

# Java 講座

第2回



# 目次



1. データ型
2. 型推論
3. 型変換

# データ型

---



# データ型



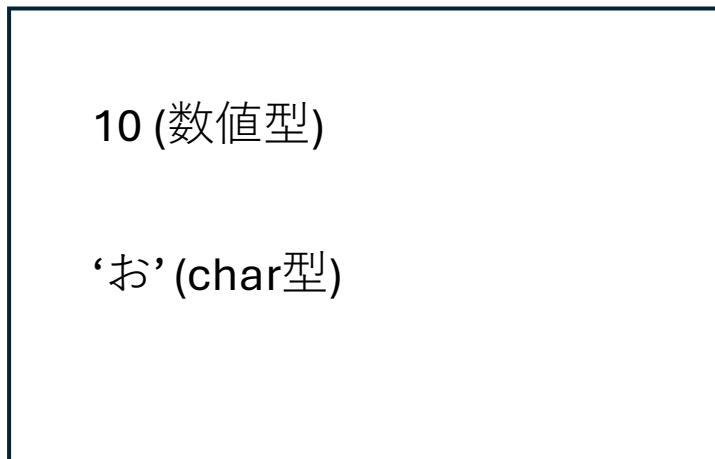
- プログラムの中で扱うデータの種類のこと
- そのデータが数値なのか、文字列なのか、それ以外なのか。数値だったらどの範囲までなのかと言った事柄を決めるのが、型の役割
- Java は **静的型付け言語** という種類のプログラミング言語なので、**変数を宣言した時点で型も決定します**
  - 反対に、変数を宣言するときに型が決定しない言語を **動的型付け言語** と言います

# データ型の分類

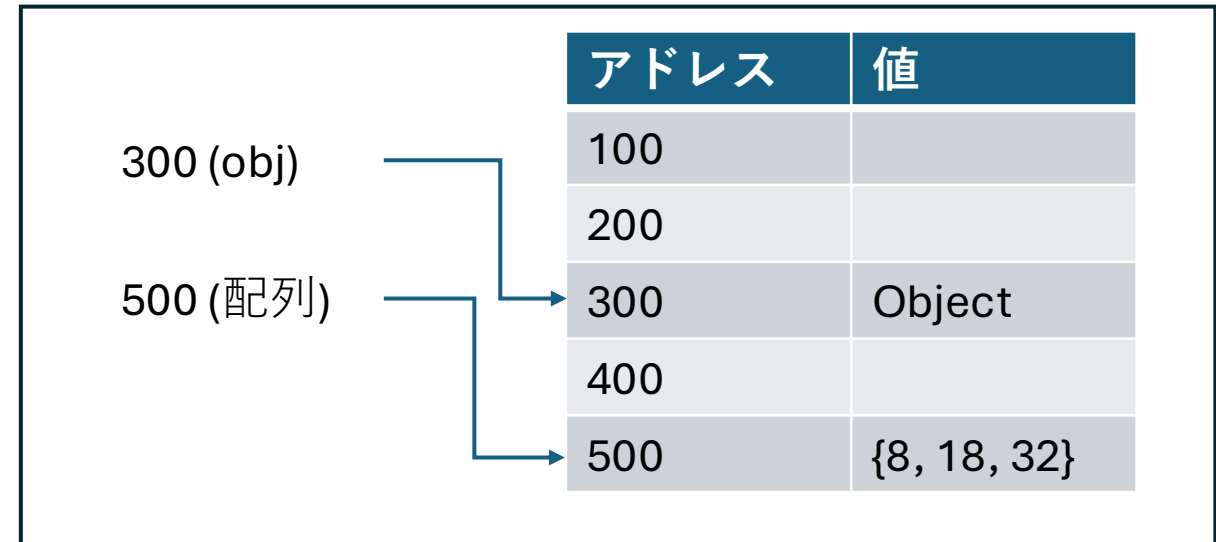


- データ型は大まかに分けて 基本型 と 参照型 にわけることができます
- 両者の違いは 値を格納する方法 にあります
- 基本型: 値そのものを格納する
- 参照型: 値の格納場所を示す情報を格納する (メモリアドレスの情報)
  - 実際の値は別の場所に格納される

基本型



参照型



# 基本型



- Java の基本となるデータ型
- 基本型は プリミティブ型 (原始型) と呼ばれることがある
  - これはその構造がシンプルで原始的だから

分類	データ型	サイズ	範囲
整数	byte	1バイト	-128~127
	short	2バイト	-32768~32767
	int	4バイト	-2147483648~2147483647
	long	8バイト	-9223372046854775808~9223372036854775807
浮動小数点	float	4バイト	
	double	8バイト	
真偽	boolean	1ビット	True / False
文字	char	2バイト	Unicode

# 型推論

---



# 型推論



- コンパイラが自動で型を予測（推論）して決定してくれる機能
  - Java では Java 10 から使えるようになった

```
String name = “Taro”;
```

↓

```
var name = “Taro”;
```



# なんでもありではない



- あくまでコンパイラがプログラマの代わりに型を決定してくれるだけなので、なんでもありのデータ型というわけではない
- 前ページの変数例でいうと、name に数値を入れようとするとエラーになる

```
var name = "Taro";
```

```
name = 1; // エラー
```

# 型変換

---



# 型変換



- Java は型を厳密に区別する
- そのため、数値を無条件に文字列に変換することはできない
- 例外的に異なる型への代入が許されている場合がある

```
int num = 108;
```

```
String str = num; // エラー
```

# 暗黙的な型変換



- 型変換には大きく分けて 暗黙的な型変換 / 明示的な型変換 というものがある
- 暗黙的な型変換が許されるのは **数値型で、値範囲が狭い型から広い型へ代入する場合** になる
- またこの場合を 拡大変換 という

```
int num = 10;
```

```
long longNum = num;
```

# 明示的な型変換



- 一方で、広い型から狭い型への型変換（これを縮小変換という）は値にかかわらずエラーになる
- ただ、値の範囲内なのに型変換できないのはちょっと使いづらい…
- そこで Java では型変換の意思を宣言した場合のみ型変換を許す仕組みが採用されていて、これが 明示的な型変換 となる
- 型キャスト（キャスト）構文を使って型変換を宣言する

```
int num = 10;
```

```
byte bNum = (byte) num;
```