27

Java の覚え書き

クラスの構成

フィールド、コンストラクタ、メソッド

Java 言語はオブジェクト指向と呼ばれる考え方を導入した言語です。

オブジェクト指向では、オブジェクト＝具体的な役割を持ったもの（小さなプログラム）をいくつも組み合わせて、大きなプログラムを作成して行きます。

Java 言語ではこのオブジェクトを「クラス」という形で宣言し、クラスはフィールド、コンストラクタ、メソッドの３つのブロックで構成されています。

```
class クラス名 {
    フィールド        オブジェクトの記憶領域（変数）
    コンストラクタ    オブジェクトの初期化（初期化専用の特別なメソッド）
    メソッド          オブジェクトの動作・機能（メソッド）
}
```

28

Java の覚え書き

クラスからインスタンスを作る

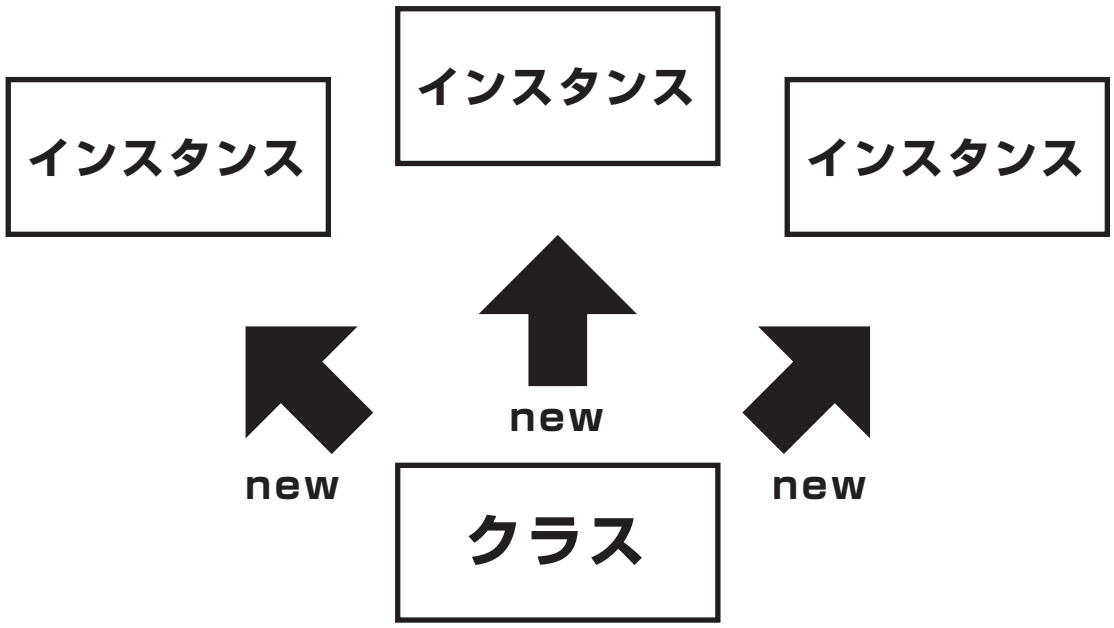
```
クラス名 変数 = new クラス名 ();
```

Java 言語で宣言したクラスを利用するには、多くの場合「インスタンス」という物を作成する必要があります。
（直接利用する特別なクラスもあります。）

インスタンスは「new 演算子」を使って作成し、変数に代入します。

ここがポイント

インスタンスの寿命は、変数のスコープ（有効範囲）に依存します。
また、インスタンスは一つのクラスからいくつでも作成する事ができます。



29

Java の覚え書き

インスタンス生成時の流れ

new →（コンストラクタ）→ インスタンス生成完了

コンストラクタはインスタンスを生成する際に
必ず実行される初期化処理を記載した部分です。
コンストラクタの宣言は

コンストラクタ名（引数型 引数名）

となり、
コンストラクタ名は属するクラスと同じ名前に
なります。

ここが
ポイント

またコンストラクタは異なる
引数にする事で複数宣言する事もできます。

（実行手順）

コンストラクタ実行後、
インスタンスが生成されます

```
1：  ClassName obj;
2：  obj = new ClassName();
3：  System.out.println(obj.fieldName);
```

1：　オブジェクト変数を宣言（インスタンスはまだ出来ていない）
2：　インスタンスを生成（コンストラクタ実行→生成完了）
3：　インスタンスを利用する

30

Java の覚え書き

インスタンスの使い方

インスタンスを通じてクラスで定義したフィールドやメソッドを利用するには、
インスタンスを格納した変数に .（ドット）を付けてアクセスします。

変数名 . フィールド名

（使用例）

```
1：  ClassName obj = new ClassName();
2：  obj.fieldName = 値 ;
3：  System.out.println(obj.fieldName);
```

1：　インスタンス生成
2：　インスタンスフィールドに代入
3：　インスタンスフィールドを参照

変数名 . メソッド名 ()

（使用例）

```
1：  ClassName obj = new ClassName();
2：  obj.methodName();
3：  String str = obj.returnMethodName();
```

1：　インスタンス生成
2：　インスタンスメソッドの呼び出し
3：　戻り値のあるインスタンスメソッドの呼び出し

31

Javaの覚え書き

クラスを利用した時の実行順序

Execute.java

```
public class Execute {
        public static void main(String[] args) {
                Rectangle r = new Rectangle(); ①
                System.out.println(r.width); ⑨
                System.out.println(r.height); ⑩
                System.out.println(r.getArea()); ⑪
        }
}
```

プログラムの実行結果

```
>java Execute
10
20
200
```

Execute.java プログラムを実行した時、
プログラムコードは 1～14 の番号順で再生されます。

Rectangle.java

```
class Rectangle {
        int width;
        int height;                                  (フィールド)

        Rectangle() { ②
                setSize(10, 20); ③
        } ⑧                                          (コンストラクタ)

        void setSize(int w, int h) { ④
                width = w; ⑤
                height = h; ⑥
        } ⑦                                          (メソッド)

        int getArea() { ⑫
                return width*height; ⑬
        } ⑭                                          (メソッド)
}
```

32

Java の覚え書き

static 修飾子について

static を付けるとクラス、付けなければインスタンス

フィールド、メソッドを宣言する際、
static 修飾子が付与されたものをクラスフィールド、
クラスメソッド、static 修飾子が付与されていないもの
をインスタンスフィールド、インスタンスメソッドと
言います。

クラスフィールド、クラスメソッドはクラス自身に保持
され、クラスから直接使用します。インスタンスフィー
ルド、インスタンスメソッドは生成したインスタンス毎
に保持され、そのインスタンスから使用します。

また、クラスからインスタンスフィールド・メソッドへ
の参照は出来ませんが、インスタンスからクラスフィー
ルド・メソッドへの参照は可能です。

static あり（クラスフィールド・メソッドの使用方法）

クラス名 . フィールド名

クラス名 . メソッド名 ()

ポイント

クラスフィールドを使う事で、
複数のインスタンス間でフィールドの値を
共有することができます。

33

Javaの覚え書き

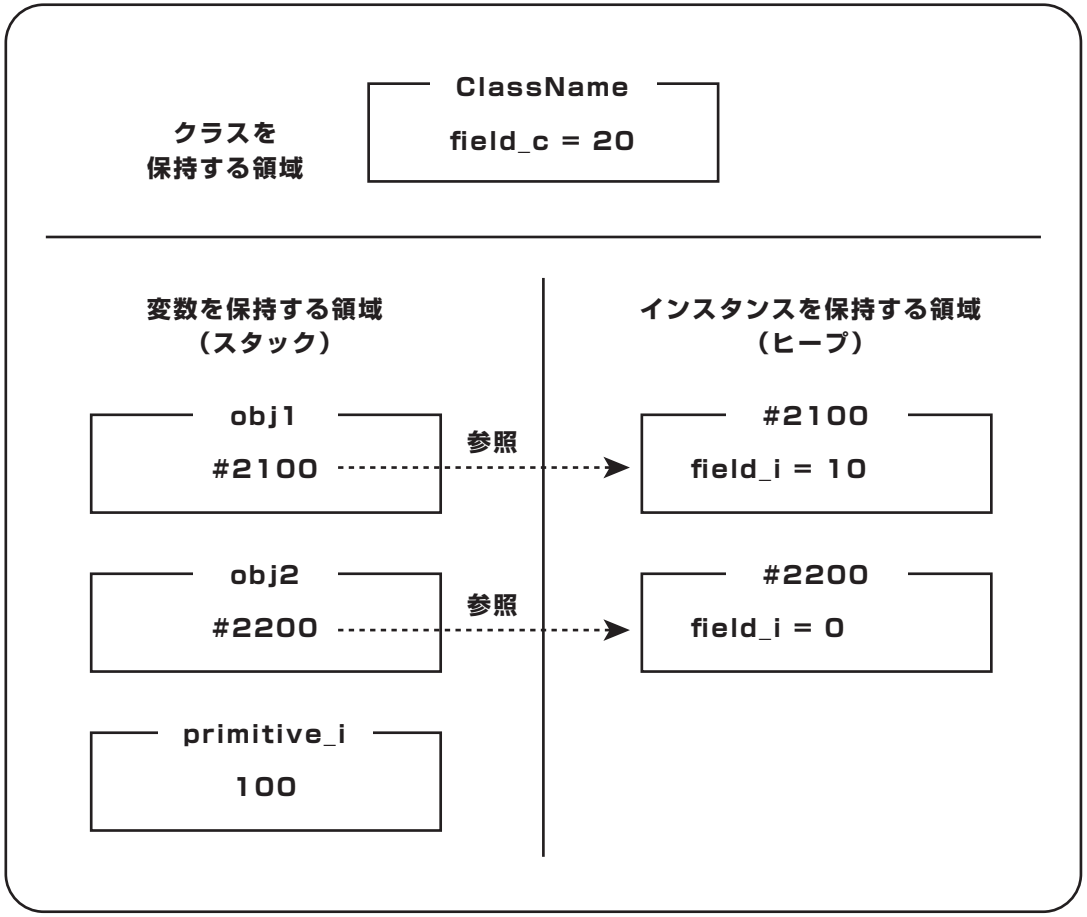
クラス（static）とインスタンスのデータ保持イメージ

```
class ClassName {
        int field_i;
        static int field_c;
}
```

```
class Execute {
        public static void main(String[] args) {
                ClassName obj1 = new ClassName();
                ClassName obj2 = new ClassName();
                int primitive_i = 100;
                obj1.field_i = 10;
                ClassName.field_c = 20;
                System.out.println(obj1.field_i);
                System.out.println(obj1.field_c);
                System.out.println(obj2.field_i);
                System.out.println(obj2.field_c);
        }
}
```

プログラムの実行結果

```
>java Execute
10
20
0
20
```



メモリ内イメージ
（※簡易説明用の為、実際のメモリモデルとは異なります。）