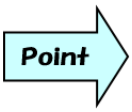
Java によるオブジェクト指向プログラミング（Java SE 11 対応）

基本編

TAC

Contents

【テキストで使われる記号】



理解を深めるポイントや補足を強調しています。読み飛ばさないように注意しましょう。

Windows OS 向けの説明です

macOS 向けの説明です

Windows/macOS 両方向けの説明です

【システム環境】

以下の環境で動作確認をしています。

- Windows 10（64 ビット版）
- Windows 11
- macOS Big Sur 11

【インストールするアプリケーション】

以下のバージョンのアプリケーションをインストールします。

- OpenJDK Eclipse Temurin 11
- Visual Studio Code 1.74.3

第2章
クラスの継承

この章では、クラスの継承について学習します。
クラスの継承を行うことで、より効率のよいプログラムを作成することができます。

2.1　継承とは？

入門編では一からクラスを定義してきましたが、既に定義済みのクラスをもとに、不足しているフィールドやメソッドを追加して新しくクラスを定義することを**継承**、あるいは**拡張する**といいます。従来のプログラミングでは、もととなるクラスの記述すべてを、新しく作成するクラスに同じように記述する必要がありました。しかし継承を使うと、もととなるクラスに記述されている内容を新しく作成するクラスに改めて記述する必要はありません。追加したいフィールドやメソッドを記述するだけで済むのです。

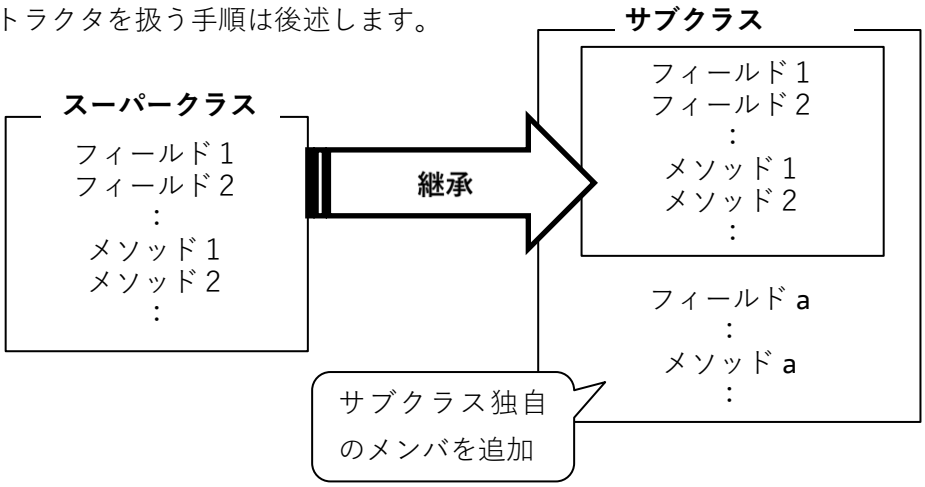
継承を表すキーワードは「extends（エクステンズ）」です。そして、もととなるクラスのことを**スーパークラス**、スーパークラスから作る新規のクラスを**サブクラス**といいます。継承を使ったクラスの定義の書式は次のとおりです。

書式	`class サブクラス名 extends スーパークラス名{`
	`　　型名　フィールド名;`
	`　　戻り値の型　メソッド名(引数リスト){`
	`　　　　処理内容`
	`　　　　return　式;`
	`　　}`
	`}`
	追加するメンバ

説明	「スーパークラスを継承してサブクラスを定義する」という意味です。

スーパークラスからサブクラスに継承されるものはスーパークラスで定義されているメンバです。

ただし、**コンストラクタは継承されません**。サブクラスからスーパークラスのコンストラクタを扱う手順は後述します。



◆◇サンプルプログラム◇◆

【Step1：Employee クラスを継承してサブクラスを作る】

入門編で作成した Employee クラス（「サンプルプログラム¥第 2 章¥Step1」フォルダから Employee.java ファイルを作業フォルダにコピーしてください）をもとにして、サブクラス Programmer と Administrator を作成します。

【参考：Employee.java】

```
 1: class Employee{
 2:     //フィールド
 3:     private static int number = 0;    //総社員数
 4:     private int      code;            //社員番号
 5:     private String   name;            //社員名
 6:     private int      year;            //入社年数
 7:     private int      age;             //年齢
 8:
 9:     //コンストラクタの定義
10:     public Employee(String name, int year, int age){
11:         setName(name);
12:         setYear(year);
13:         setAge(age);
14:         code = ++number; //number を 1 カウントアップする
15:     }
16:     //引数なしのコンストラクタ
17:     public Employee(){
18:         this("未定", 0, 0);
19:     }
20:     //引数 1 つのコンストラクタ
21:     public Employee(String name){
22:         this(name, 0, 0);
23:     }
24:     //引数 2 つのコンストラクタ
25:     public Employee(String name, int age){
26:         this(name, 0, age);
27:     }
28:
29:     //フィールドのデータを表示するメソッド
30:     public void display(){
31:         System.out.println("社員番号:" + code);
32:         System.out.println("社員名:" + name);
33:         System.out.println("入社年数:" + year);
34:         System.out.println("年齢:" + age);
35:     }
36:     //社員名を設定するメソッド
37:     public void setName(String name){
38:         this.name = name;
39:     }
40:     //入社年数を設定するメソッド
41:     public void setYear(int year){
```

```
42:             if(year < 0){
43:                 this.year = 0;
44:             }else{
45:                 this.year = year;
46:             }
47:         }
48:         //年齢を設定するメソッド
49:         public void setAge(int age){
50:             if(age < 0){
51:                 this.age = 0;
52:             }else{
53:                 this.age = age;
54:             }
55:         }
56:         //社員名を取得するメソッド
57:         public String getName(){
58:             return name;
59:         }
60:         //入社年数を取得するメソッド
61:         public int getYear(){
62:             return year;
63:         }
64:         //年齢を取得するメソッド
65:         public int getAge(){
66:             return age;
67:         }
68:         //総社員数を表示するメソッド
69:         public static void showTotalNumber(){
70:             System.out.println("総社員数は" + number + "人です。");
71:         }
72: }
```

【Programmer クラスの仕様】

Employee クラスから拡張するメンバを下記に記述します。

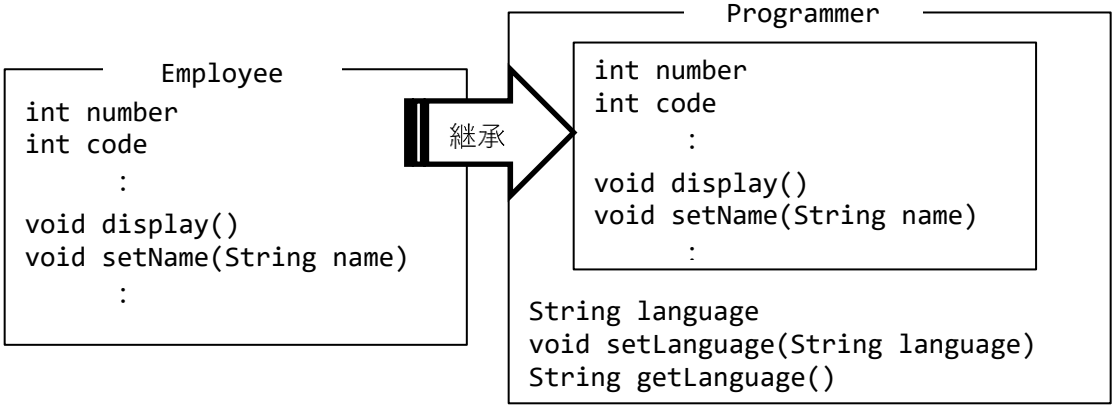
フィールド	説明
language	仕事で使用しているプログラム言語を格納するフィールド
メソッド	**説明**
setLanguage	フィールド language にデータを格納するアクセサメソッド
getLanguage	フィールド language のデータを返すアクセサメソッド

【ファイル名：Programmer.java】

```
 1: class Programmer extends Employee{
 2:     //インスタンス変数
 3:     private String language;      //プログラム言語
 4:     //メソッド
 5:     public void setLanguage(String language){
 6:         this.language = language;
 7:     }
 8:     public String getLanguage(){
 9:         return language;
10:     }
11: }
```

【解　説】

1 行目で「class Programmer」に続けて「extends Employee」を記述します。この記述をしないと Employee クラスのメンバが継承されないので、注意してください。



【Administrator クラスの仕様】

Employee クラスから拡張するメンバを下記に記述します。

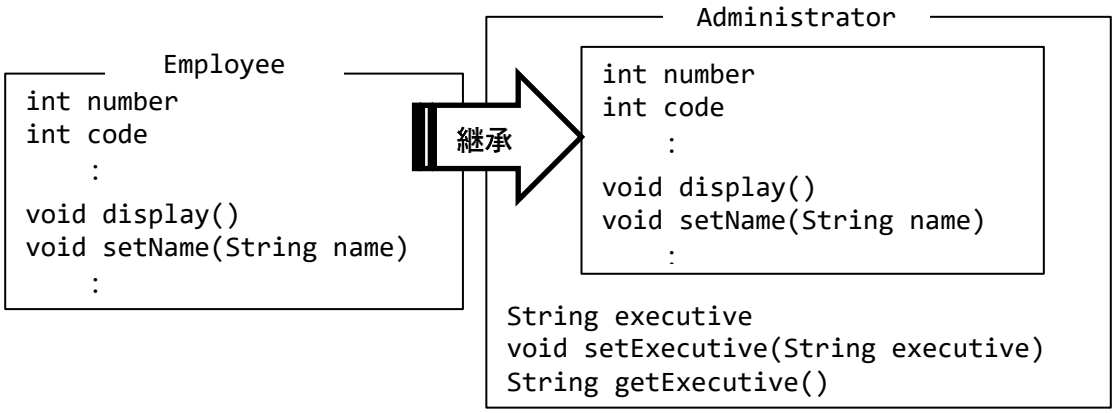
フィールド	説明
executive	仕事で使用しているプログラム言語を格納するフィールド
メソッド	**説明**
setExecutive	フィールド executive にデータを格納するアクセサメソッド
getExecutive	フィールド executive のデータを返すアクセサメソッド

【ファイル名：Administrator.java】

```
1:  class Administrator extends Employee{
2:      //インスタンス変数
3:      private String executive;        //役職
4:      //メソッド
5:      public void setExecutive(String executive){
6:          this.executive = executive;
7:      }
8:      public String getExecutive(){
9:          return executive;
10:     }
11: }
```

【解　説】

ここでも Programmer クラスを作成するときと同じく、1 行目で「class Administrator」に続けて「extends Employee」を記述します。



◆◇サンプルプログラム◇◆

【Step2：サブクラスでインスタンスを生成】

Step1 で作成した Programmer クラスと Administrator クラスのインスタンスを生成し、各インスタンスからメソッドを呼び出すようなプログラムを作成します。

【ファイル名：ExtendsTest.java】

```
 1: class ExtendsTest{
 2:     public static void main(String[] args){
 3:         //Programmer クラスでインスタンスを生成する
 4:         Programmer takahashi = new Programmer();
 5:         //Administrator クラスでインスタンスを生成する
 6:         Administrator  saito = new Administrator();
 7:
 8:         //継承したメソッド(社員名の設定)を呼び出す
 9:         takahashi.setName("高橋　耕太");
10:         //Programmer クラスで拡張したメソッドを呼び出す
11:         takahashi.setLanguage("Java 言語");
12:         //継承したメソッド(情報表示)を呼び出す
13:         takahashi.display();
14:         //継承したメソッド(社員名の設定)を呼び出す
15:         saito.setName("斎藤　博文");
16:         //Administrator クラスで拡張したメソッドを呼び出す
17:         saito.setExecutive("チーフ");
18:         //継承したメソッド(情報表示)を呼び出す
19:         saito.display();
20:     }
21: }
```

実行結果

```
C:\Java>java ExtendsTest
社員番号:1
社員名:高橋　耕太
入社年数:0
年齢:0
社員番号:2
社員名:斎藤　博文
入社年数:0
年齢:0
```

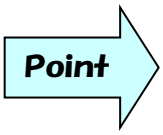
【解　説】

サブクラスにはコンストラクタを定義していないため、デフォルトコンストラクタ（下記 Point 参照）を呼び出して、インスタンスを生成します。その後で、社員名とサブクラスで拡張したフィールドのデータを設定し、最後に各インスタンスが格納している情報を表示します。

setName・display メソッドはスーパークラス（Employee クラス）のメソッドで、それを継承したサブクラス（Programmer・Administrator クラス）のインスタンスから呼び出すことができます。

実際にサブクラスで setName・display メソッドを記述しなくても、「Employee クラスを継承（拡張）する」という意味の「extends　Employee」と記述するだけで、Employee クラスで定義したメンバをサブクラスのインスタンスから呼び出すことができるのです。このようにクラスを継承（拡張）することで、スーパークラスに足りない部分だけをコーディングすればよく、効率よく新しいクラスを作成することができます。

オブジェクト指向言語には「継承」という仕組みがあるからこそ、上記のような**差分コーディング**が実現できるのです。



デフォルトコンストラクタ

ひとつもコンストラクタを定義しなかった場合、暗黙のうちに定義されるコンストラクタを「**デフォルトコンストラクタ**」と呼びます。Step1 におけるサブクラス「Programmer」「Administrator」ともコンストラクタを定義していませんので、暗黙のうちにデフォルトコンストラクタが定義されています。例えば「Programmer」クラスにおいては、

```
class Programmer extends Employee{
    //インスタンス変数
    private String language;          //プログラム言語
    // 暗黙のうちに定義されるデフォルトコンストラクタ
    public Programmer(){
    }
    //メソッド
    public void setLanguage(String language){
  …
```

のように、**「引数なし」「処理内容なし」のコンストラクタ**が定義されているのです。