ISEL

Ambientes Virtuais de Execução

2021

Week 11 – Generics

Chapter 12 of CLR via C# (Jeffrey Richter)

Generics

Goals:

- Type safety => avoid cast

```
~~return ((Student) o).Number > 47000;~~
```

- Expressiveness

!!! Does not express the type of elements in the sequence! => object !!!

```
IEnumerable items = Convert(Lines("isel-AVE-2021.txt"), o => Student.Parse((string) o));
```

- Performance
  - ⇒ Avoid IL `castclass` that includes a type check compatibility that impacts performance
  - ⇒ When we deal with value type elements, we are avoiding box/unbox

Not for every VM. For example JVM does not support generics at runtime, but only at Java language level instead!

Generics

Goals:

- Type safety

```
return o.Number > 47000;
```

- Expressiveness

```
IEnumerable<Student> items = Convert(..., o => Student.Parse((string) o));
```

- Performance

  ⇒ Avoid IL `castclass` that includes a type check compatibility that impacts performance

  ⇒ When we deal with value type elements, we are avoiding box/unbox

Not for every VM. For example, JVM does not support generics at runtime, but only at Java language level instead!

Generics

- Parametrized (Generic) Types (i.e. classes, interfaces or delegates)

  => types that receive a type parameter.

```
interface I<U> {}

class A<T> { }
```

Open Type

Type argument

```
A<String> a1;
A<int> a2;
A<Point> a3;
```

Closed Type

- **Parametrized (Generic) Methods**

  ⇒ methods that receive a type parameter

Formal parameters

```
public int Bar<U>(T msg, U other) {…
```

⇒ Calling a generic method:

```
a1.Bar<Point>("super", new Point());
```

Generics constraints

- Example

```
class A<T> where T : IComparable{
```

Generic Delegates

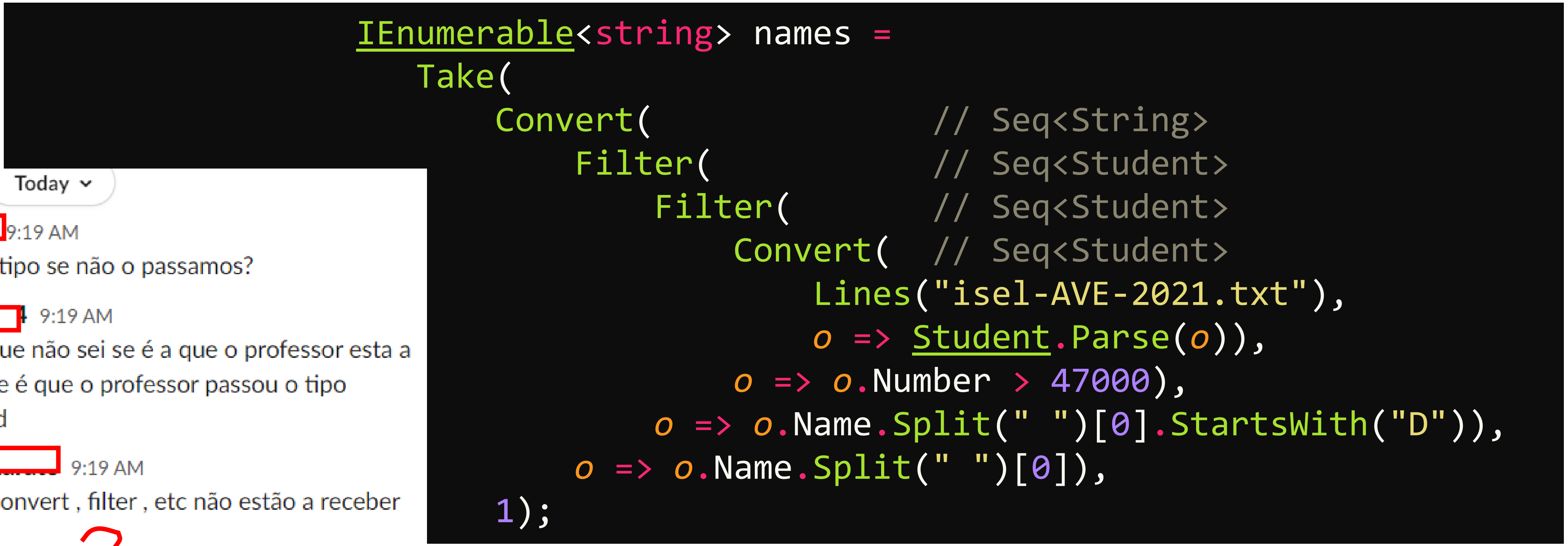
```
Predicate<T>
```

⇔

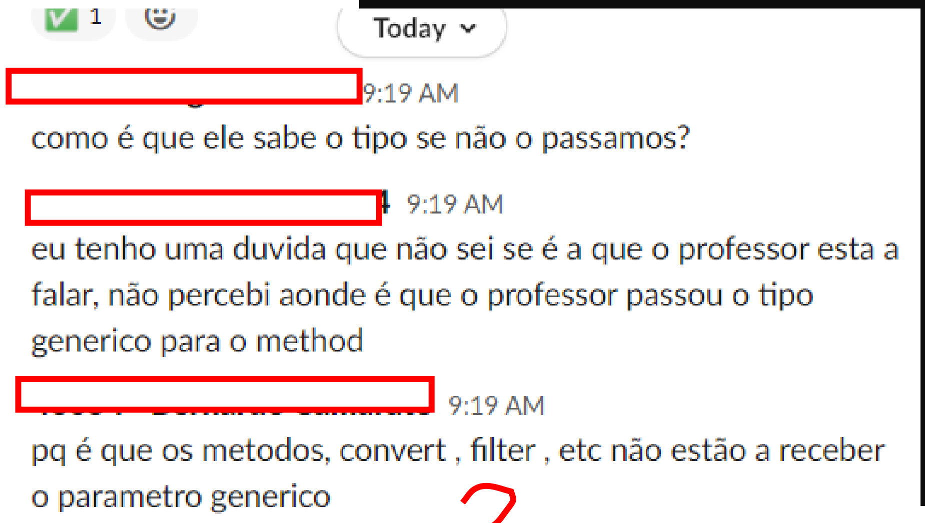
```
Func<T, bool>
```

Type inference

Where are the type arguments?



```
IEnumerable<string> names =
   Take(
       Convert(              // Seq<String>
           Filter(           // Seq<Student>
               Filter(       // Seq<Student>
                   Convert(  // Seq<Student>
                       Lines("isel-AVE-2021.txt"),
                       o => Student.Parse(o)),
                   o => o.Number > 47000),
               o => o.Name.Split(" ")[0].StartsWith("D")),
           o => o.Name.Split(" ")[0]),
       1);
```



[redacted] 9:19 AM
como é que ele sabe o tipo se não o passamos?

[redacted] 9:19 AM
eu tenho uma duvida que não sei se é a que o professor esta a falar, não percebi aonde é que o professor passou o tipo generico para o method

[redacted] 9:19 AM
pq é que os metodos, convert , filter , etc não estão a receber o parametro generico

?

Type inference

```
static IEnumerable<T> Filter<T>(IEnumerable<T> stds, Predicate<T> pred)
```

- T can be anything
- Must be the SAME type in stds and pred

Implicit

```
Filter<Student>(
    Convert(                                // IEnumerable<Student>
        Lines("isel-AVE-2021.txt"),
        o => Student.Parse(o)),
    o => o.Number > 47000),                 // o is Student too
```

Type inference

Infered from Lines

Infered from the return type of Student.Parse

```
        Convert<String, Student>(   // Seq<Student>
                        Lines("isel-AVE-2021.txt"),   // Seq<String>
                        o => Student.Parse(o)),
```