



cp-3. サブクラス, 継承

(C++ オブジェクト指向プログラミング演習,
全3回)

<https://www.kkaneko.jp/cc/cpp/index.html>

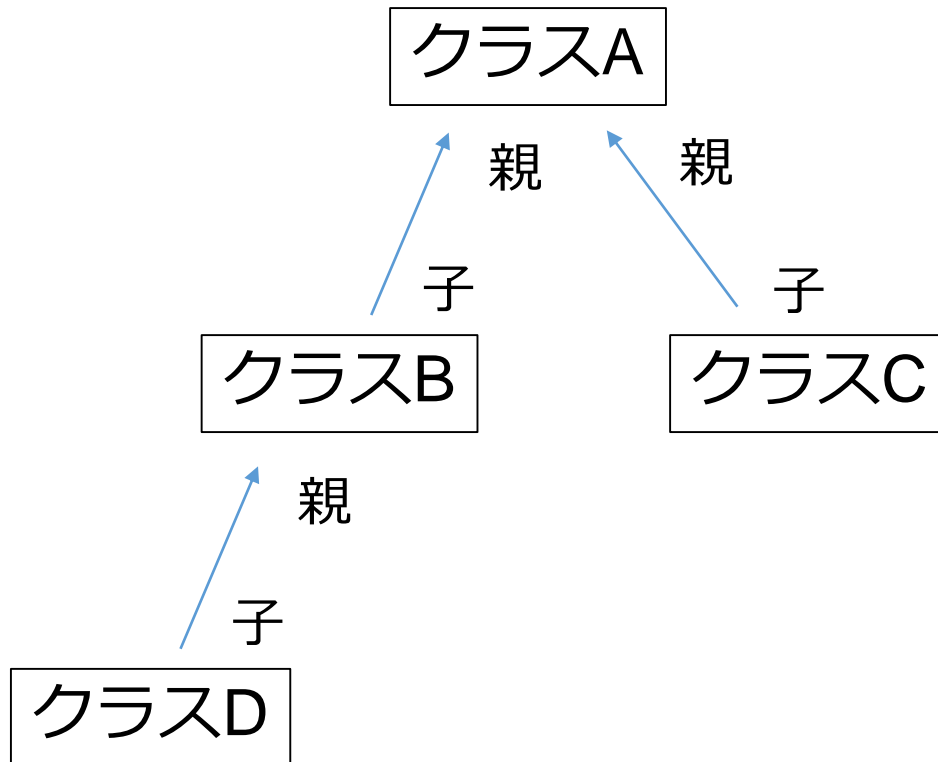
金子邦彦



クラス階層



クラス階層とは、複数のクラスが親子関係をなすこと



継承



- **継承**とは、**親クラスの属性とメソッド**を**子クラス**が**受け継ぐ**こと
- **親クラス**のことを「**スーパークラス**」、**子クラス**のことを「**サブクラス**」ともいう

オブジェクトの生成



- 次の**オブジェクト**を生成

| | | | |
|-----------|-----------|-----------|---------------|
| b1 | 3 | 4 | 0 |
| | _x | _y | _color |

- オブジェクト生成を行うプログラム

```
ColorBall* b1 = new ColorBall( 3, 4, 0 );
```



クラスの類似性

- 類似した2つの**クラス**

Ball

属性

`_x`
`_y`

ColorBall

属性

`_x`
`_y`
`_color`

↔
`_x, _y` は同じ
`_color` の有り
無しが違う

メソッド

`distance_to_0`

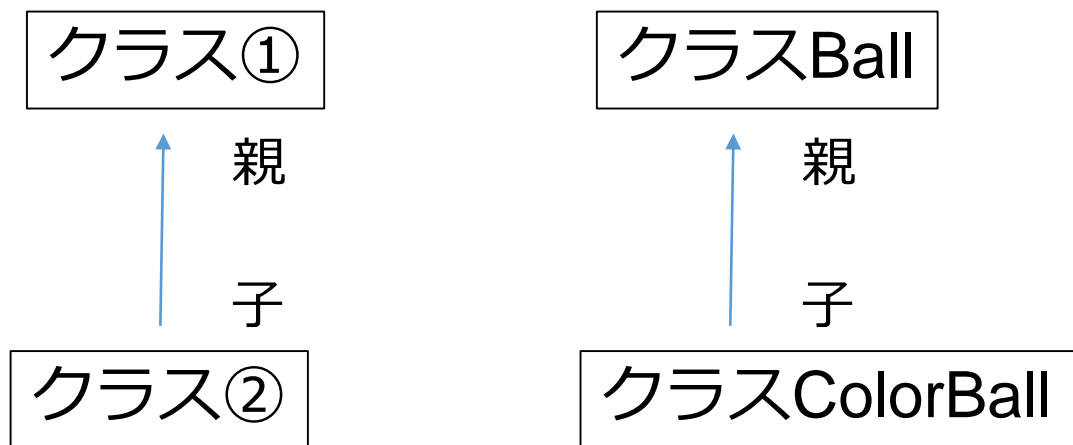
`distance_to_0`

↔
メソッドの名前も
中身も全く同じとする

クラスの親子関係



- クラス①が**親**，クラス②が**子**であるとき
 - クラス②は，クラス①の**属性**と**メソッド**を**すべて持つ**
 - クラス②で，クラス①にない**属性**や**メソッド**が**追加される**ことがある

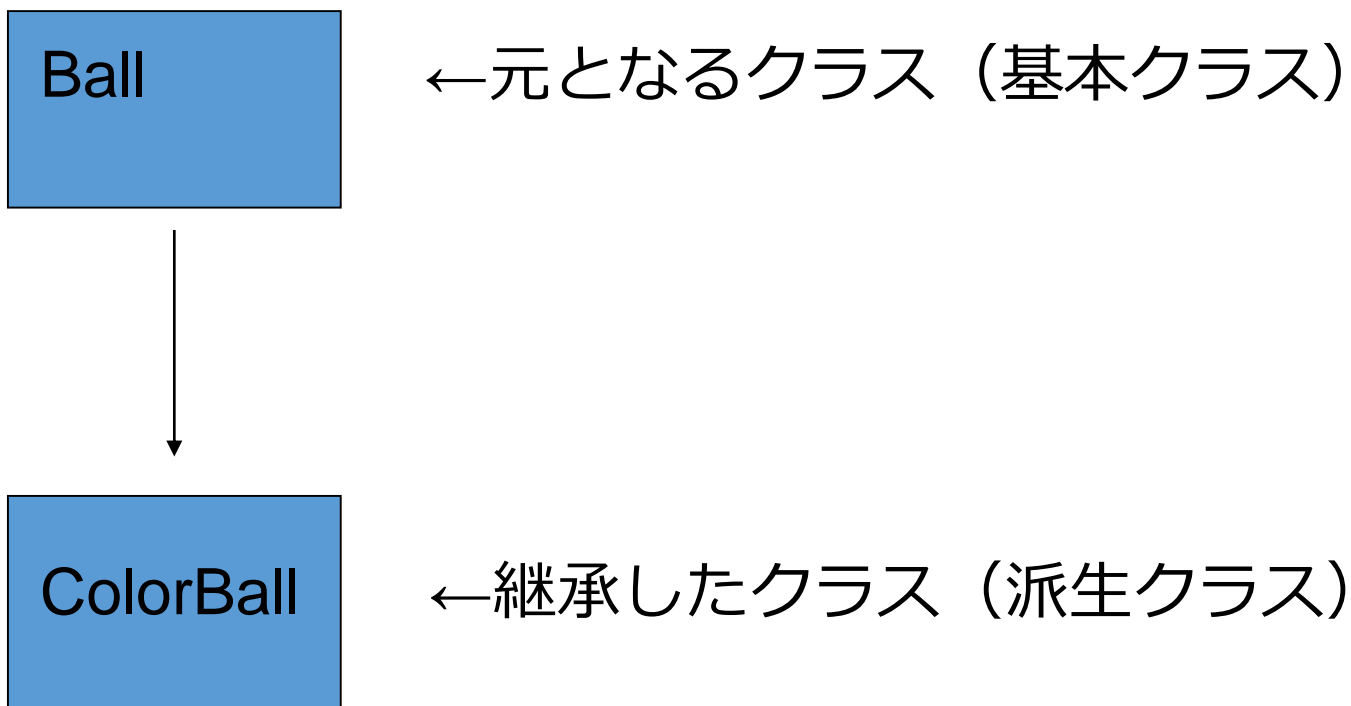


属性 `_color` を追加

派生クラス



- あるクラスを継承して作成したクラス



ソースコード



ファイル名: Ball.h

```
#pragma once
class Ball {
protected:
    double _x, _y; } 属性 (メンバ変数ともいう)
public:
    Ball( const double x, const double y );
    Ball( const Ball& ball );
    Ball& operator= ( const Ball& ball );
    ~Ball();
    double distance_to_0() const;
    double x() const { return this->_x; };
    double y() const { return this->_y; }; } アクセサ
};
```

protected 指定により, サブクラスから属性アクセス可能

ソースコード



ファイル名: Ball.cpp

```
#include "Ball.h"
#include <math.h>
Ball::Ball( const double x, const double y, const int color ) : _x( x ), _y( y )
{
    /* do nothing */
}
Ball::Ball( const Ball& ball ) : _x( ball.x() ), _y( ball.y() )
{
    /* do nothing */
}
Ball& Ball::operator= (const Ball& ball )
{
    this->_x = ball.x();
    this->_y = ball.y();
    return *this;
}
Ball::~~Ball()
{
    /* do nothing */
}
double Ball::distance_to_0() const
{
    return sqrt( ( this->x() * this->x() ) + ( this->y() * this->y() ) );
}
```

ソースコード



ファイル名: ColorBall.h

```
#pragma once
#include "Ball.h"
class ColorBall : public Ball {
protected:
    int _color;
public:
    ColorBall( const double x, const double y, const int color );
    ColorBall( const ColorBall& ball );
    ColorBall& operator= ( const ColorBall& ball );
    ~ColorBall();
    int color() const { return this->_color; };
};
```

ソースコード



ファイル名: ColorBall.cpp

```
#include "ColorBall.h"
ColorBall::ColorBall( const double x, const double y , const int color ) : Ball( x, y ),
_color( color )
{
    /* do nothing */
}
ColorBall::ColorBall( const ColorBall& ball ) : Ball( ball.x(), ball.y() ), _color( ball.color() )
{
    /* do nothing */
}
ColorBall& ColorBall::operator= (const ColorBall& ball )
{
    this->_x = ball.x();
    this->_y = ball.y();
    this->_color = ball.color();
    return *this;
}
ColorBall::~ColorBall()
{
    /* do nothing */
}
```

ソースコード



ファイル名: main.cpp

```
#include <stdio.h>
#include "ball.h"
#include "ColorBall.h"
int main( int argc, char** argv )
{
    ColorBall* b1 = new ColorBall( 3, 4, 0 );
    fprintf( stderr, "b1: %f, %f, %d¥n", b1->x(), b1->y(), b1->color() );
    delete b1;
}
```

アクセサによる
属性アクセス



Microsoft Visual Studio のデバッグコンソール

```
b1: 3.000000, 4.000000, 0
```

```
C:\Users\user\source\repos\ConsoleApp1\bin\Debug\net48\ConsoleApp1.exe  
ました。  
このウィンドウを閉じるには、任意のキーを
```

プログラムの実行結果が
表示されている

まとめ



- **クラス階層**とは、複数のクラスが親子関係をなすこと
- クラス①が**親**，クラス②が**子**であるとき
 - クラス②は，クラス①の**属性とメソッド**をすべて持つ
 - クラス②で，クラス①にない**属性やメソッド**が追加されることがある
- **親子関係**の指定は，「`class ColorBall : public Ball`」のように書く．ColorBall が子，Ball が親．
- **継承**とは，**親クラスの属性とメソッド**を子クラスが受け継ぐこと
- **親クラス**のことを「**スーパークラス**」，**子クラス**のことを「**サブクラス**」ともいう