12.クラスのインスタンス生成とメソッドの理解

12.1 サンプルを見って理解しましょう。

例）

**class** Calculation{　　　//クラス名定義

**int** left, right;　　　//クラスの変数

**public** **void** setValue(**int** left, **int** right){　　　//メソッド

**this**.left = left;

**this**.right = right;

}

**public** **void** sum(){　　　//メソッド

System.out.println(**this**.left+**this**.right);

}

**public** **void** avg(){　　　//メソッド

System.out.println((**this**.left+**this**.right)/2);

}

}

**public** **class** CalculationDemo4 {

**public** **static** **void** main(String[] args) {

Calculation cal1 = **new** Calculation();　　//インスタンス生成

cal1.setValue(10, 20);

cal1.sum();

cal1.avg();

Calculation cal2 = **new** Calculation();　　//インスタンス生成

cal2.setValue(20, 40);

cal2.sum();

cal2.avg();

}

}

**12.2クラス**

クラスは、変数とメソッドの集合（設計図：複数の所から使える共通関数）です。

上の例では

**class** Calculation{　　　//クラス名定義

**int** left, right;　　　//クラスの変数

・

**public** **void** setValue(**int** left, **int** right)　　//メソッド

.

**public** **void** sum(){　　　//メソッド

・

**public** **void** avg(){　　　//メソッド

｝

として定義しています。

**12.3インスタンス**

それでは、クラスの使い方を学びましょう。 クラスは一種の設計図だ。 クラスを定義すること自体ではできることはあまりありません。 設計図を具体的な製品にしなければならない。 それから使用するキーワードがnewである。

Calculation cal1 = new Calculation();

new Calculation() はクラス Calculation を具体的な製品にするコマンドです。 このように作られた具体的な製品をインスタンス（instance）と呼ぶ。 以下の関係を覚えておこう。

クラス：設計図

インスタンス:製品

上記のコードは、newを使って作成したインスタンスを変数cal1に格納しています。 変数cal1にインスタンスを入れた理由は、cal1を介してインスタンスを制御する必要があるからです。

cal1の前にCaculatorが位置している。 これまで、私たちはさまざまな変数を宣言してきましたようにごれも変数宣言になります。

では、以下の構文は変数名cal1のデータ型が何という意味なのか？

Calculation cal1

Calculationという意味だ。 つまり、クラスを作成することは、カスタムデータ型を作成するのと同じ意味です。 だが今の段階ではカスタムタイプという概念が容易に届かないので、これについての説明は次第とする。 今覚えているのは、クラスをインスタンス化するときは変数に入れなければならないということと、このとき使用する変数のデータ型はそのクラスになるという点だ。

クラスとインスタンスは設計図と製品だと説明した。 それでは、オブジェクトはクラスなのかインスタンスなのか？ 多少議論があるが、一般的に設計図であるクラスが具体的な実体であるインスタンスになったときにオブジェクトと呼ばれる。 通常、具体的なコード上に現れるオブジェクトをインスタンスと呼び、ロジックを設計するときに現れるインスタンスをオブジェクトと呼ぶ。 あるいは、クラス、インスタンスの区別なく包括的にオブジェクトという言葉を書くこともある。 このような混乱を避けるために、私はできるだけクラスとインスタンスを使うようにします。

クラスを単に変数とメソッドの束で見れば不足している。 オブジェクトを作成して使用する理由は、リサイクル性を高めるためです。

Calculation cal1 = **new** Calculation();

cal1.setValue(10, 20);

cal1.sum();

cal1.avg();

Calculation cal2 = **new** Calculation();

cal2.setValue(20, 40);

cal2.sum();

cal2.avg();

上記のコードを見ると、2つのインスタンスをそれぞれcal1とcal2に入れました。 そしてそれぞれのインスタンスに所属するメソッドを呼び出している。 各インスタンスはメソッド setValue を介して変数 left, right の値を設定している。 その結果、インスタンスCAL1とCAL2は以下のように異なる変数の値を内部的に持っています。

**public** **void** setValue(**int** left, **int** right){

**this**.left = left;

**this**.right = right;

}

このときメソッドsetValue内にthis.というものがある。 thisはクラスを通して作られたインスタンス自身を指す。 上記のコードでは、leftはパラメータであり、この変数はsetValue以外ではアクセスできません。 一方、this.leftは4行目で宣言した変数に値を設定するもので、メソッド外で宣言した変数はインスタンス内のすべてのメソッドからアクセス可能です。 これについての話は後ろから次第にする。 理解していなくても心配しないでください。

例）

// クラスの定義

public class MyClass {

private int myNumber;

// コンストラクター

public MyClass(int number) {

myNumber = number;

}

// メソッド

public void printNumber() {

System.out.println("My number is " + myNumber);

}

// mainメソッド

public static void main(String[] args) {

MyClass instance = new MyClass(123);

instance.printNumber();

}

}

この例では、MyClassというクラスを定義しています。このクラスには、myNumberというプライベートフィールドと、printNumber()というパブリックメソッドがあります。コンストラクターは、myNumberフィールドを引数で初期化します。

また、MyClassクラスには、main()メソッドも定義されています。main()メソッドは、Javaアプリケーションのエントリーポイントとして機能し、プログラムの実行を開始します。

main()メソッドでは、MyClassのインスタンスを生成し、printNumber()メソッドを呼び出しています。newキーワードを使って、MyClassのインスタンスを生成し、instance変数に代入しています。その後、instance変数を使って、printNumber()メソッドを呼び出しています。printNumber()メソッドは、インスタンスのmyNumberフィールドの値をコンソールに出力します。

繰り返して書きますがmain()メソッドがpublic、static、voidであり、String[] argsを引数に取ることに注意してください。これは、Javaアプリケーションのエントリーポイントとして必要な規定事項です。