

A classe wrapper embrulha o tipo primitivo dentro dela para que a gente consiga tratá-los como objeto

Com wrappers, transformamos tipos primitivos em classes de objetos.

tipo primitivo

wrapper

int	→	Integer
long	→	Long
float	→	Float
double	→	Double
char	→	Character
boolean	→	Boolean

Para setar o valor em qualquer wrapper, basta usar o método `valueOf()`.

Um dos pontos fortes dos wrappers são as conversões, que podem ser feitas da seguinte forma.

```
Integer diasEntrega = Integer.valueOf(30)  
Short short = Short.valueOf(diasEntrega.shortValue())
```

↙  
conversão

A partir do Java 5, não é mais necessário o `valueOf()`, pois o compilador já faz isso de forma automática, então basta setar como se fosse de forma normal.

Esse processo recebe o nome de autoboxing

unboxing = Setar um wrapper em um tipo primitivo.

Para comparar objetos, utilize-se o `.Equals()`  
mas só pode comparar objetos do mesmo  
tipo

# Cuidados:

## 1- Comparação entre wrappers

↳ Outra forma de comparar é usando o `compareTo`...

```
numero1.compareTo(numero2) == 0;
```

↳ Se for 0, é `true`.

## 2- Com wrapper é mais fácil tomar nullpointer

## 3- NO Boxing e unboxing vai perdendo eficiência, é menos performático.

↳ Sempre que possível, utilizar tipos primitivos, porque? São mais simples e mais rápidos

# Quando vale a pena usar tipos Primitivos?

- Quando não for possível usar tipo Primitivo, exemplo: coleções, generics
- Quando precisa ser atribuído valor Null na variável, porque tipos primitivos não aceitam valor Null.

\* 0 zero não significa a falta de informação.

Ex:

valor Renda = null (não tem renda)

valor Renda = 0.0 (a renda é zero reais)