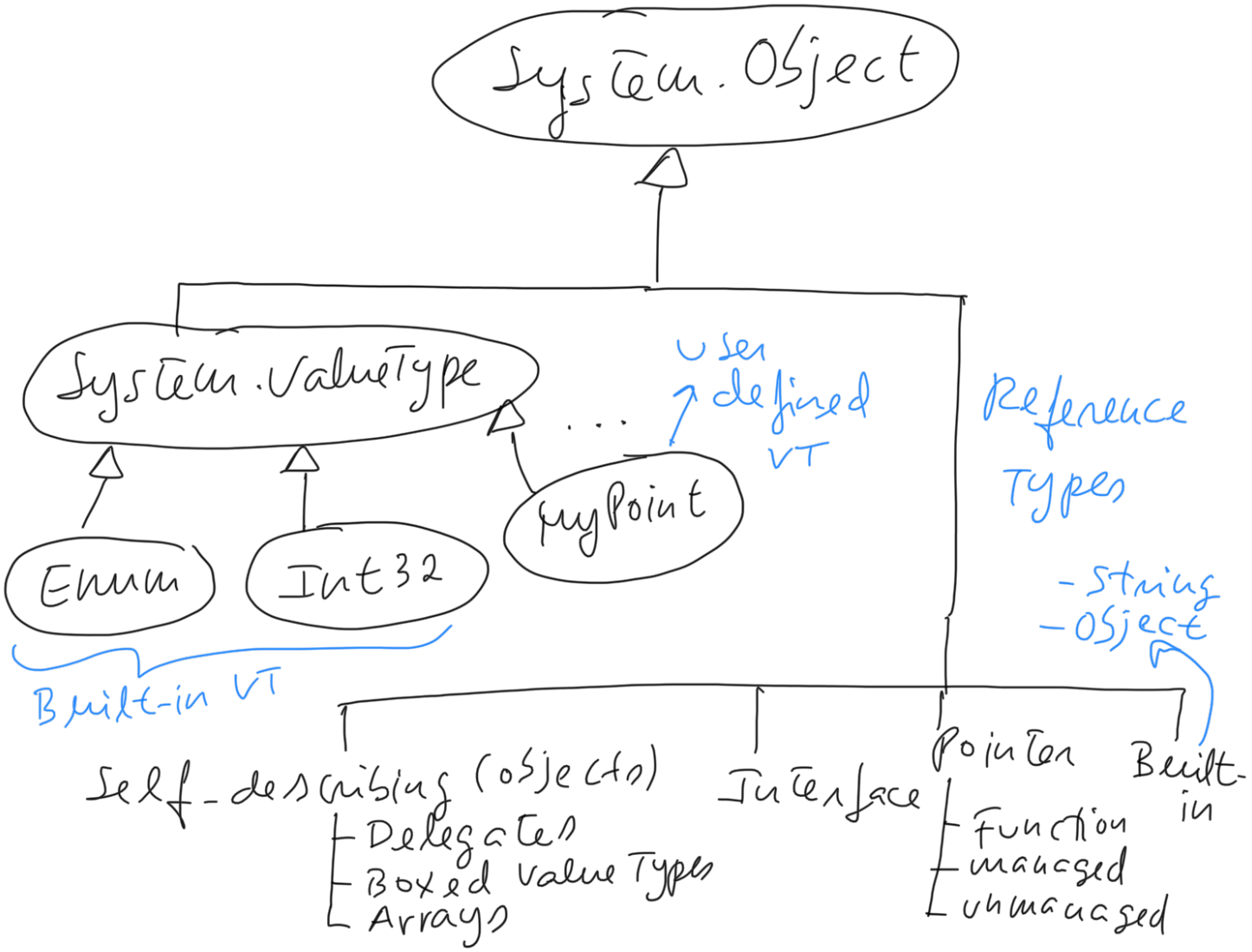
CTS - Common Type System

1.

- Especifica a forma como os tipos são definidos e se comportam
- Define duas categorias de tipos:
  - Reference Types (RT) - são sempre alojados no heap
  - Value Types (VT) - são alojados in-place
    - ↳ no stack frame
    - ↳ ou no espaço reservado no objeto (no heap) que contenha o campo de tipo valor

- Instâncias de VT também então no heap se forem um elemento dum array ou se forem boxed.



Sejam R e V tipos referência e tipos valor, respetivamente:

R r = ... ; → r é uma referência para um objeto no heap com o tipo R.

V v = ... ; → v contém a instância de V

Instâncias de VT não têm associados o overhead dos objetos
- não têm referência ln type e

↓
Type object ptr

sync#

↓
Sync block index

- não são alojados no heap, nem a sua memória reciclada pelo GC

Ambos os tipos valor e tipo referência podem ter membros como campos e métodos:

```
string s = 53.ToString();
```

`string` ⇧ `System.String`

`53` → instância de `System.Int32`

- Os tipos VT não podem ser estendidos (são sealed)

Em C#:

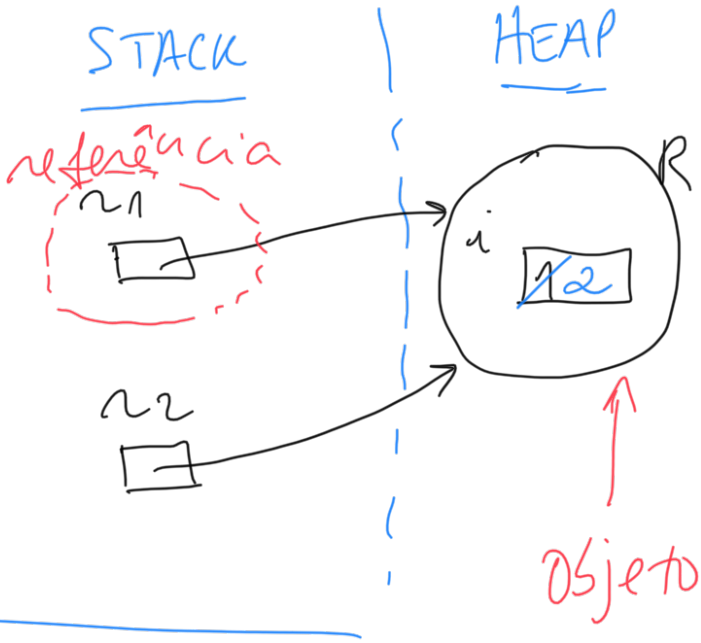
`class R { }` → RT

`struct V { }` → VT

Manipulação de VT e RT

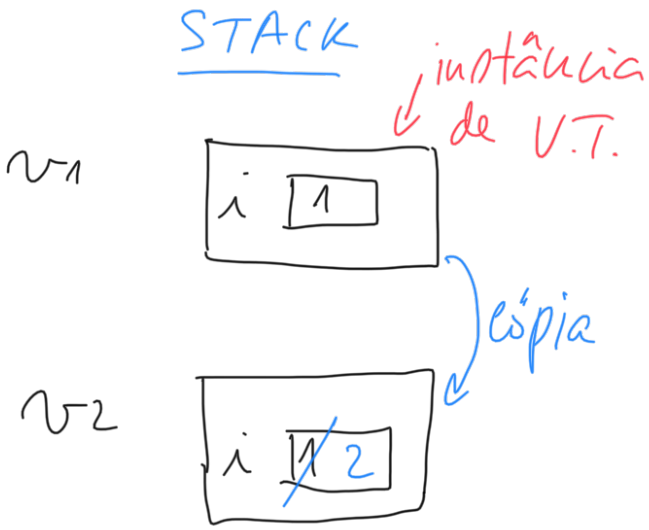
R.T.

```
R r1 = new R();
r1.i = 1;
R r2 = r1;
r2.i = 2;
Console.Write(r1.i); // 2
```



V.T.

```
V v1 = new V();
v1.i = 1;
V v2 = v1;
v2.i = 2;
Console.Write(v1.i); //1
```



		RT	VT
Alojamento		managed heap	in-place
Tempo de vida	Início	criação explícita	var. local / argumento / campo
	Fim	destruição implícita pelo GC	" "
cópia / afetação		cópia da referência	cópia do valor

Quando var. local / argumento ou campo são criados (início) ou destruídos (fim)