11

Para saber mais: Generics

Em Java, generics permitem criar classes, interfaces e métodos que podem trabalhar com tipos desconhecidos ou parâmetros genéricos. Eles fornecem uma forma de escrever código flexível e reutilizável, tornando-o independente de tipos específicos e permitindo que ele funcione com diferentes tipos de dados.

Para criar um método ou classe genérico, você precisa usar parâmetros de tipo (tipos genéricos) que são representados entre colchetes angulares `< >`. Geralmente, usamos letras maiúsculas únicas para representar os tipos genéricos, mas você pode usar qualquer identificador válido em Java. Aqui está um exemplo de uma classe genérica chamada Caixa, que armazena um valor de um tipo desconhecido:

```
public class Caixa<T> {
    private T conteudo;

    public T getConteudo
        return conteudo;
    }

    public void setConte
        this.conteudo =
    }
}
```

COPIAR CÓDIGO

No exemplo acima, podemos criar um objeto do tipo Caixa e armazenar qualquer tipo de valor no mesmo, veja um exemplo:

```
public class TestaCaixa
    public static void m
        Caixa<String> ca
        caixaDeTexto.set

        Caixa<Integer> c
        caixaDeIdade.set

        Caixa<Double> ca
        caixaDeValor.set
    }
```

```
}
```

COPIAR CÓDIGO

Observe que podemos utilizar a classe caixa para incluir valores de tipos diferentes. Para a variável caixaDeTexto, o compilador irá garantir que apenas valores do tipo `String` possam ser armazenados. Ja para a variável caixaDeIdade, o compilador irá garantir que apenas valores do tipo `Integer` possam ser armazenados, e assim sucessivamente.

Método Genérico

Para criar um método genérico, você pode usar a mesma sintaxe com parâmetros de tipo entre colchetes angulares. Aqui está um exemplo de um método genérico, que pertence à classe Caixa e que deverá somar o `valor` passado por parâmetro ao conteúdo da caixa:

```
    public <T> T somaConteu
        if (this.conteud
            Integer resul
            return (T) re
        } else if (this.
            Double resul
            return (T) r
        } else if (this.
            String resul
            return (T) r
        }

        return null;
    }
```

COPIAR CÓDIGO

O objetivo do método acima é realizar a soma entre o conteúdo atual da caixa (this.conteudo) e o valor passado como parâmetro (valor). O método é genérico e pode ser usado para diferentes tipos de conteúdo que podem ser somados, como Integer, Double e String.

Vamos descrever o que acontece passo a passo:

```
public <T> T somaConteud
```

COPIAR CÓDIGO

O método é genérico e recebe um parâmetro valor do tipo genérico T, que é o mesmo tipo que será retornado como resultado da soma.

Logo em seguida, o método começa com uma série de condicionais `if` que verificam o tipo do conteúdo atual da caixa (this.conteudo) e o tipo do valor passado como parâmetro (valor).

```
if (this.conteudo instan
    // Realiza a soma en
    Integer resultado =
    // Retorna o resulta
    return (T) resultado
}
```

COPIAR CÓDIGO

A verificação é realizada usando os operadores `instanceof` e os operadores de pattern matching (instanceof

com pattern variables) disponíveis a partir do Java 16.

Se o conteúdo atual (this.conteudo) e o valor (valor) forem ambos do mesmo tipo, é feita a soma ou a concatenação, como no caso da String. Caso o tipo da variável valor seja diferente do tipo do conteúdo, devolvemos o valor anterior do conteúdo. Vejamos como ficaria em nossa classe TestaCaixa:

```
public static void main(
        Caixa<String> ca
        caixaDeTexto.set
        System.out.print

        Caixa<Integer> c
        caixaDeIdade.set
        System.out.print

        Caixa<Double> ca
        caixaDeValor.set
        System.out.print
        System.out.print
    }
```

COPIAR CÓDIGO

Ao executar o código acima, teremos como saída em nosso terminal, os seguintes valores:

```
Guardando texto na minha
Mais uma linha
56
501.0
null
```

COPIAR CÓDIGO

Repare que na última linha do código, ao tentar incluir uma String “texto” em nossa `caixaDeValor`, ao executar esse código tivemos um retorno `null`, pois só realizamos a soma caso ambos os tipos fossem iguais.