4章 データの扱い方を理解しよう

Javaで使われるデータの種類を知り、Javaで簡単な計算や文字列の連結を行います。

60分 - 未読

4.1 本章の目標

本章では以下を目標にして学習します。

- Javaで使われるデータの種類（データ型）を知ること
- Javaで簡単な計算や文字列の連結を行うこと

Javaに限らず、プログラミングでは当たり前のように「データ」を扱います。データとは、プログラムにおける数値や文字列といった情報の総称です。

データを知らないと、ほかのJava文法も理解できません。今後の学習を大きく左右する部分であるため、Javaで使われるデータの種類や簡単な計算などは、しっかり学びましょう。

4.2 データの種類（データ型）

Javaをはじめとするプログラミング言語では、さまざまなデータを扱います。

- 「こんにちは」のような文字列データ
- 「100」のような数値データ
- 「2022/12/31」のような日付データ

プログラミング言語で扱う、このようなデータの種類を**データ型**と呼びます。Javaのデータ型は数多くありますが、特に基本的な以下の4つをまず押さえましょう。

1. 整数型
2. 浮動小数点型
3. 論理型
4. 文字列型

なお、データ型をEclipseで実践学習する場合は、前章で作成したプロジェクトを使えばOKです。プロジェクトに**パッケージ「text.section_04」を作成**し、その中にクラスファイルを順番に作成していきましょう（作成方法は3章を参照）。

では順番に解説します。

（1）整数型

整数型は「100」「0」「-123」など、整数を表すデータ型のことです。前章の「Hello World」と同様に、整数を指定すれば同じようにコンソールに表示できます。

試しに、text.section_04の中にクラスファイル「Data_1.java」を作成して、以下のようにソースコードを書きましょう。

Data_1.java

```
1  package text.section_04;
2
3  public class Data_1 {
4      public static void main(String[] args) {
5          System.out.println(100);
6          System.out.println(0);
7          System.out.println(-123);
8      }
9  }
10
```

なお、上記の「100」のように、ソースコードに直接書かれた値のことを **「リテラル」** と呼びます。上記のコードを実行して、以下のように表示されることを確認しましょう。

```
1  100
2  0
3  -123
4
```

Javaの整数型には以下の4種類があり、それぞれ扱える数値の範囲が変わります。数値の範囲まで細かく覚える必要はありませんが、範囲の大小は覚えておいてください。

データ型	値の範囲	サイズ
byte	-128～127	8ビット
short	-32,768～32,767	16ビット
int	-2,147,483,648～2,147,483,647	32ビット

データ型	値の範囲	サイズ
long	-9,223,372,036,854,775,808～9,223,372,036,854,775,807	64ビット

上記の「サイズ」は、そのデータを使うために必要な **「ビット」** のケタ数を指します。ビットは情報の最小単位で、ビットごとに0か1が入ります。byte型なら、ビットが8ケタです。

データは「変数」（詳細は5章）という容器に入れて使いますが、データ型によって容器の大きさも変わります。扱う数値の大きさに合わせて、データ型を選びましょう。

たとえば、「123」を扱うならbyte型でOKですが、「1234」はshort～long型でなければ扱えません。整数型はよく使うため、4種類の存在をしっかり覚えておきましょう。

なお実際の開発では、取りうる値がすべて収まる範囲で、できる限りサイズの小さいデータ型を選ぶのが理想です。必要以上にサイズの大きいデータ型だと、ソフトウェアの資源を圧迫することもあります。

補足：大きな数値を扱う場合は「L」を付ける

整数型のリテラルはint型として扱われることに注意が必要です。たとえば、以下のように書いてプログラムを実行してみましょう。

Data_2.java

```
1  package text.section_04;
2
3  public class Data_2 {
4      public static void main(String[] args) {
5          System.out.println(2147483648);
6      }
7  }
8
```

すると、以下のようにエラーとなります。リテラルに指定した値が、int型の最大値（2,147,483,647）を超えているためです。

```
1  型 int のリテラル 2147483648 は範囲外です
2
```

これを解決したい場合は、**リテラルの末尾に「L」** を付けます。

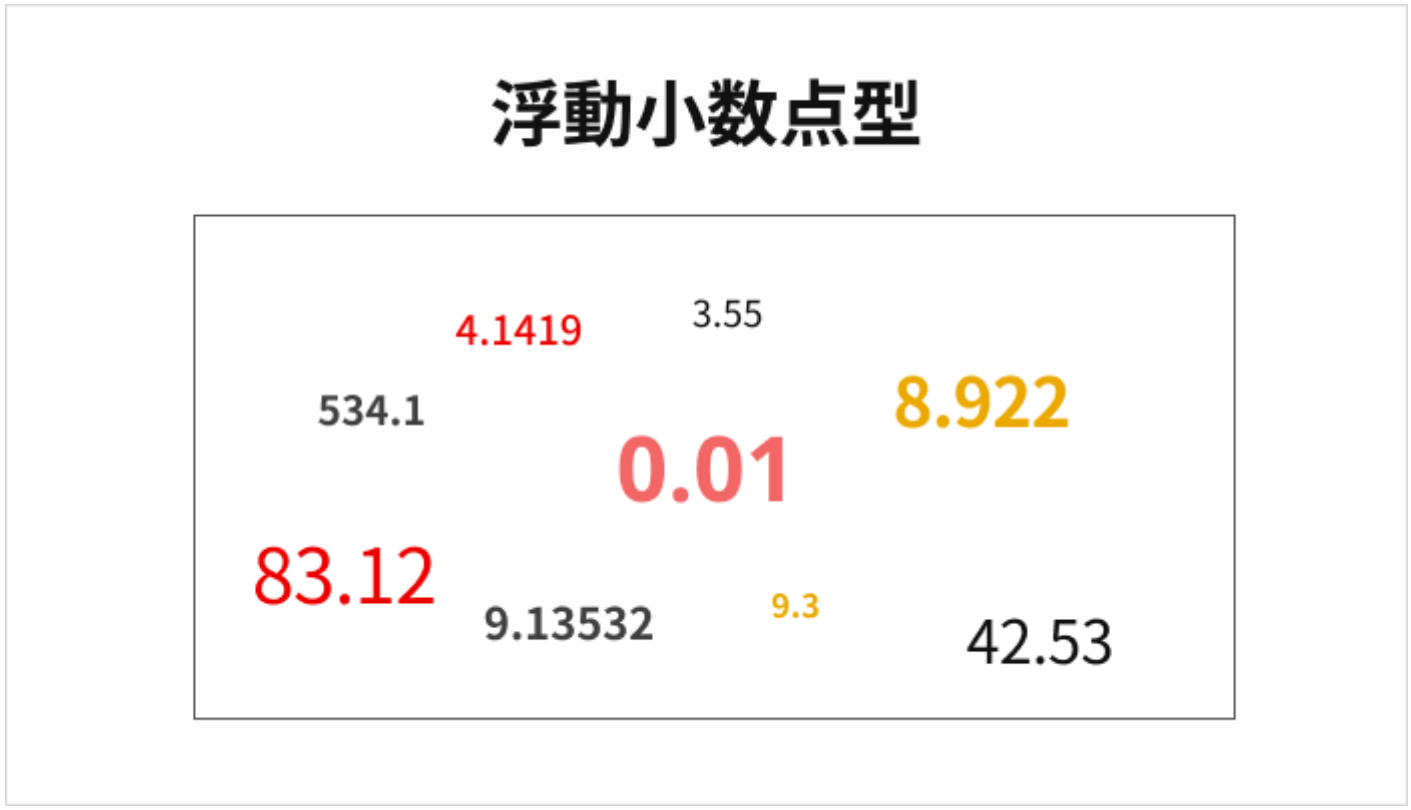
```
1  System.out.println(2147483648L);
2
```

Lを付けたリテラルはlong型で扱われます。int型の最大値を超えた数値も扱えるようになるため、エラーなく実行できるのです。

```
1  2147483648
2
```

このように、末尾に付けるキーワードを **「サフィックス」** と呼びます。リテラルで大きな数値を扱うときは、サフィックスとしてLを付けると覚えておきましょう。

（2）浮動小数点型



浮動小数点型は「0.1」「1.618」など、1よりも小さい数を含んだ数値のデータ型です。間のどこかに小数点（.）を1つだけ含んだ数値ともいえますね。

整数型のときと同様に、以下を実行して確認しましょう。

Data_3.java

```java
package text.section_04;

public class Data_3 {
    public static void main(String[] args) {
        System.out.println(0.1);
        System.out.println(1.618);
        System.out.println(-9876.0123);
    }
}

```

実行結果

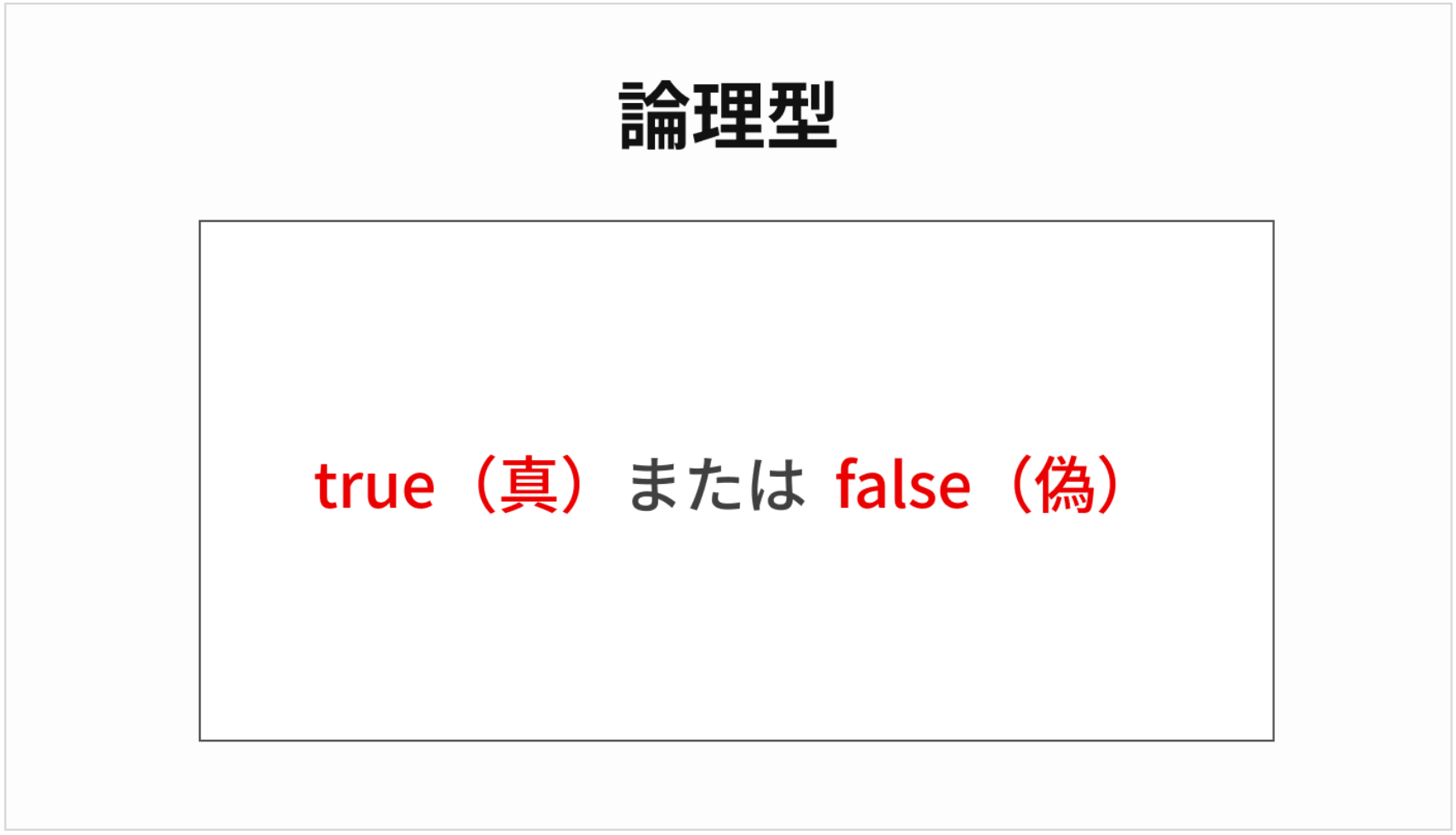
```
0.1
1.618
-9876.0123

```

Javaの浮動小数点型には以下の2種類があります。どちらも非常に大きな数値を扱えますが、範囲は覚えなくて構いません。double型のほうが扱える値の範囲が大きいことだけは覚えましょう。

データ型	値の範囲	サイズ
float	(-3.4028235 × 10の38乗)　～(3.4028235 × 10の38乗)	32ビット
double	(-1.7976931348623157 × 10の308乗)　～(1.7976931348623157 × 10の308乗)	64ビット

なお、**浮動小数点型のリテラルはdouble型扱い**となります。上記のように数値の範囲が非常に大きいため、データが収まりきらずに困ることはまずないでしょう。

（3）論理型



論理型はtrue（真）またはfalse（偽）、つまり「本当」か「嘘」かを表すデータ型です。「大人か子供（大人以外）か」のように、白黒はっきりしたデータの扱いに役立ちます。

Javaの論理型は「true」「false」のように、全て小文字で書いてください。「True」や「FALSE」だとエラーになるため注意が必要です。

Data_4.java

```java
package text.section_04;

public class Data_4 {
    public static void main(String[] args) {
        System.out.println(true);
        System.out.println(false);
    }
}
```

実行結果

```
true
false

```

Javaで論理型といえば、**boolean型**のことを指します。整数型のように、複数の種類はありません。

データ型	値の範囲	サイズ
boolean	true または false	1バイト※環境による

論理型のメリットは、5章で変数を学ぶときにわかります。今は、白黒はっきりしたデータを扱うときに使うのが論理型と覚えておきましょう。

（4）文字列型

文字列型は「こんにちは」「Hello, world!」など、文字列（文字の集まり）を表すデータ型です。Javaでは、ダブルクォーテーション（”）で囲むことで文字列型を扱えます。

Data_5.java

```
1  package text.section_04;
2
3  public class Data_5 {
4      public static void main(String[] args) {
5          System.out.println("こんにちは");
6          System.out.println("今日はいい天気ですね");
7      }
8  }
9
```

実行結果

```
1  こんにちは
2  今日はいい天気ですね
3
```

Javaの文字列型といえば**String型**のことを指しますが、ほかのデータ型と毛色が異なります。そのため、String型については5章で簡単な使い方を、18章で詳細を学びましょう。

なお、1文字だけのデータはシングルクォーテーション（’）で囲みます。

```
System.out.println('a');
System.out.println('あ');

```

実行結果

```
a
あ

```

これはあくまで「文字」であり、「文字列」ではありません。言語によっては文字列もシングルクォーテーションで囲めますが、Javaではエラーになるため注意が必要です。

Javaで**1文字だけを扱うときはchar型**を使います。ただし、charの値は文字コード（文字の識別番号）です。たとえば、「12354」という値の文字コードが「あ」となります。

データ型	値の範囲	サイズ
char	0～65,535※文字コードを表す	2バイト

文字列と文字で、それぞれデータの扱い方が異なることを覚えておきましょう。

4.3 簡単な計算（四則演算）をしてみよう

整数型や浮動小数点型のデータは、計算が行えます。数値同士で計算するときは、**計算したいデータの間に算術演算子（計算用の記号）を書く**のが基本です。

```
System.out.println(1 + 1);

```

上記のように、**演算子の両側には半角スペースを入れる**と読みやすくなります。入れなくても動作上は問題ありませんが、本章以降は半角スペースを入れる書き方で統一します。

実行結果

```
2

```

Javaの代表的な算術演算子は、以下の5つです。

算術演算子	処理の内容	例（出力結果）
+	左の値と右の値を足す（加算）	45 + 18（63）
-	左の値から右の値を引く（減算）	30 - 12（18）
*	左の値と右の値を掛ける（乗算）	15* 6（90）

算術演算子	処理の内容	例（出力結果）
/	左の値を右の値で割る（除算）	30 / 3（10）
%	左の値を右の値で割った余りを求める（剰余演算）	45 % 8（5）

これらの算術演算子を使って、実際に簡単な計算をしてみましょう。

整数型のみを使った計算

まずは、整数型のみを使った計算を実践しましょう。Eclipse上で以下のソースコードを自分で書き、実行してください。

Data_6.java

```
package text.section_04;

public class Data_6 {
    public static void main(String[] args) {
        System.out.println(45 + 18); // 加算
        System.out.println(30 - 12); // 減算
        System.out.println(15 * 6);  // 乗算
        System.out.println(30 / 3);  // 除算
        System.out.println(45 % 8);  // 剰余演算
    }
}
```

実行結果

```
63
18
90
10
5
```

上記のように、計算した結果が表示されればOKです。

整数・小数が混在する計算

整数同士、小数同士だけでなく、整数と小数が混在する計算も可能です。Javaでは、計算するデータに**小数が含まれるとき、計算結果は浮動小数点型**として扱われます。

では、浮動小数点型を含む計算をしてみましょう。以下のソースコードを自分で書き、実行してください。

Data_7.java

```java
package text.section_04;

public class Data_7 {
    public static void main(String[] args) {
        System.out.println(45 + 1.8); // 加算
        System.out.println(3 - 1.2);  // 減算
        System.out.println(15 * 0.6); // 乗算
        System.out.println(3 / 0.3);  // 除算
        System.out.println(45 % 0.8); // 剰余演算
    }
}

```

以下のように表示されればOKです。

```
46.8
1.8
9.0
10.0
0.1999999999999975

```

最終行の結果を見て「あれ？」と気になった人もいるでしょう。45を0.8で割った余りは「0.2」ですが、結果に若干のずれがありますね。このずれについて説明します。

補足：浮動小数点型は誤差が生じる場合がある

そもそも「0.8」は、プログラム上は「0.8の近似値（おおよその数値）」として扱われます。厳密には0.8ではないため、「45 % 0.8」の結果も少しずれてしまうのです。

たとえば99.9という小数は、コンピュータの内部的には99.90000000000000568434…という近似値になっています。**コンピュータは小数の扱いが苦手**なため、誤差が生じるのです。

消費税の計算など正確な結果が求められる場面では、浮動小数点型をそのまま使うことはできません。誤差を切り捨てるなど、専用の処理が必要となります。

ただし、多少の誤差が生じても問題ない場面（大まかな平均値が必要な場合など）では、浮動小数点型は便利です。浮動小数点型は誤差が生じる場合があると覚えておきましょう。

4.4 文字列を連結してみよう

文字列型のデータ同士は、先ほど使った「+」を使うことで連結できます。以下のソースコードを自分で書き、実行してください。

Data_8.java

```java
package text.section_04;

public class Data_8 {
    public static void main(String[] args) {
        System.out.println("侍" + "テラコヤ");
    }
}

```

以下のように、「侍」と「テラコヤ」が連結して表示されればOKです。

```
侍テラコヤ

```

なおJavaでは、片方のデータが数値（整数や小数）でも連結が可能です。その場合、**数値を文字列型に変換したうえで連結**します。たとえば、

```java
System.out.println("侍テラコヤ" + 123 );

```

を実行すると、以下のように「侍テラコヤ」と「123」が連結されます。

```
侍テラコヤ123

```

このように文字列同士や、文字列と数値は「+」で連結できると覚えておきましょう。

補足：計算や連結は左から順番に処理される

複数の計算や連結を一度に行う場合、**左から順番に処理される**のが基本です。この性質を考慮しないと、計算結果が自分の想定どおりにならない場合があるため注意しましょう。

たとえば以下のように、「侍テラコヤ」「123」「654」の3データを「+」で計算・連結する場合を2パターン考えます。両者の違いは、「侍テラコヤ」の位置です。

Data_9.java

```java
package text.section_04;

public class Data_9 {
    public static void main(String[] args) {
        System.out.println("侍テラコヤ" + 123 + 654); // パターン1
        System.out.println(123 + 654 + "侍テラコヤ"); // パターン2
    }
}

```

上記を実行すると、パターン1とパターン2では数値部分の表示内容が変わります。

```
侍テラコヤ123654
777侍テラコヤ

```

パターン1だと、最初に「侍テラコヤ」という文字列があるため、1つ目の「+」が文字列連結として処理されます。「侍テラコヤ123」に2つ目の「+」で「654」を連結するのです。

パターン2だと、最初の2データが数値同士のため、「123 + 654」の計算により「777」となります。そこに「侍テラコヤ」を連結するため、結果は「777侍テラコヤ」です。

なお、**後ろの計算を先に行いたいときは( )ではさむ**手があります。

```
System.out.println("侍テラコヤ" + (123 + 654) ); // (123 + 654)が先に計算される

```

上記のようにすると、( )ではさんだ「123 + 654」が先に計算されて「777」となります。

```
侍テラコヤ777

```

このように、3つ以上のデータをまとめて計算するときは、順序も意識しなければなりません。特に数値や文字列が混在していると、順序により結果が変わりやすいため注意しましょう。

本章の学習は以上です。お疲れさまでした。

まとめ

本章では以下の内容を学習しました。

- 用語
  - リテラル：ソースコードに書かれた具体的な値
  - サフィックス：末尾に付けるキーワード
- データの種類（データ型）
  - （1）整数型
    - 「100」「0」「-123」など、整数を表すデータ型

データ型	値の範囲	サイズ
byte	-128～127	8ビット
short	-32,768～32,767	16ビット
int	-2,147,483,648～2,147,483,647	32ビット
long	-9,223,372,036,854,775,808～9,223,372,036,854,775,807	64ビット

- （2）浮動小数点型
  - 「0.1」「1.618」など、1よりも小さい数を含んだ数値のデータ型
  - 浮動小数点型は誤差が生じる場合がある

データ型	値の範囲	サイズ
float	(-3.4028235 × 10の38乗)　～(3.4028235 × 10の38乗)	32ビット
double	(-1.7976931348623157 × 10の308乗)　～(1.7976931348623157 × 10の308乗)	64ビット

- （3）論理型
  - true（真）またはfalse（偽）を表すデータ型

データ型	値の範囲	サイズ
boolean	true または false	1バイト※環境による

- （4）文字列型
  - 「こんにちは」「Hello, world!」など、文字列（文字の集まり）を表すデータ型
  - 文字列型は基本的にString型を使う
  - 1文字だけを扱う場合はchar型を使う

- 数値の計算方法
  - 計算したいデータの間に算術演算子（計算用の記号）を書く
  - 計算するデータに小数が含まれるとき、計算結果は浮動小数点型になる
  - 代表的な算術演算子は以下の5つ

算術演算子	処理の内容	例（出力結果）
+	左の値と右の値を足す（加算）	45 + 18（63）
-	左の値から右の値を引く（減算）	30 - 12（18）
*	左の値と右の値を掛ける（乗算）	15* 6（90）
/	左の値を右の値で割る（除算）	30 / 3（10）
%	左の値を右の値で割った余りを求める（剰余演算）	45 % 8（5）

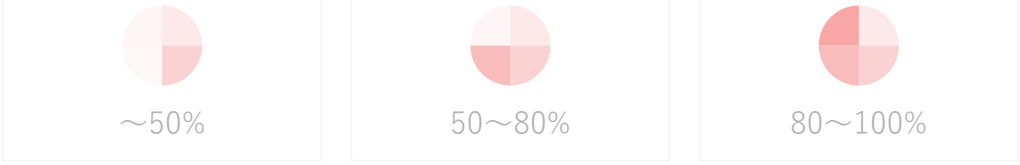
- 文字列の連結方法
  - 連結したいデータ2つを「+」でつなぐ
  - 片方のデータが数値の場合、数値を文字列型に変換したうえで連結する
  - 計算や連結は左から順番に処理されるため、順番によっては結果が変わる

データの扱いは、Javaに限らずプログラミングにおいて基本中の基本です。今後の学習でつまずかないためにも、しっかり復習してデータの扱いに慣れましょう。

次章では、変数について学びます。

理解度を選択して次に進みましょう

ボタンを押していただくと次の章に進むことができます



前に戻る

4 / 31 ページ

次に進む

< 一覧に戻る

改善点のご指摘、誤字脱字、その他ご要望はこちらからご連絡ください。