34

Java の覚え書き

クラスの拡張（継承）

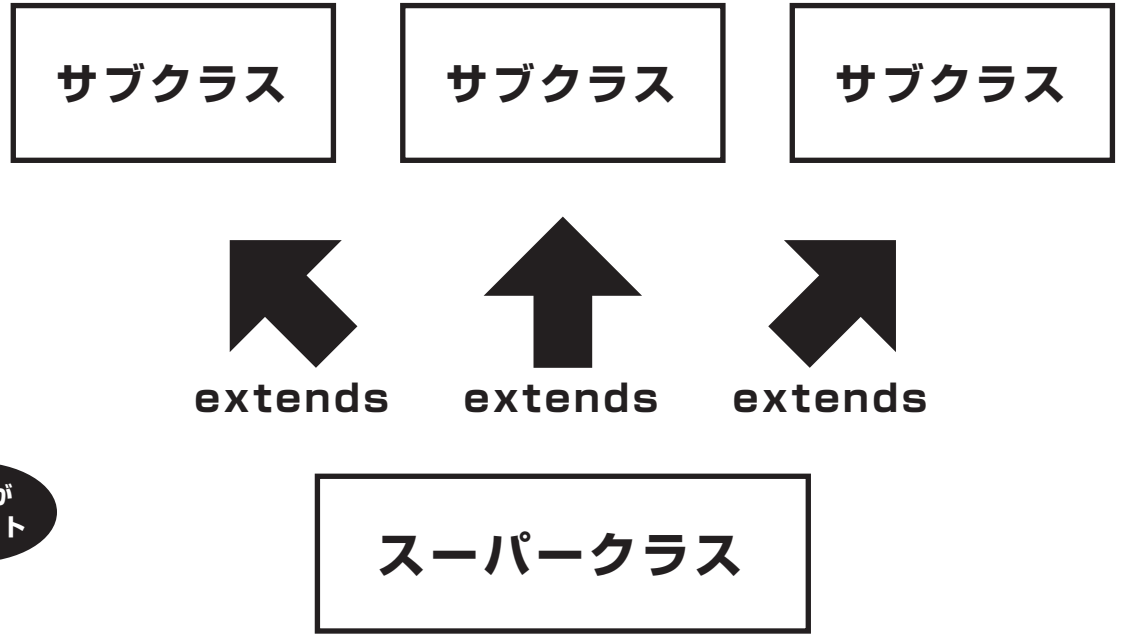
クラス名 extends スーパークラス名

新しいクラスを宣言する時、すでに存在するクラスをベースにして、新たにフィールドやメソッドを追加したり、メソッドの内容を変更して宣言する事ができます。

ベースとなるクラスを「スーパークラス」
新たに作成するクラスを「サブクラス」といい、
スーパークラスの内容（フィールド、メソッド）
をサブクラスに引き継ぐ事を「継承」といいます。
（コンストラクタは継承されません）

ここがポイント

継承できるスーパークラスは１つのみですが、
サブクラスは幾つでも宣言する事ができます。



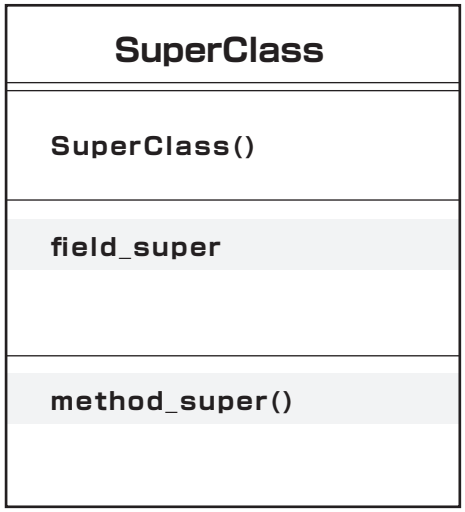
35

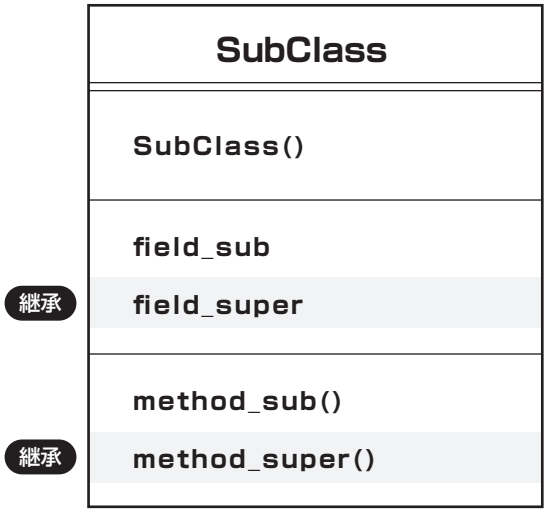
Java の覚え書き

継承のイメージ

```
class SuperClass {
        int field_super;

        SuperClass() {
                ...
        }

        void method_super() {
                ...
        }
}
```

```
class SubClass extends SuperClass {
        int field_sub;

        SubClass() {
                ...
        }

        void method_sub() {
                ...
        }
}
```





SubClass のコンストラクタは１つ
・SubClass()
（コンストラクタは継承されない）

SubClass で利用できるフィールドは２つ
・field_sub
・field_super
（フィールドは継承される）

SubClass で利用できるメソッドは２つ
・method_sub()
・method_super()
（メソッドは継承される）

サブクラスのインスタンス生成時の流れ

```
public class Execute {
        public static void main(String[] args) {
                SubClass obj = new SubClass();
        }
}
```

```
class SubClass extends SuperClass {
        int field_sub;

        SubClass() {
                super();
                System.out.println("Sub");
        }

        void method_sub() {
                …
        }
}
```

super() の記述は省略できます。
引数ありのコンストラクタの場合は
明示的に記述する必要があります。

```
class SuperClass {
        int field_super;

        SuperClass() {
                System.out.println("Super");
        }

        void method_super() {
                …
        }
}
```

① new 演算子で SubClass のインスタンスを生成

② SubClass のコンストラクタより super() が呼ばれる
（SuperClass のコンストラクタ処理を実行）

③ SubClass のコンストラクタ処理を実行

プログラムの実行結果

```
>java Execute
Super
Sub
```

スーパークラスのコンストラクタは継承されませんが、
スーパークラスのコンストラクタが実行されている事を
意識しましょう。

37

Java の覚え書き

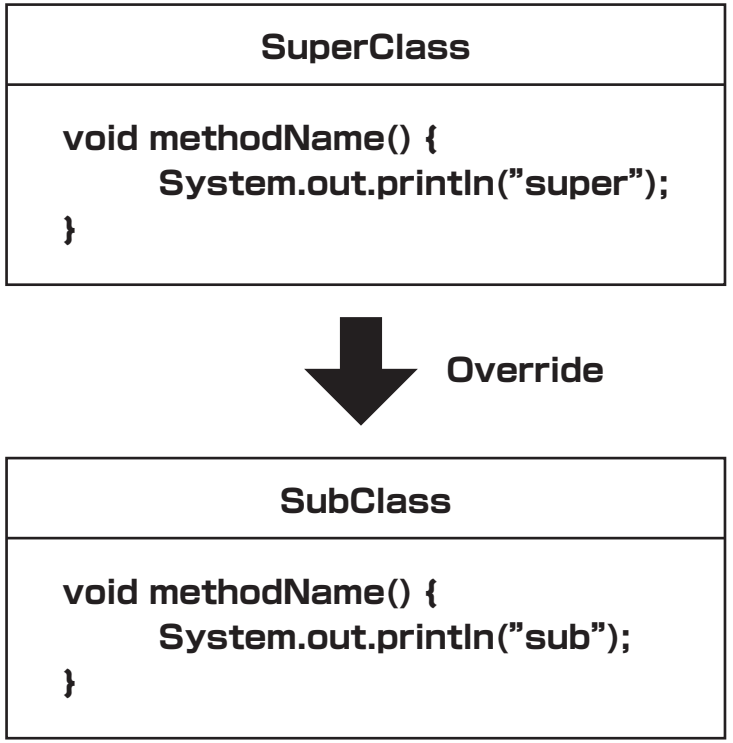
オーバーライド

スーパークラスのメソッドの書き換え（再定義）

オーバーライドとはスーパークラスにおいて定義されているインスタンスメソッドを、サブクラス内で再定義することを言います。
同名のメソッド名で、メソッドの振る舞いをサブクラスに特化させた内容に変更する時に使用します。

重要

オーバーライドする側はオーバーライドされる側と戻り型、インスタンスメソッド名、引数型、引数の数が同じでなければなりません。どれか一つでも異なる場合はオーバーライドとは見なされません。



38

Java の覚え書き

アクセス修飾子（private）について

そのクラス内だけで使用したいものに private をつける

Java 言語ではアクセス修飾子と呼ばれる、メソッドやフィールド、クラスなどにアクセス権を指定するしくみが用意されています。アクセス権を指定することで、オブジェクトの使用者に対して想定外の操作等によるデータ改ざんを防ぐことができ、より安全なソースコードを作成することができます。

```
class ClassName {
    private fieldName;
    private void methodName() {
        …
    }
}
```

private が付いているメソッドやフィールドは他のクラスから参照できなくなります。

private が付いているメソッドやフィールドは、サブクラスには継承されなくなります。

豆知識

原則として、フィールドは private にして、メソッドを通してフィールドの内容にアクセスできるようにしておきます。フィールドから値を取り出して返すメソッドを「Getter」、フィールドに値を入れるメソッドを「Setter」といい、メソッド名も「get フィールド名」や「set フィールド名」の様にします。

39

Java の覚え書き

抽象クラス（abstract）

具体的な実装はサブクラスにゆだねて、枠のみを規定する

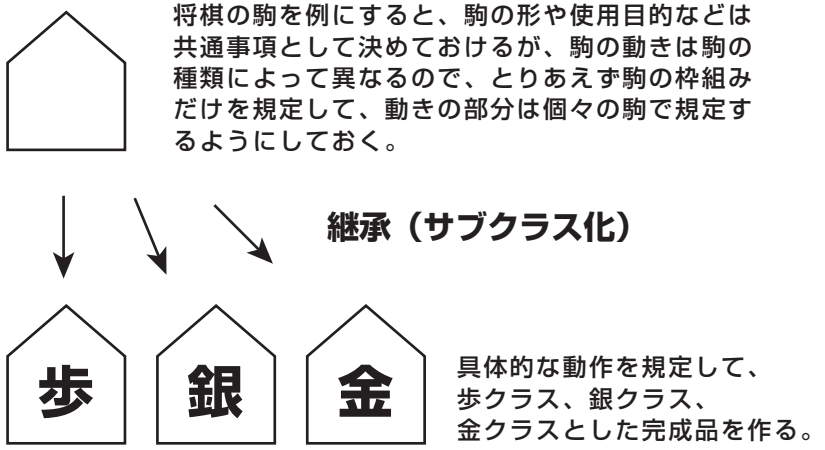
抽象クラスはオブジェクトとしての枠組みを規定して、それを元にサブクラスを量産する時などに使用します。
乱暴な言い方をすれば、その性質上それがまだ未完成な状態である事を宣言しています。

抽象メソッドがあるクラスは必ず抽象クラスになります。

抽象クラスのインスタンス化はできません。

インスタンス化できるクラスにするには抽象メソッドをオーバーライドして、具体的な処理を記述します。（実装）

```
abstract class Koma {
        String name;
        abstract void move();
}
```



40

Java の覚え書き

ポリモルフィズム（多態性）

オブジェクト毎に違う振る舞いをさせる

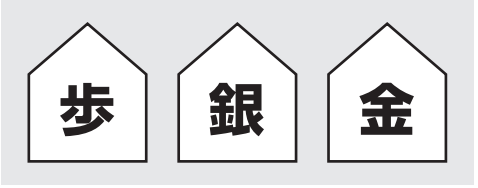
クラスは必ず何かのオブジェクトに属する事（継承関係）になります。
すなわち、そのオブジェクトのクラス階層において、クラスは何らかのサブクラスであるといえます。
ポリモルフィズムとは、サブクラスが属しているクラス（同じ性質を有している）の型を利用してオブジェクト毎の振る舞いを変更する事ができる機能です。

前項の例で言うと、歩クラス、銀クラス、金クラスは駒クラスに属しているので駒クラスの型で統一する事ができます。
駒クラスで規定されたフィールド名やメソッド名を使って、歩クラスや銀クラスなど個々に定義したフィールドやメソッドにアクセスする事で、オブジェクト毎の動作を変更する事ができます。

```
Koma[] koma = new Koma[3];
koma[0] = new Hu();
koma[1] = new Gin();
koma[2] = new Kin();

for (int i=0; i<koma.length; i++) {
      System.out.println(koma[i].toString());
}
```

koma 型の配列で複数の型を一元管理しています。

共通の型で処理を行う事で、複数の型に対する処理をコンパクトにまとめる事ができます。



共通の型：Koma 型、Object 型

歩、銀、金において、共通の型はスーパークラスである「Koma」とその祖先である「Object」になります。
よって、Koma 型だけではなく、Object 型でも扱う事ができます。