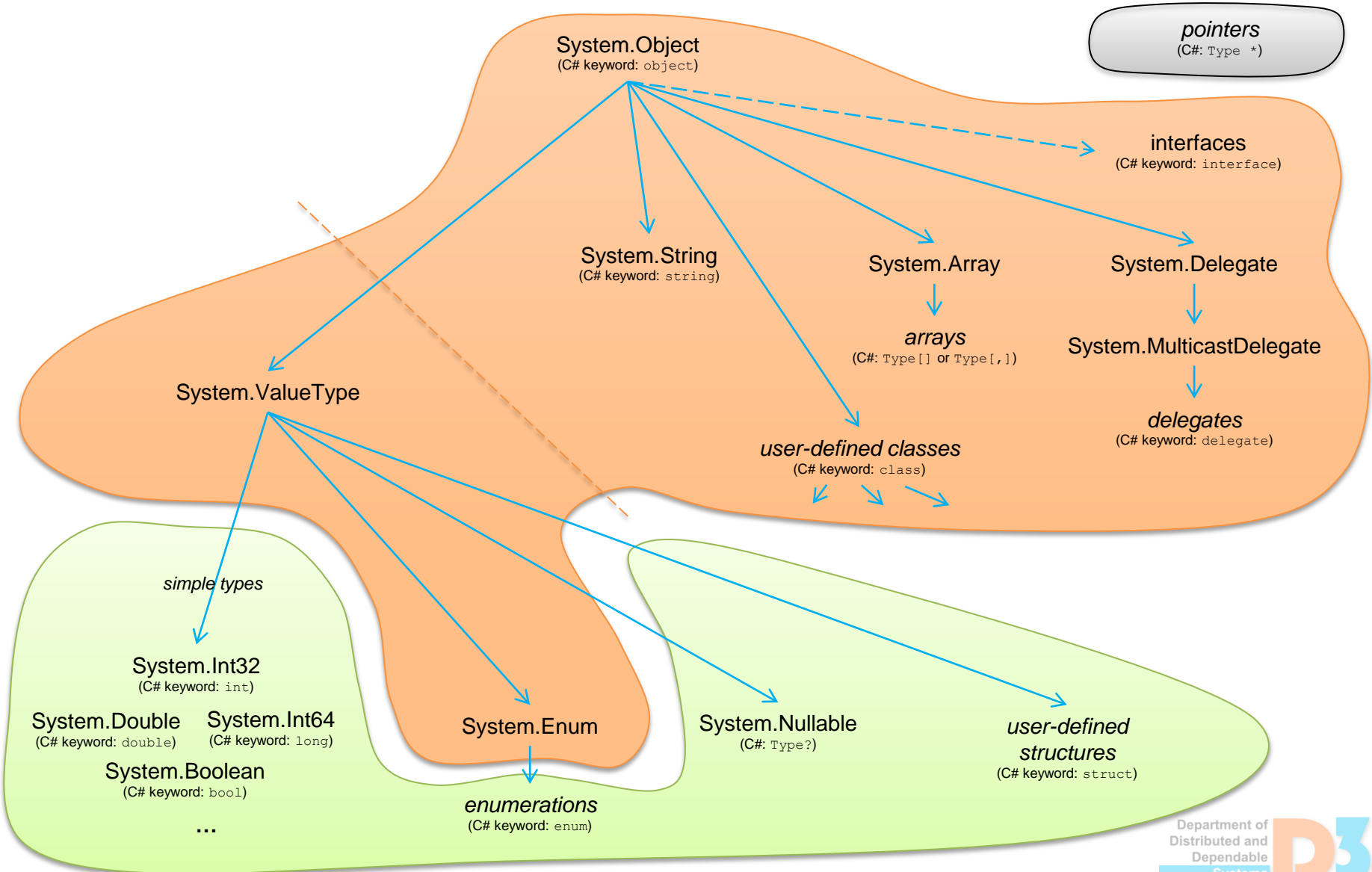
- "call by reference"
- formal parameter is an alias for the actual parameter
(address of actual parameter is passed)
- actual parameter must be a variable

- similar to ref parameters
but no value is passed by the caller.
- must not be used in the method before it got a value.

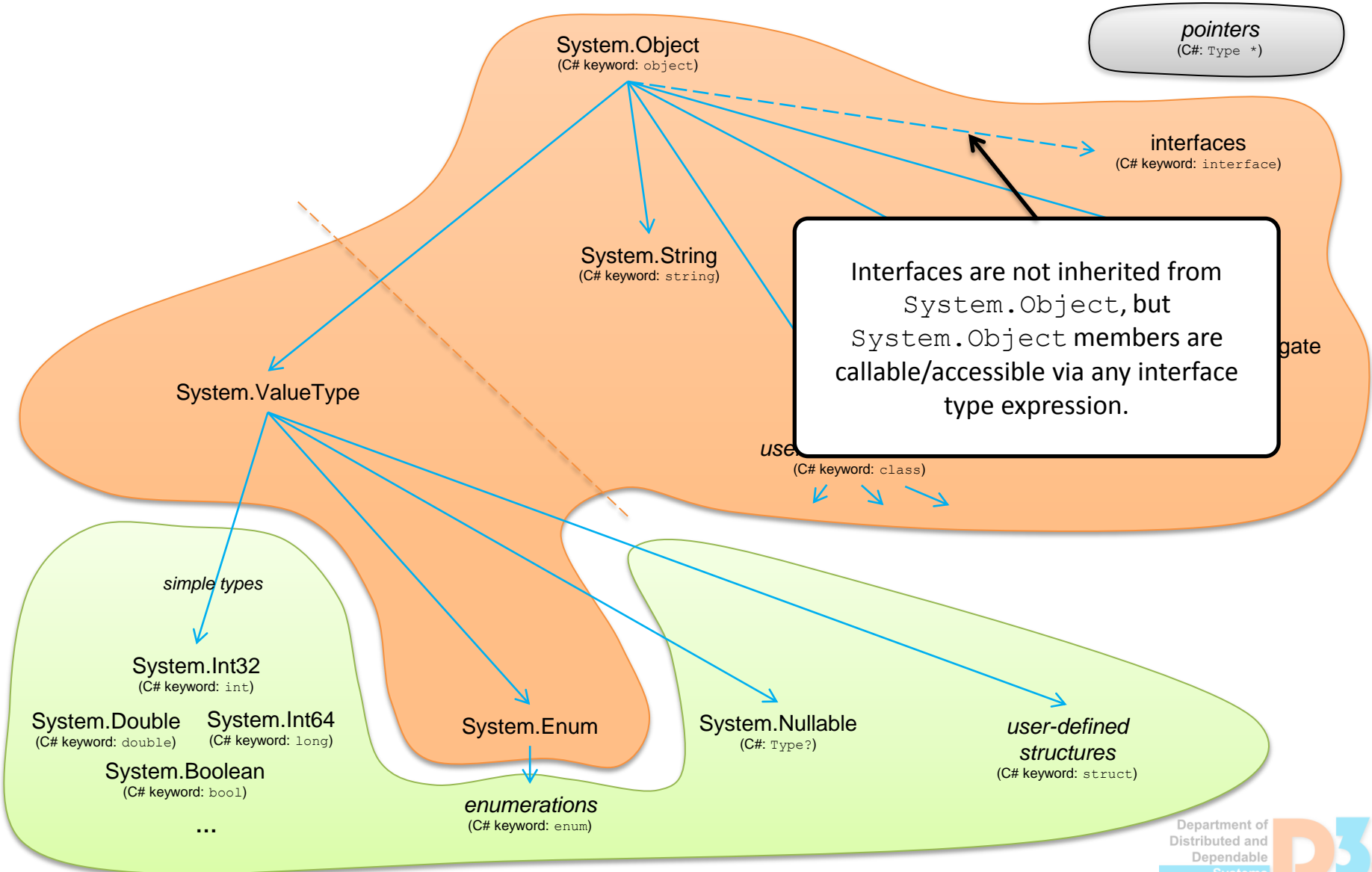
CLI Type System



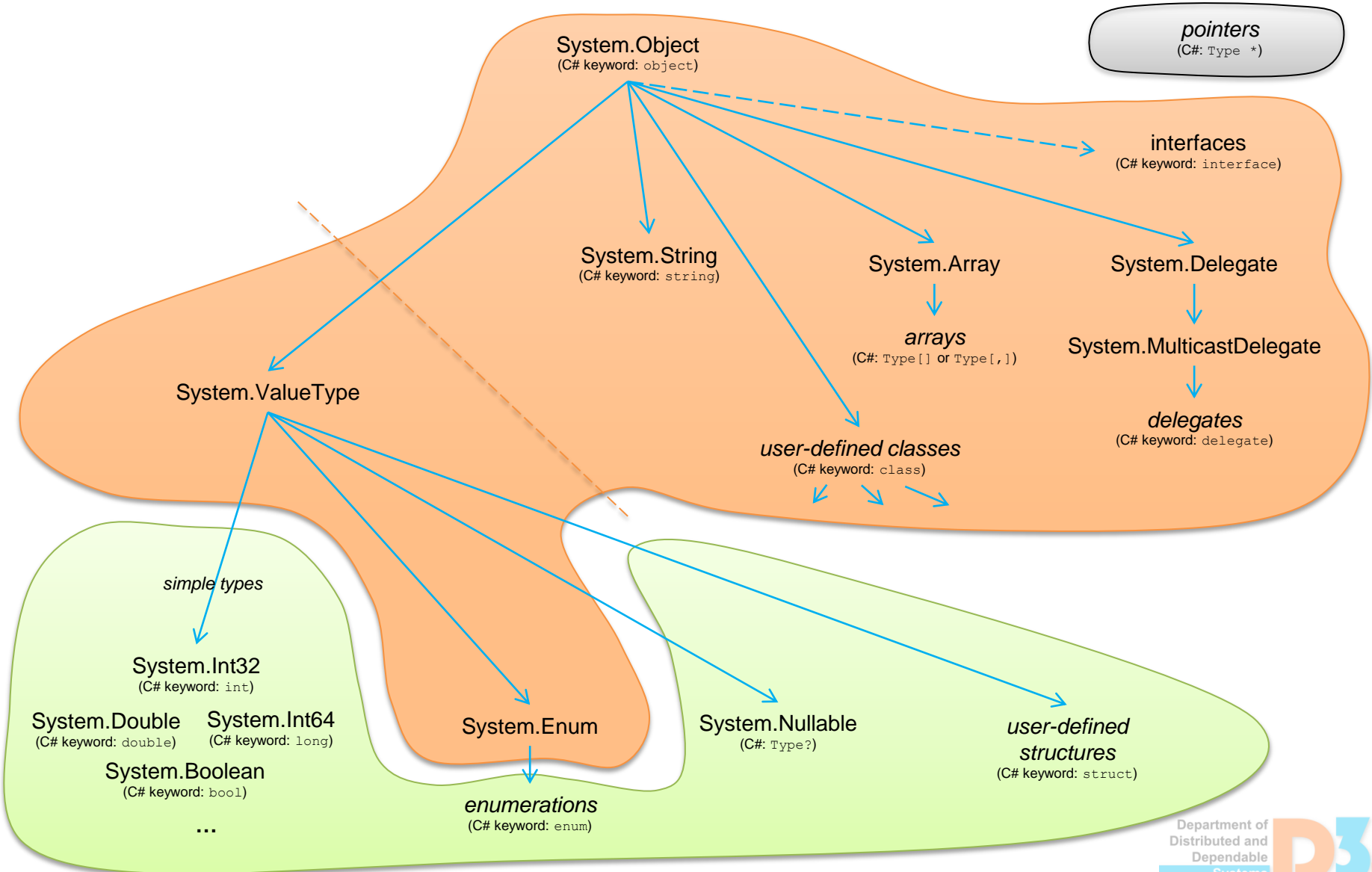
CLI Type Inheritance



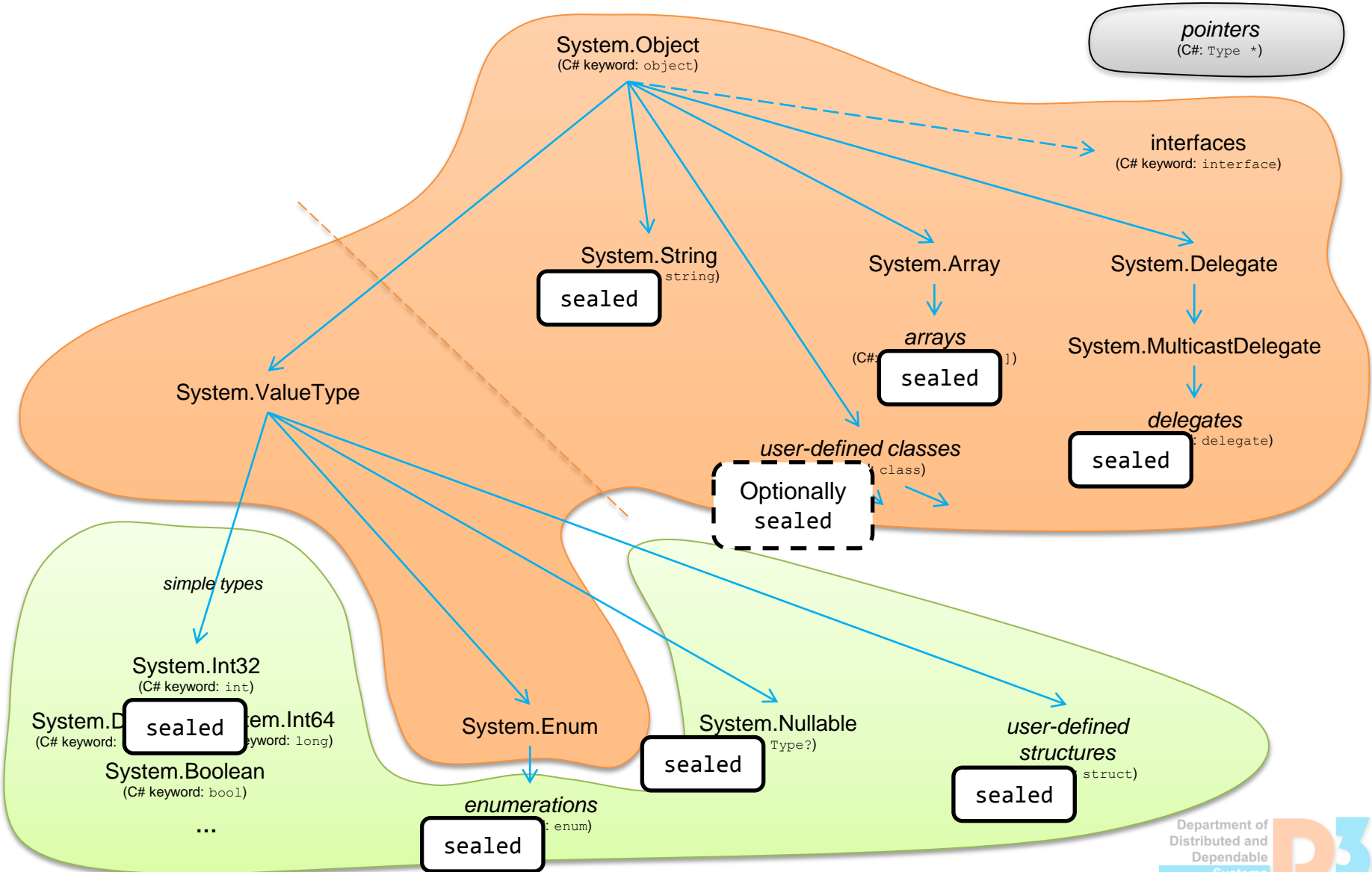
CLI Type Inheritance



CLI Type Inheritance



CLI Type Inheritance (Sealed Types)



Crossing Value/Reference Type Boundary

