A PARTIR DO JAVA 10 HABEMUS INFERÊNCIA PARA VARIÁVEIS LOCAIS!

OLÁ!!

Ruan Victor

Baiano que passa frio em Porto Alegre

Guilherme Reis

Tô aqui só pra passar os slides e pq disseram q teria pão de queijo

AGENDA

- O que é inferência de tipo em variáveis locais?
- Quais as vantagens?
- Como funciona?
- Qual o impacto?
- Benefícios
- Limitações
- Utilizando *var* em funções lambda

O QUE É INFERÊNCIA?

INFERÊNCIA É UMA DEDUÇÃO FEITA COM BASE EM INFORMAÇÕES OU UM RACIOCÍNIO QUE USA DADOS DISPONÍVEIS PARA SE CHEGAR A UMA CONCLUSÃO.

Inferência é uma dedução feita com base em informações ou um raciocínio que usa dados disponíveis para se chegar a uma conclusão.

Traduzindo... o compilador vai adivinhar o tipo da variável para você baseado na inicialização.

VEM PRO CÓDIGO, VEM!

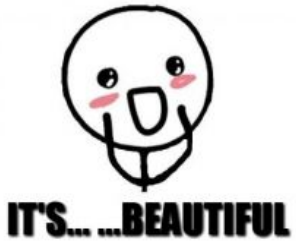
```
var numeroDez = 10;
var usuario = new Usuario();
var minhaLista = List.of("1","2");
```

+ LEGIBILIDADE

- VERBOSIDADE

```
//Antes do Java 10
QueueAccessRQ queueAccessRQ = new QueueAccessRQ();

QueueAccessRQ.Navigation navigation = new QueueAccessRQ.Navigation();

QueueAccessRQ.Navigation.Direction direction = new
QueueAccessRQ.Navigation.Direction();

InternationalCustomerOrderProcessor<AnonymousCustomer, SimpleOrder<Book>>
orderProcessor = createInternationalOrderProcessor(customer, order);
```

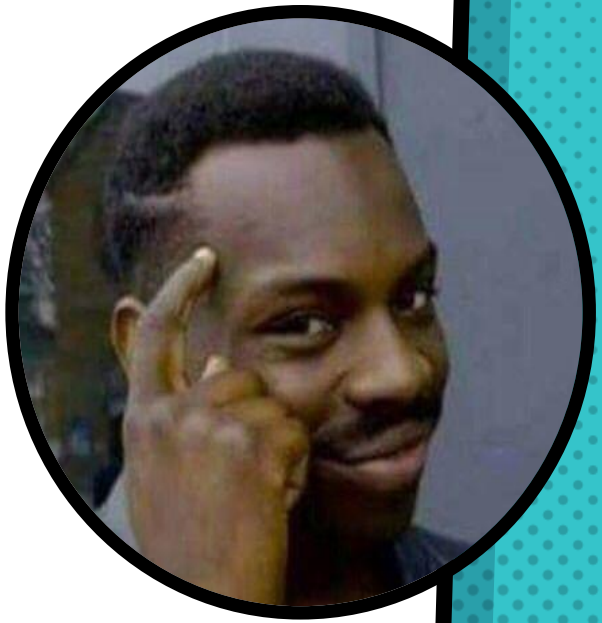
```java
//A partir do Java 10
var queueAccessRQ = new QueueAccessRQ();

var navigation = new QueueAccessRQ.Navigation();

var direction = new QueueAccessRQ.Navigation.Direction();

var orderProcessor = createInternationalOrderProcessor(customer,
order);
```



COMO FUNCIONA ?

Ao encontrar a palavra *var* no teu código, o compilador vai saber que o tipo da variável vai estar no lado direito da declaração, também conhecida como inicializador.

Na hora de compilar esse código ele irá inferir o tipo do lado esquerdo da declaração.



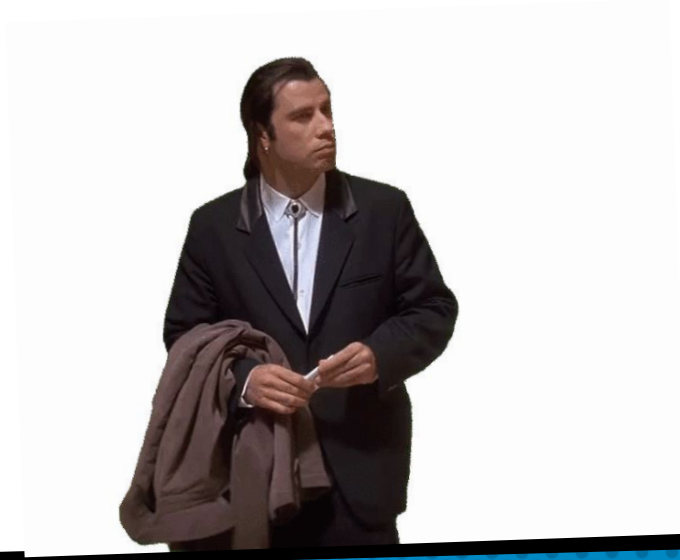
VEM PRO CÓDIGO, VEM!

```
//como escrevemos
var numeroDez = 10;
var user = new User();
var list=List.of("1","2");

//depois de compilado
int numeroDez = 10;
User user = new User();
List<String> list = List.of("1","2");
```

QUAL O IMPACTO?

QUAL O IMPACTO?

- **Desempenho:**
  Nenhum!



QUAL O IMPACTO?

- **Compatibilidade com códigos antigos:** Só para classes ou interfaces com o nome ***var***

```
var var = 5; //sintaticamente correto
package var; //sintaticamente correto
class var{ } //erro de compilação
interface var{ } //erro de compilação
```

Benefícios

- Aumenta a clareza do código
- Reduz o *boilerplate*
- Melhora a experiência do desenvolvedor

Limitações

```
var nome;
var numero = null;
var usuario = "Bojack", sobrenome = "Horseman";
var appendSpace = a -> a + " ";
var compareString = String::compareTo;
public var getNomeDoFilho(){
   return this.nomeDoFilho;
}
```

UTILIZANDO VAR EM FUNÇÕES LAMBDA

- O Java 11 trouxe com ele uma atualização na funcionalidade de inferência de tipo. Agora é possível que *var* seja usado para declarar os parâmetros formais de uma expressão lambda implicitamente tipada.

VANTAGENS

Com isso podemos utilizar anotações de tipo. Essa anotações adicionam metadados que ajudam a reduzir o número de erros no código. Baseado nessas anotações, o compilador pode emitir alertas ao concluir que o código não atende a certos requerimentos.

VEM PRO CÓDIGO, VEM!

```
(@Nonnull var valor, @Nonnull var conversor) ->
conversor.converter(valor);


//Ao executar o código acima teríamos a oportunidade de capturar o
seguinte erro em tempo de compilação
Null type mismatch: required ''@NonNull Double'' but the provided value
is null
```

VLWS!

rvsilva@thoughtworks.com
greis@thoughtworks.com