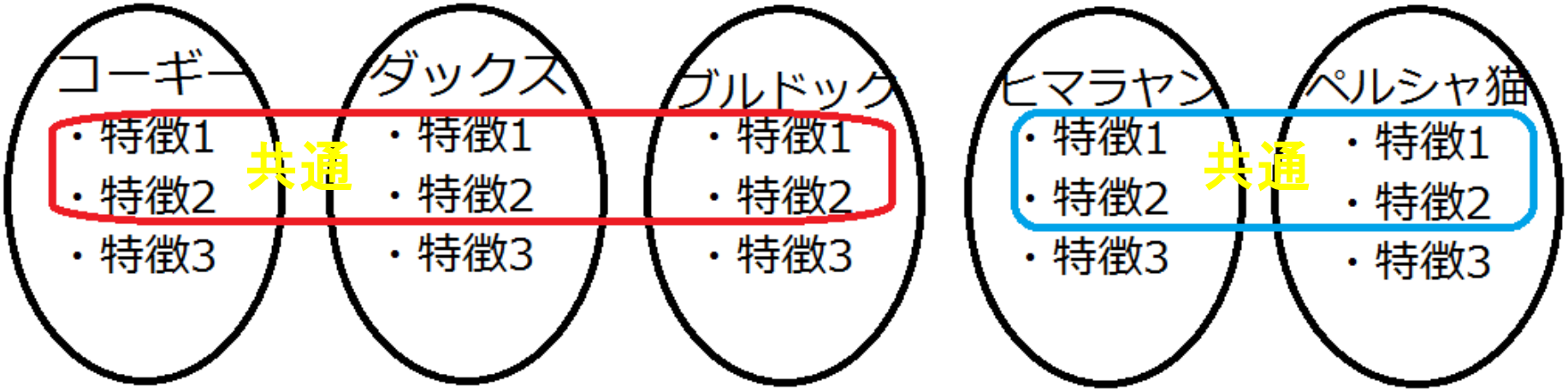
[Prog2] Programming C++ (C6) Exercise Guide (Ex07)

10/26, Thursday 3rd period.

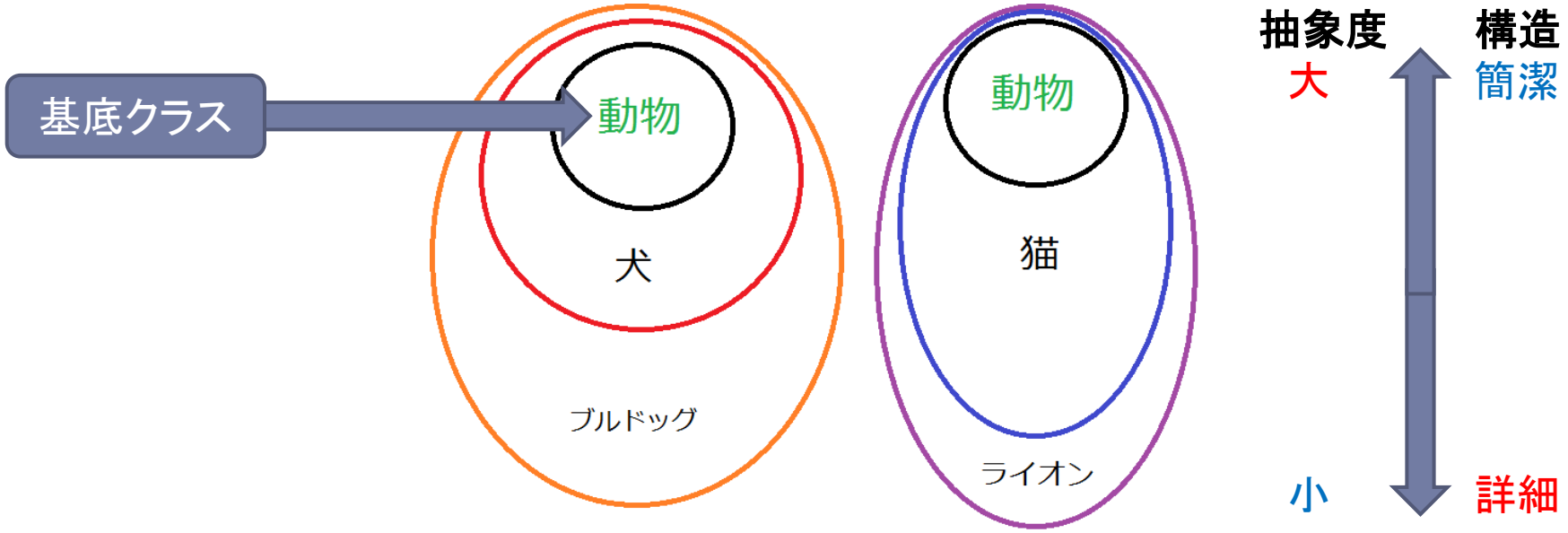
Ex07　クラスの継承 (Inheritance)

- よく似た構造のクラス型をたくさん定義するときに、
- 同じようなメンバーを持つクラスを繰り返し作るのは面倒



ちょっと違うだけのデータ型なのに、
同じようなデータ構造をたくさん書かないといけないのか…

Ex07　クラスの継承 (Inheritance)



⇒ クラス間の共通事項を見つけ出し、抽象化クラスを作る

⇒ その抽象化クラスに、特徴を付け足して具体的なクラスを作る

＝ 継承 (Inheritance)

例：犬、猫、人間 ⇒ 抽象的に言うと「動物」

　　おじさん、おばさん、友達、先生 ⇒「人間」

クラスの継承

- 継承の仕方

class [継承して新しく作りたいクラスの名前] : public [継承元の名前]

- Animalクラス

```
1  class Animal{//Animalクラス
2  private:
3    string name;              //動物の名前
4    int age;                  //年齢
5    int hungry;               //満腹度
6  public:
7    Animal();                 //デフォルトコンストラクタ
8    void eat(int);            //引数に与えられた量のエサを食べる
9    void show_status(void);   //動物の調子を見る（出力するだけ）≠ゲッター（アクセス関数）
10   void set_name(string);    //動物の名前をセットする（セッター）
11   void set_age(int);        //動物の年齢をセットする（セッター）
12 };
```

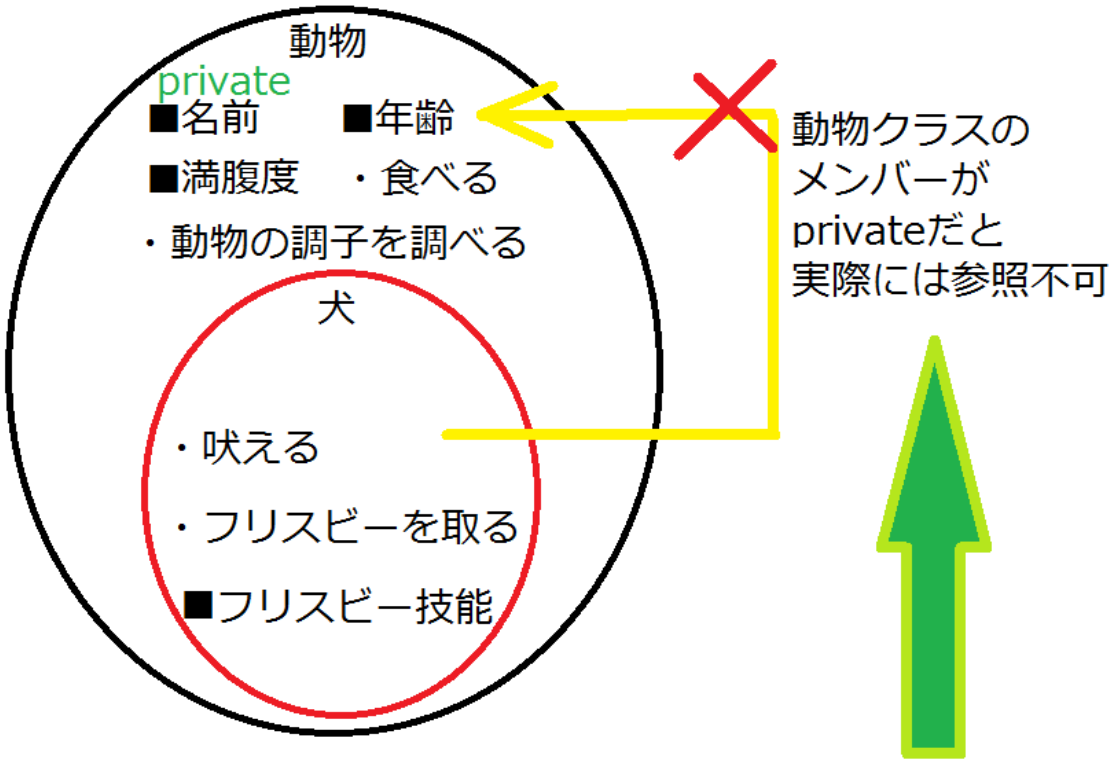
- Dogクラス（Animalクラスから継承する）

```
1  class Dog : public Animal  //Dogクラス（Animalから継承）
2  {
3  private:
4    int ability;          //芸を行うための能力値
5  public:
6    Dog();
7    int frisbee(void);    //フリスビーを投げて、この動物に取らせる
8    void bark(void);      //この動物を吠えさせる
9  };
```

クラスの継承：隠蔽されるメンバー

- 継承するときには、継承元のクラスのアクセシビリティに注意

⇒ アクセス指定子 **private**は、あくまで そのクラス内からしか アクセス不可能



privateはダメか…
⇒ じゃあ public か？
⇒ どこからも丸見えじゃないか!!

「犬」クラスは「動物」クラスを継承している
⇒ 「動物」クラスのデータも「犬」クラスは持っているはず

クラスの継承：隠蔽されるメンバー

アクセス指定子 **protected** を使えば、

そのクラスおよび、継承クラス内からアクセス可能

継承元クラスのメンバー変数を **protected** 指定に変更！

```
private:
   string name;
   int age;
   int hungry;
↓
protected:
   string name;
   int age;
   int hungry;
```

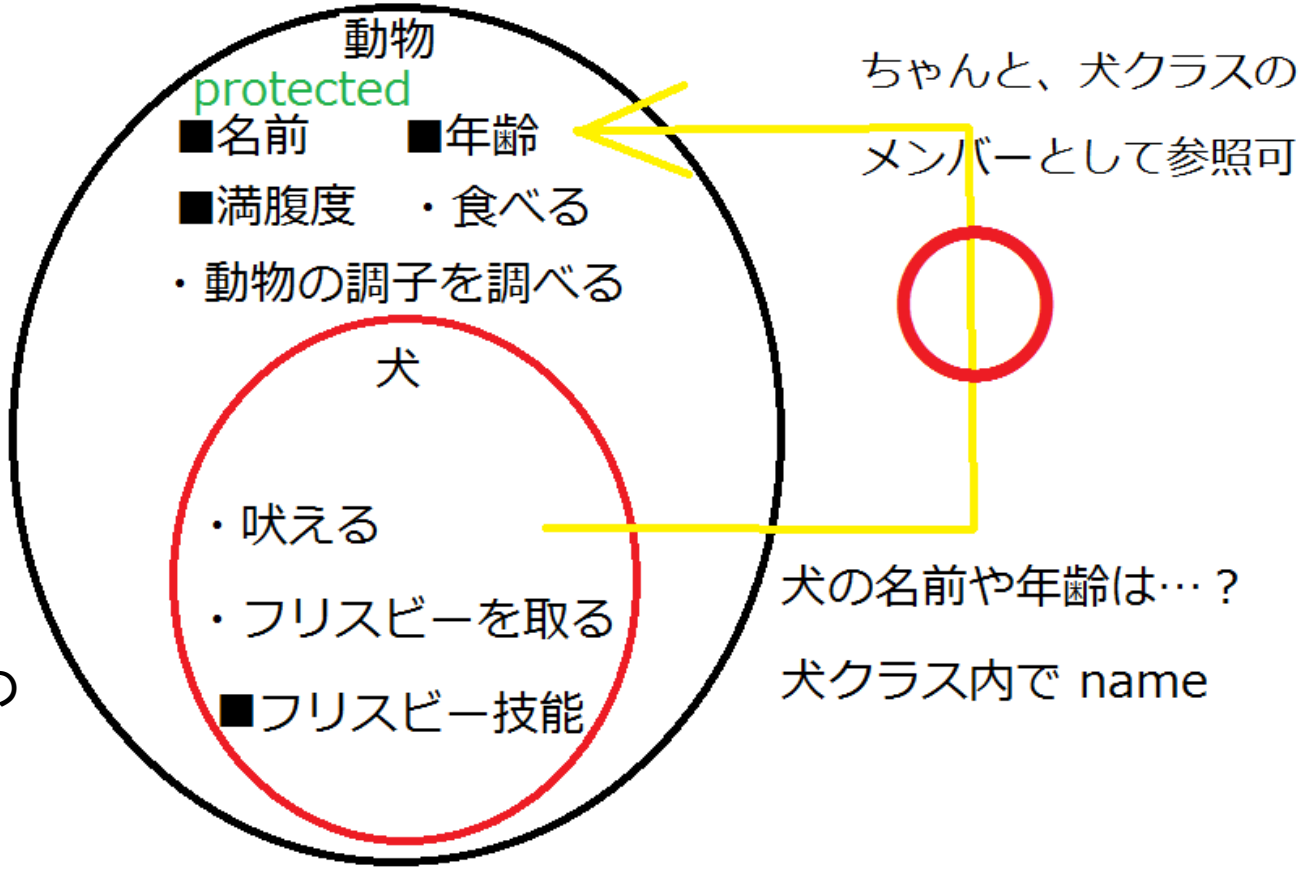
- **ex07-2.cc**

```
#include <iostream>
#include <string>
#include <cstdlib>      //乱数用 （犬系のチャレンジ判定用）

using namespace std;

class Animal{//Animalクラス
protected:
  string name;              //動物の名前
  int age;                  //年齢
  int hungry;               //満腹度
public:
  Animal();                 //デフォルトコンストラクタ
  void eat(int);            //引数に与えられた量のエサを食べる
  void show_status(void);   //動物の調子を見る（出力するだけ）≠ゲッター（アクセス関数）
  void set_name(string);    //動物の名前をセットする（セッター）
  void set_age(int);        //動物の年齢をセットする（セッター）
};

  class Dog : public Animal  //Dogクラス（Animalから継承）
  {
  protected:
    int ability;        //芸を行うための能力値
  public:
    Dog();
    int frisbee(void);  //フリスビーを投げて、この動物に取らせる
    void bark(void);    //この動物を吠えさせる
  };
```

クラスの継承：隠蔽されるメンバー

アクセス指定子 **protected** を使えば、

そのクラスおよび、継承クラス内からアクセス可能



これなら、関連するクラスの
外からは覗かれないよ!!
（privateと同じ状態）

クラスの継承

継承関係

・ クラスA、B、Cがある

クラスB、クラスCは**クラスAを継承して**作ったものとする

・クラスBは、クラスAとBのメンバーを利用できる

・クラスCは、クラスAとCのメンバーを利用できる

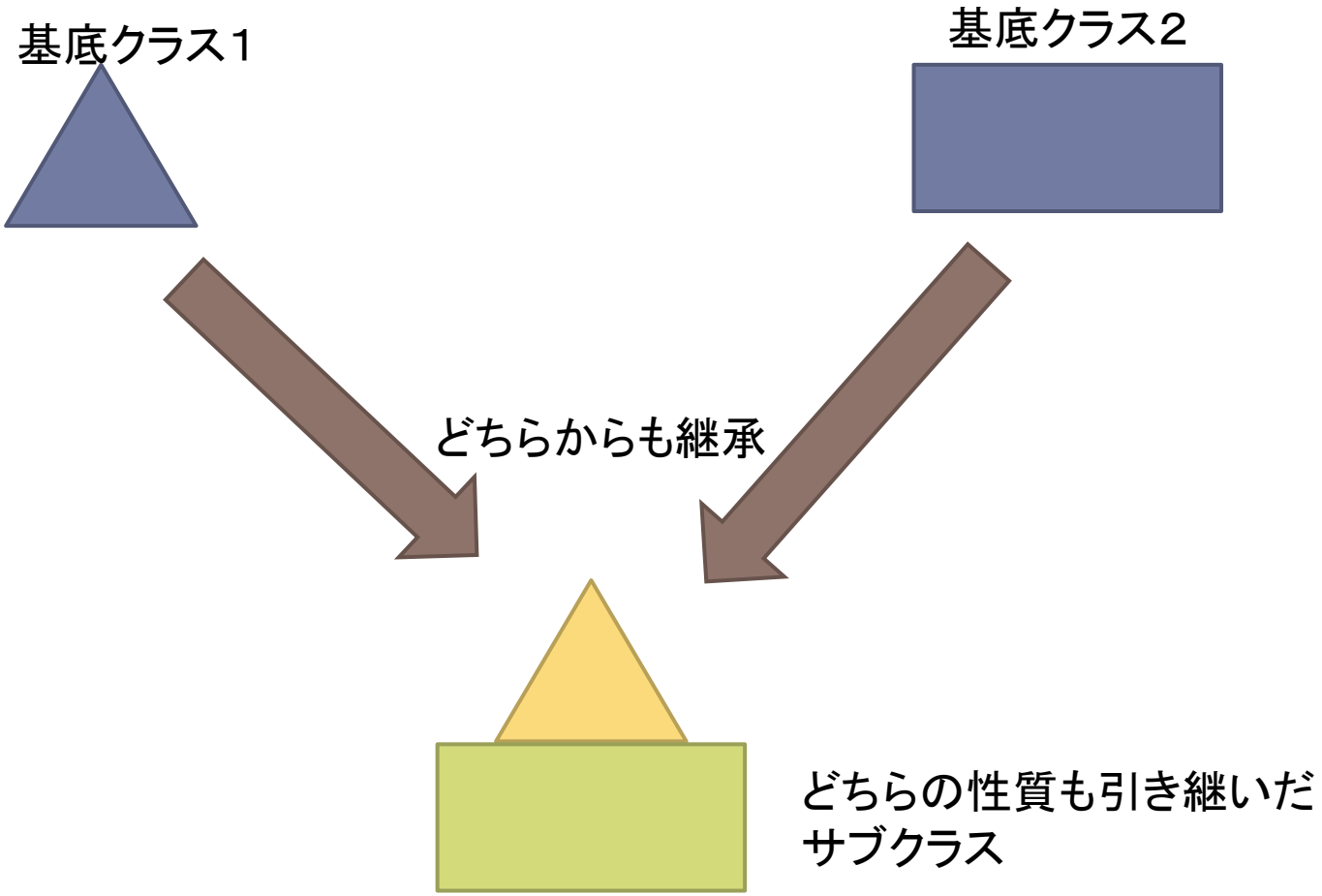
・継承クラス生成時のコンストラクタは、

親クラスのものから順に呼ばれる

⇒ Bの実体を生成すると、まずAのコンストラクタが呼ばれる

多重継承の基礎的な話

・C++独特のクラス継承



多重継承のやり方は基本的に同じ

class [継承して新しく作りたいクラスの名前] : public [継承元の名前], …

```
6  class A{
7  protected: //継承されるprivateはprotected指定に
8    int x;
9    double y;
10
11 public:
12   A(){
13     x = 3;
14     y = 1.25;
15     cout << "Mother is A." << endl;
16   }
17   ~A(){
18     cout << "Mother was dead." << endl;
19   }
20 };
21
```

```
23 class B{
24 protected: //継承されるprivateはprotected指定に
25  string s;
26  double y;
27
28 public:
29   B(){
30     s = "test";
31     y = 8.0;
32     cout << "Father is B." << endl;
33   }
34   ~B(){
35     cout << "Father was dead." << endl;
36   }
37 };
38
```

```
40 class KID : A , B {
41 private:
42   double ans_y;
43 public:
44   KID(){
45     cout << "Mom!! Dad!!  (A kid was born.)" << endl;
46   }
47   ~KID(){
48     cout << "I loved Mom and Dad." << endl;
49   }
50   void calcy(void){ ans_y = A::y * B::y; } //AとBのクラスのメンバーyの積（※名前が被るので名前空間の明示が必要）
51   void setstr(string a){ s = a; } //文字列をそのまま返すだけ
52   void xsquare(void){ x = x*x; } // xを2乗する（※xはMotherの方にだけあるので、名前空間を与えなくても自動で引き継いで判別）
53   string printstr(void){ return s; }  //Fatherから引き継いだ x へ値を代入
54   int getx(void){ return x; } //Motherから引き継いだ x へ値を代入
55   double getmuly(void){ return ans_y; } //自分自身の ans_y を返す
56   void setMomy(double v){ A::y = v; } //Motherのyに代入
57   void setFaty(double v){ B::y = v; } //Fatherのyに代入
58 };
59
```

複数のクラスから継承するときは、
カンマ(,)でつないでいく

多重継承で作られたクラスは、
どちらの親の性質も受け継ぐ

多重継承のやり方は基本的に同じ

実体を宣言したとき…

- コンストラクタは、先に書いた親のクラスから（左側）順に
- 最後に継承したクラスのコンストラクタが呼び出される

```
40  class KID : A , B {
41  private:
42    double ans_y;
43  public:
44    KID(){
45      cout << "Mom!! Dad!!  (A kid was born.)" << endl;
46    }
47    ~KID(){
48      cout << "I loved Mom and Dad." << endl;
49    }
50    void calcy(void){ ans_y = A::y * B::y; } //AとBのクラスのメンバーyの積（※名前が被るので名前空間の明示が必要）
51    void setstr(string a){ s = a; } //文字列をそのまま返すだけ
52    void xsquare(void){ x = x*x; } // xを2乗する（※xはMotherの方にだけあるので、名前空間を与えなくても自動で引き継いで判別）
53    string printstr(void){ return s; }  //Fatherから引き継いだ x へ値を代入
54    int getx(void){ return x; } //Motherから引き継いだ x へ値を代入
55    double getmuly(void){ return ans_y; } //自分自身の ans_y を返す
56    void setMomy(double v){ A::y = v; } //Motherのyに代入
57    void setFaty(double v){ B::y = v; } //Fatherのyに代入
58  };
59
```

デストラクタは逆順

- 継承したクラス、親（右から順）に

多重継承のやり方は基本的に同じ

- 複数の親が、同じ名前のメンバー変数を持っていたら？

⇒　エラーにはならず、どちらも引き継ぐことができる

- ただし、参照するときはどちらの親のものかを

明示しなければ、コンパイラが判断できない

```
void calcy(void){ ans_y = A::y * B::y; } //AとBのクラスのメンバーyの積（※名前が被るので名前空間の明示が必要）
void setstr(string a){ s = a; } //文字列をそのまま返すだけ
void xsquare(void){ x = x*x; } // xを2乗する（※xはMotherの方にだけあるので、名前空間を与えなくても自動で引き継いで判別）
string printstr(void){ return s; }  //Fatherから引き継いだ x へ値を代入
int getx(void){ return x; } //Motherから引き継いだ x へ値を代入
double getmuly(void){ return ans_y; } //自分自身の ans_y を返す
void setMomy(double v){ A::y = v; } //Motherのyに代入
void setFaty(double v){ B::y = v; } //Fatherのyに代入
```

A::y　⇒　親クラスAのメンバー変数y

B::y　⇒　親クラスBのメンバー変数y

親のクラス名 : : メンバー変数

重複が無ければ、名前空間が無くても自動的に判別される