

# Generics

- Generics permitem que **classes**, **interfaces** e **métodos** possam ser parametrizados por tipo. Seus benefícios são:
  - Reuso
  - Type safety
  - Performance
- Uso comum: coleções

```
List<string> list = new List<string>();  
list.Add("Maria");  
string name = list[0];
```

## Problema motivador 1 (reuso)

Deseja-se fazer um programa que leia um conjunto de N números inteiros (N de 1 a 10), e depois imprima esses números de forma organizada conforme exemplo. Em seguida, informar qual foi o primeiro valor informado.

How many values? 3

10

8

23

[10, 8, 23]

First: 10

Criar um serviço de impressão:

PrintService
+ addValue(value : int) : void
+ first() : int
+ print() : void

## Problema motivador 2 (type safety & performance)

Deseja-se fazer um programa que leia um conjunto de N números inteiros (N de 1 a 10), e depois imprima esses números de forma organizada conforme exemplo. Em seguida, informar qual foi o primeiro valor informado.

How many values? 3

10

8

23

[10, 8, 23]

First: 10

Criar um serviço de impressão:

PrintService
+ addValue(value : object) : void + first() : object + print() : void

## Solução com generics

Deseja-se fazer um programa que leia um conjunto de N números inteiros (N de 1 a 10), e depois imprima esses números de forma organizada conforme exemplo. Em seguida, informar qual foi o primeiro valor informado.

How many values? 3

10

8

23

[10, 8, 23]

First: 10

Criar um serviço de impressão:

PrintService<T>
+ addValue(value : T) : void + first() : T + print() : void

<https://github.com/acenelio/generics1-csharp>