



# Módulo 22





# BackEnd Java

---

Jeferson Tigik



A decorative graphic on the left side of the slide. It features a large central hexagon with a blue-to-cyan gradient, containing the number '1'. Surrounding this central hexagon are several smaller hexagons of varying shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and radiating lines, and a speech bubble icon. The entire graphic is set against a dark blue background.

1

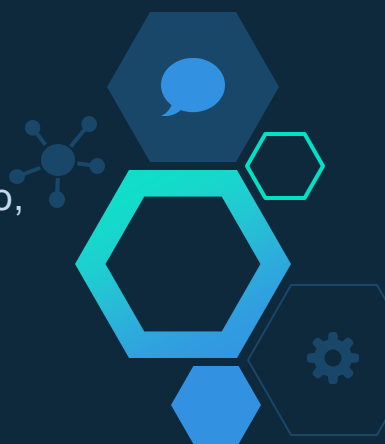
Generics



# O que são?

Generics é uma funcionalidade introduzida na versão 5 do Java. Seu objetivo é fornecer ao desenvolvedor a capacidade de escrever código que seja reutilizável e ao mesmo tempo com a segurança da verificação de tipos em tempo de compilação. As APIs da própria linguagem utilizam largamente estas funcionalidades, como é o caso das conhecidas interfaces List e Map por exemplo.

O **Generics** é delimitado pelos caracteres “<>”, ou seja, quando houver esse par de caracteres em uma parte qualquer do código, significa que o Generics está sendo utilizado





# Exemplo

```
List<String> listaDeStrings = new ArrayList<String>();
```





# O que são?

Generics também provê em tempo de compilação uma verificação de type-safety de código, removendo riscos de `ClassCastException` durante a execução, o qual era um erro comum antes da versão 1.5.

Essa verificação consiste em verificar se o que está sendo atribuído a uma instância de uma classe está de acordo com o especificado





# Exemplo

// Antes do Java 1.5


////////////////////////////////////

```
List palavras = new ArrayList();  
palavras.add("ABC");//Tudo ok, compatível com o especificado  
palavras.add(12); //Erro de ClassCastException em tempo de  
execução
```

// Após o Java 1.5

////////////////////////////////////

```
List<String> palavras = new ArrayList<String>();  
palavras.add("ABC");//Tudo ok, compatível com o especificado  
palavras.add(12); //Erro de compilação
```





# Onde são utilizados Generics

- Generics em Classes e Interfaces
- Generics em Métodos
- Generics em Construtores







# Generics em classes e interfaces

```
public interface List<T> extends Collection<T>{  
  
    ...  
  
}
```





# Generics em métodos

```
public <T> void getFirst(List<T> list) {  
    ...  
}
```





# Generics em construtores

```
public class GenericEntry<T> {  
    private T data;  
    private int rank;  
}  
  
public GenericEntry(T data, int rank) {  
    this.data = data;  
    this.rank = rank;  
}
```





# Wildcards

Existem 3 tipos de Wildcards em Generics:

- Unknown Wildcard, ou seja, Wildcard desconhecido
  - ◆ `<?>`
- Extends Wildcard
  - ◆ `<? extends XXX>`
- Super Wildcard
  - ◆ `<? super XXX>`

