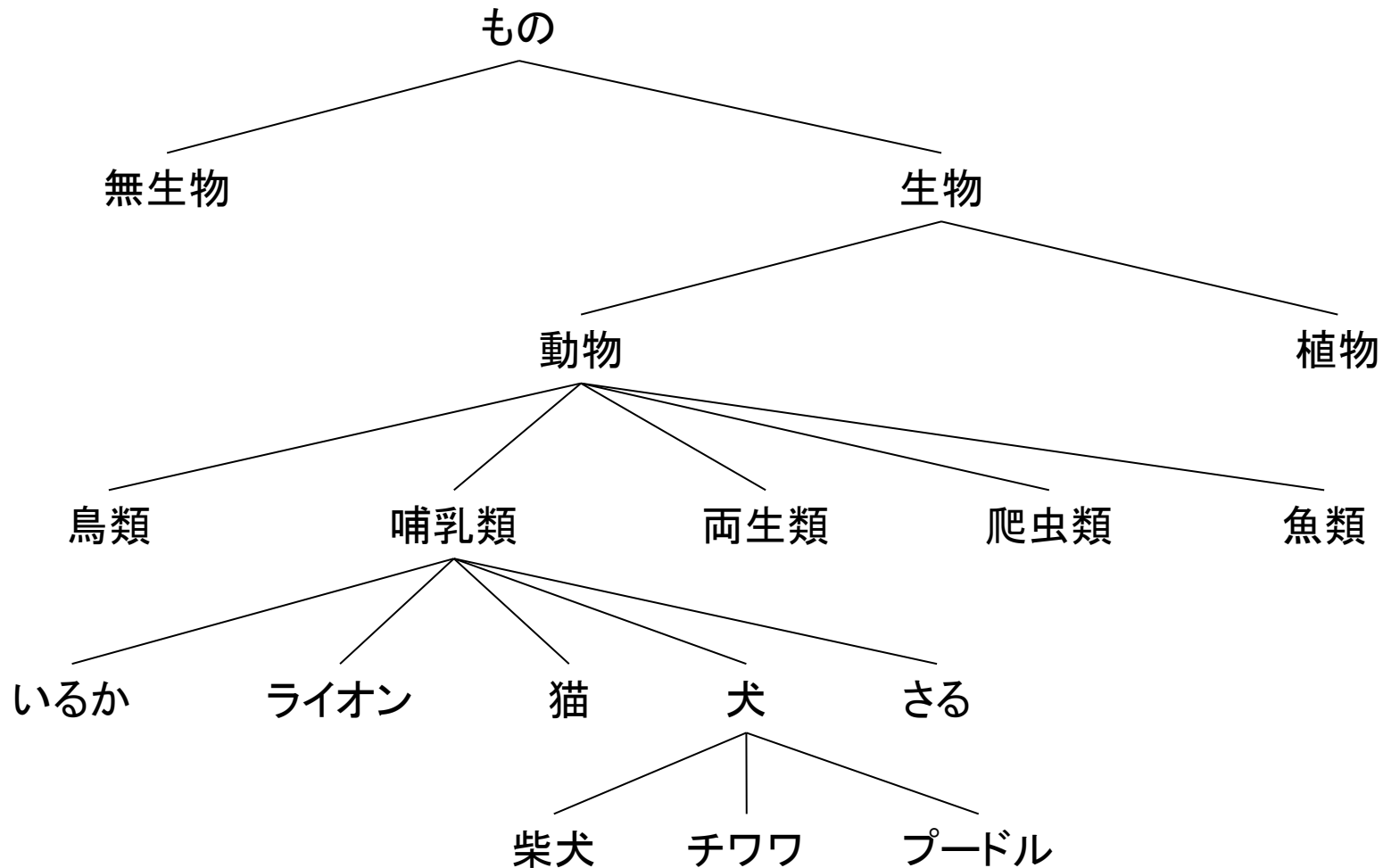


Javaプログラミング(5)

継承

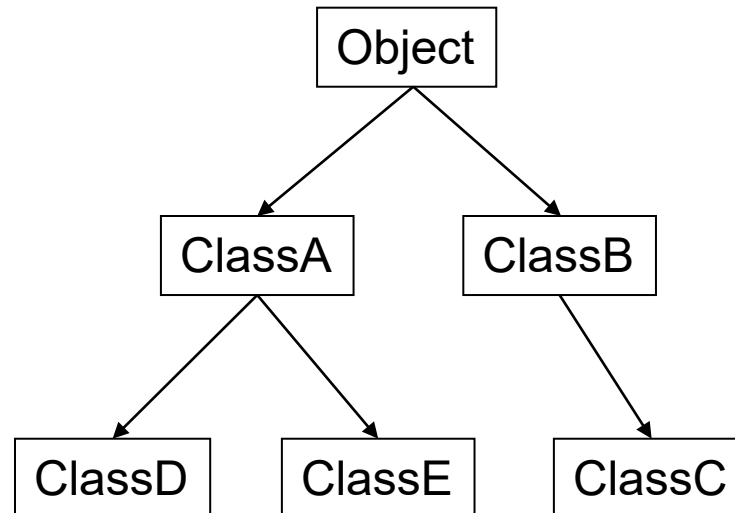
成蹊大学理工学部
情報科学科

世界のクラス階層



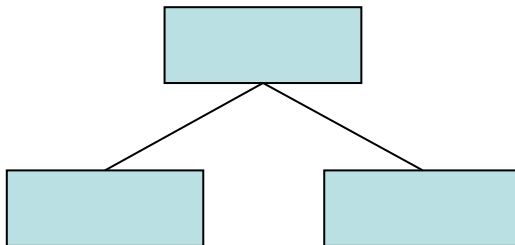
Javaのクラス階層

- Javaのクラスは, objectクラスをトップ(根)とする, 木構造になっている.

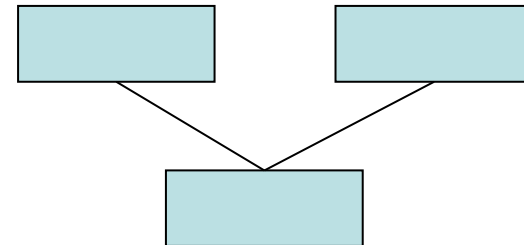


スーパークラスとサブクラス

- クラス中のメソッドやフィールドを追加することにより、クラスを「拡張」することができる。
- 拡張によりできたクラスを「サブクラス」と呼ぶ。
- 拡張された元のクラスを「スーパークラス」と呼ぶ。
- ただし、スーパークラスを2つ以上持つことはできない。つまり、サブクラスは1つのスーパークラスからしか拡張できない。
(C++ではクラスの多重継承が可能であった)
- Java言語全体は、クラスが階層的に定義されてできている。
- Object クラスは、Javaのクラス階層のルートである。すべてのクラスは、スーパークラスとして Object を持つ。



○ 複数のサブクラス



× 複数のスーパークラス

Javaのクラス階層の抜粋

- o java.lang.[Object](#)
 - o javax.swing.[AbstractAction](#) (implements javax.swing.[Action](#), java.lang.[Cloneable](#))
 - o javax.swing.text.[TextAction](#)
 - o javax.swing.text.[DefaultEditorKit.BeepAction](#)
 - o javax.swing.text.[DefaultEditorKit.CopyAction](#)
 - o javax.swing.text.[DefaultEditorKit.CutAction](#)
 - o javax.swing.text.[DefaultEditorKit.DefaultKeyTypedAction](#)
 - o javax.swing.text.[DefaultEditorKit.InsertBreakAction](#)
 - o javax.swing.text.[DefaultEditorKit.InsertContentAction](#)
 - o javax.swing.text.[DefaultEditorKit.InsertTabAction](#)
 - o javax.swing.text.[DefaultEditorKit.PasteAction](#)
 - o javax.swing.text.[StyledEditorKit.StyledTextAction](#)
 - o javax.swing.text.[StyledEditorKit.AlignmentAction](#)
 - o javax.swing.text.[StyledEditorKit.BoldAction](#)
 - o javax.swing.text.[StyledEditorKit.FontFamilyAction](#)
 - o javax.swing.text.[StyledEditorKit.FontSizeAction](#)
 - o javax.swing.text.[StyledEditorKit.ForegroundAction](#)
 - o javax.swing.text.[StyledEditorKit.ItalicAction](#)
 - o javax.swing.text.[StyledEditorKit.UnderlineAction](#)
 - o javax.swing.text.[AbstractDocument](#) (implements javax.swing.text.[Document](#), java.lang.[Cloneable](#))
 - o javax.swing.text.[DefaultStyledDocument](#) (implements javax.swing.text.[AbstractDocument](#))
 - o javax.swing.text.[PlainDocument](#)
 - o javax.swing.text.[AbstractDocument.AbstractElement](#) (implements javax.swing.text.[AbstractDocument](#), javax.swing.text.[MutableAttributeSet](#), java.io.[Serializable](#), javax.swing.tree.[TreeNode](#))
 - o javax.swing.text.[AbstractDocument.BranchElement](#)

スーパークラスの指定

スーパークラスの指定方法

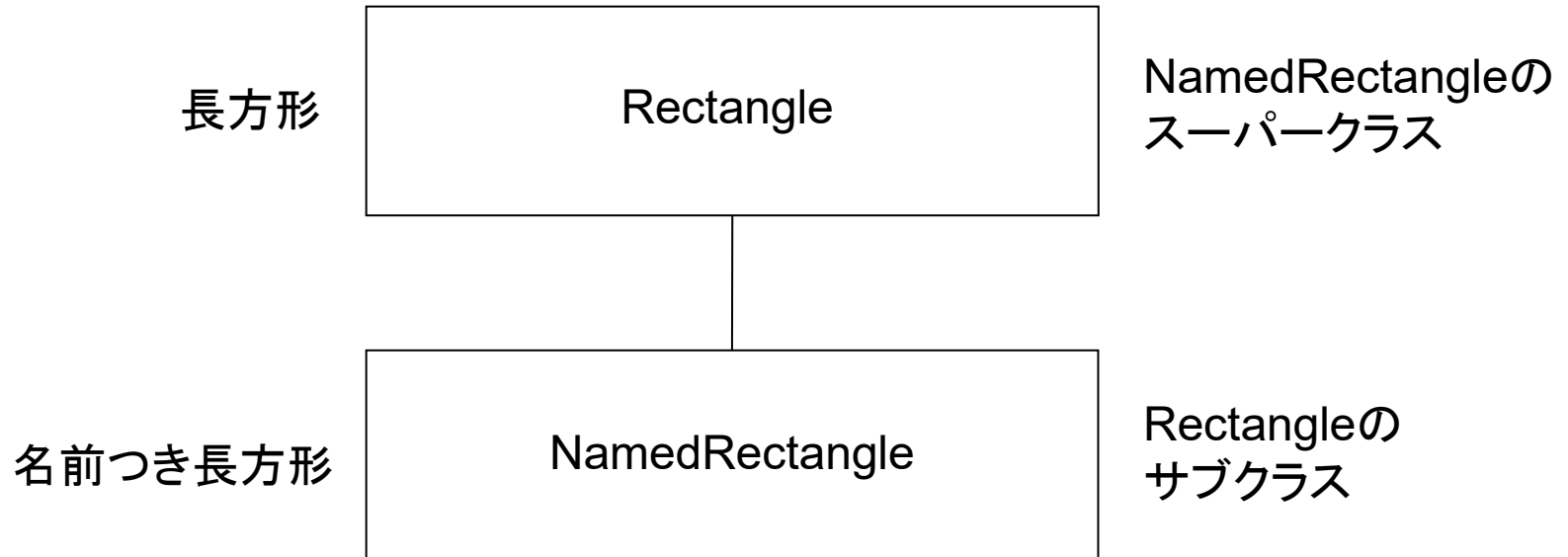
```
class クラス名 extends スーパークラス名 {  
    ...  
}
```

Rectangleクラスを拡張して, NamedRectangleクラスを作成する

```
class NamedRectangle extends Rectangle{  
    ...  
}
```

スーパークラスを指定せずにクラス宣言すると, 自動的にObjectクラスのサブクラスとして宣言される. つまり,
class NamedRectangle extends Object {
と同じ意味.

クラスの関係



- ・NamedRectangleクラスはRectangleクラスのサブクラスである
- ・Rectangleクラスを拡張して, NamedRectangleクラスを作成する

フィールドの継承

- サブクラスはスーパークラスのフィールドとメソッドを受け継ぐ。このことを「継承」とよぶ。
- つまり、スーパークラスが持っているフィールドは、サブクラスも持つ。
- NamedRectangleのフィールド宣言には、width, heightはないが、スーパークラスで宣言されているので、二重に宣言する必要はない。
- NamedRectangleのオブジェクトnrに対して、nr.width, nr.heightのようなフィールドアクセスができる
 - フィールド参照
System.out.println(nr.width);
 - フィールド値の設定
nr.height=123;

フィールドの継承の例

```
class Rectangle {  
    int width;  
    int height;  
    . . . コンストラクタ, メソッドなど  
}  
//サブクラスの宣言  
public class NamedRectangle extends Rectangle { //Rectangleを拡張  
    String name; //追加のフィールドのみ宣言  
    NamedRectangle (String n, int w, int h) { // コンストラクタ  
        this.name = n;  
        this.width = w; //スーパークラスのフィールドを利用  
        this.height = h; //スーパークラスのフィールドを利用  
    }  
    public static void main(String[] args) {  
        NamedRectangle nr= new NamedRectangle("Ex1", 123, 45);  
        System.out.println(nr.width);  
        nr.width=120;  
    }  
}
```

メソッドの継承

- スーパークラスが持っているメソッドは、サブクラスも利用可能である。

```
class Rectangle {  
    ...  
    int getArea() {  
        return this.width * this.height;  
    }  
}
```

//サブクラスの宣言

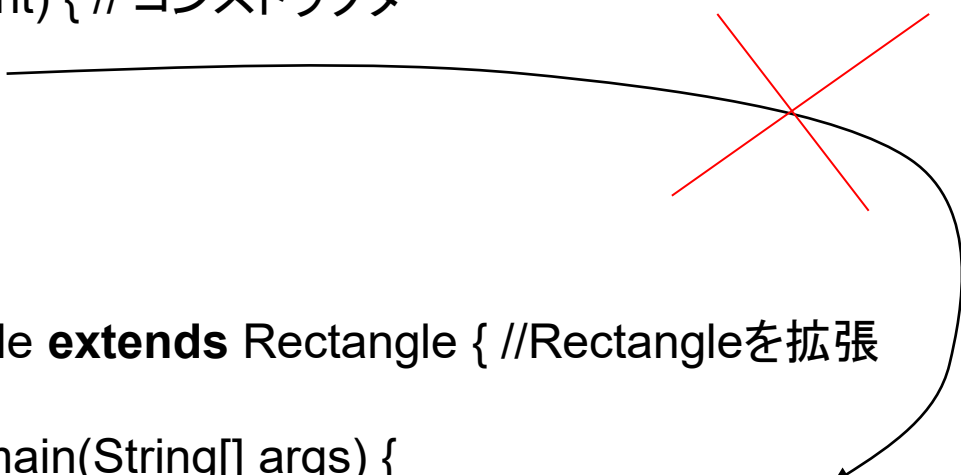
```
public class NamedRectangle extends Rectangle { //Rectangleを拡張  
    ...  
    void printNamedRectangle() {  
        System.out.println(this.name+" 横:"+this.width+" 高さ:"+this.height);  
        int area = this.getArea(); //スーパークラスのメソッドを利用  
        System.out.println("面積:"+area);  
    }  
    ...  
}
```

getArea()はRectangleクラスで宣言されているメソッドであるが、NamedRectangleクラスのオブジェクトであるnrに対しても適用可能である。

継承におけるコンストラクタの扱い

- コンストラクタは**継承されない**ので、スーパークラスのコンストラクタをそのまま使うことはできない。

```
class Rectangle {  
    int width;  
    int height;  
    Rectangle (int width, int height) { // コンストラクタ  
        this.width = width;  
        this.height = height;  
    }  
    ...  
}  
  
public class NamedRectangle extends Rectangle { //Rectangleを拡張  
    ...  
  
    public static void main(String[] args) {  
        × NamedRectangle nr= new NamedRectangle(123, 45);  
        nr.printNamedRectangle();  
    }  
}
```



サブクラスのコンストラクタ

- スーパークラスの引数なしコンストラクタが自動的に呼び出される
- スーパークラスから継承されているフィールドを初期化しなくてもよい.

```
class Rectangle {  
    int width;  
    int height;  
    Rectangle () { // 引数なしコンストラクタ  
        width=0;  
        height=0;  
    }  
}  
  
public class NamedRectangle extends Rectangle { //Rectangleを拡張  
    String name; //追加のフィールドのみ宣言  
    NamedRectangle() {  
        name="noname"; ここで呼ばれている  
    }  
    NamedRectangle(String nm) {  
        name=nm; ここで呼ばれている  
    }  
}  
  
...  
}
```

サブクラスのコンストラクタ(Cont.)

- コンストラクタのはじめに, スーパークラスの引数なしコンストラクタ, `super()`が自動的に呼び出される. この場合, `name` フィールドは変数`nm`で初期化され, 残りのフィールドについては, スーパークラスの引数なしコンストラクタが呼び出される.
- 以下のように書いても同じことを意味する. つまり, `super()`は省略可.

```
NamedRectangle() {  
    super();  
    name="noname";  
}  
NamedRectangle(String nm) {  
    super();  
    name=nm;  
}
```

スーパークラスの引数つきコンストラクタの呼び出し

- スーパークラスの引数付きのコンストラクタを呼び出すには、引数を入れる.

```
NamedRectangle (String n) { // コンストラクタ
    super (200, 32); //スーパークラスの引数つきコンストラクタの呼び出し
    this.name=n;
}
```

width=200, height=32, name=nでオブジェクトが初期化される

自分のクラスのコンストラクタの 明示的な呼び出し

- 自分のクラスのコンストラクタを呼び出すにはthis()を用いる

```
NamedRectangle () { //コンストラクタ1
    this("NO NAME"); //コンストラクタ2の呼び出し
}
NamedRectangle (String n) { // コンストラクタ2
    super (200, 32); //スーパークラスの引数つきコンストラクタの呼び出し
    this.name=n;
}
```

オーバーライド

- 継承したフィールドやメソッドを変更, つまり上書きすることができる

```
class Rectangle {  
    ...  
    int getArea() {  
        return this.width * this.height;  
    }  
}  
  
public class NamedRectangle extends Rectangle { //Rectangleを拡張  
    void printNamedRectangle() {  
        System.out.println(this.name+" 横:"+this.width+" 高さ:"+this.height);  
        int area = this.getArea(); //NamedRectangleクラスのgetArea()を使用  
        //スーパークラスのメソッドを使いたい場合は"super."をつける  
        //int area = super.getArea()  
        System.out.println("面積:"+area);  
    }  
    int getArea() { // RectangleクラスのgetArea()をオーバーライド  
        System.out.println("NamedRectangleクラス:"+this.name);  
        return this.width * this.height;  
    }  
    ...  
}
```

継承

継承とアクセス制御

- privateがついているメソッドやフィールドは、サブクラスに継承されない。
 - publicもprivateも付いていない場合は、同じパッケージ（ディレクトリ）内で継承可能。
- finalという修飾子をつけると、そのクラスは拡張禁止となる。
 - String, Integer, Systemなどはfinalクラスである。
- final修飾子がついたフィールドは、サブクラスで値を変更することができない。
- final修飾子がついたメソッドはオーバーライドできない
 - サブクラスに継承されるが、オーバーライドはできない。

Javaドキュメント

Mathクラスの説明

java.lang クラス Math
java.lang.Object └ java.lang.Math
public final class Math extends Object

MathクラスはObjectクラスを継承している

Mathクラスのメソッドの説明

メソッドの修飾子

メソッド名

引数

戻り値

static double	sin (double a)	指定された角度の正弦 (サイン) を返します。
static double	sinh (double x)	double 値の双曲線正弦を返します。
static double	sqrt (double a)	double 値の正しく丸めた正の平方根を返します。
static double	tan (double a)	指定された角度の正接 (タンジェント) を返します。

Javaドキュメントを使いこなす

- Javaのドキュメントを読めるようにしておくでプログラムの幅が大きく広がる
- <http://docs.oracle.com/javase/jp/8/api/>

まとめ

- Javaではクラスが階層構造(木構造)になっている.
- Javaでは(基本的に)多重継承が不可である.
- 継承をうまく利用するとプログラムが簡潔になり, クラスの概念がより明確になる.
- プログラムの設計を最初にしっかり行うことが重要になる.