## Generics

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

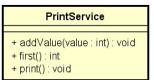
```
List<string> list = new List<string>();
list.Add("Maria");
string name = list[0];
```

## 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:

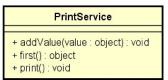


## Problema motivador 2 (type safety & performance)

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:

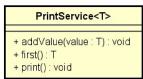


## Solução com generics

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:



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