البرمجة و الخوارزميات 2

الدكتور: سمير جعفر

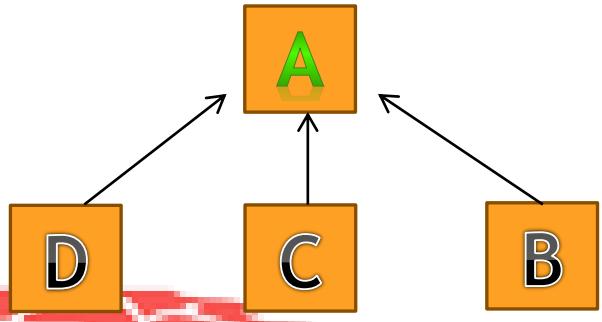
المحاضرة : الثانية عشر " والأخيرة "

التاريخ : ٢٠/١٢/٢٠١٦

إعداد : فهمي القاضي && محمد فليون&&سندس العمر

سنكمل معكم اصدقائي بالمحاضرة الأخيرة من مادة البرمجة والخوارزميات ببحث الوراثة .

- يمكن أن يكون للصف الواحد عدة أبناء :
كما في الشكل:



للصف A الأبناء B,C,D و يرثون (المحمي،العام) في A جميعهم و لكن يختلف نوع الوراثة

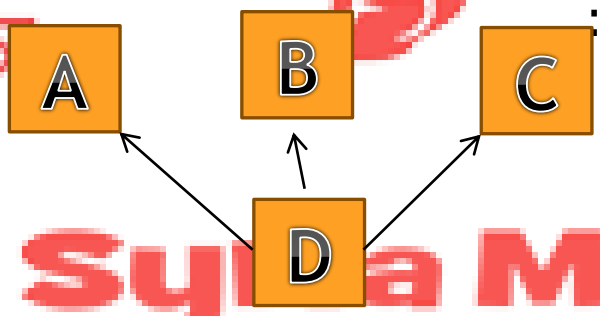
$class\ A\{\ldots\ldots\ldots\};$

$class\ B$: نوع الوراثة $A\{\ldots\ldots\};$

$class\ C$: نوع الوراثة $A\{\ldots\ldots\};$

$class\ D$: نوع الوراثة $A\{\ldots\ldots\};$

- سمحت لغة C++ (خلافاً للغات أخرى) أن يرث الصف من عدة صفوف أي أن يكون للصف عدة آباء وتدعى هذه الوراثة "بالوراثة المتعددة":
كما في الشكل:



$class\ A\{\ldots\ldots\ldots\};$

$class\ B\{\ldots\ldots\ldots\};$

$class\ C\{\ldots\ldots\ldots\};$

$class\ D$: نوع الوراثة A, نوع الوراثة B, نوع الوراثة C{

$D(\ldots): A(\ldots.), B(\ldots.), C(\ldots.)\{\ldots.\}$

};

لاستدعاء بواني الاباء لـ D

مثال: بناء الأب داخل الابن بغير باني الأب :

```
class A{
public:
int x;
A ( ){x = 0;}
A(int a){x = a;}
};
class B : public A {
public:
double y;
B(double b){y = b;}
```

اللغة تلقائياً ستستدعي باني الاب الخالي : موجود ضمنياً

```
B(int a, double b):A()
{y = b;}
};
```

قمنا باستدعاء a وليس بتعريفها لأنها معرفة بصف A

مثال(٢) لصف له" أبين"

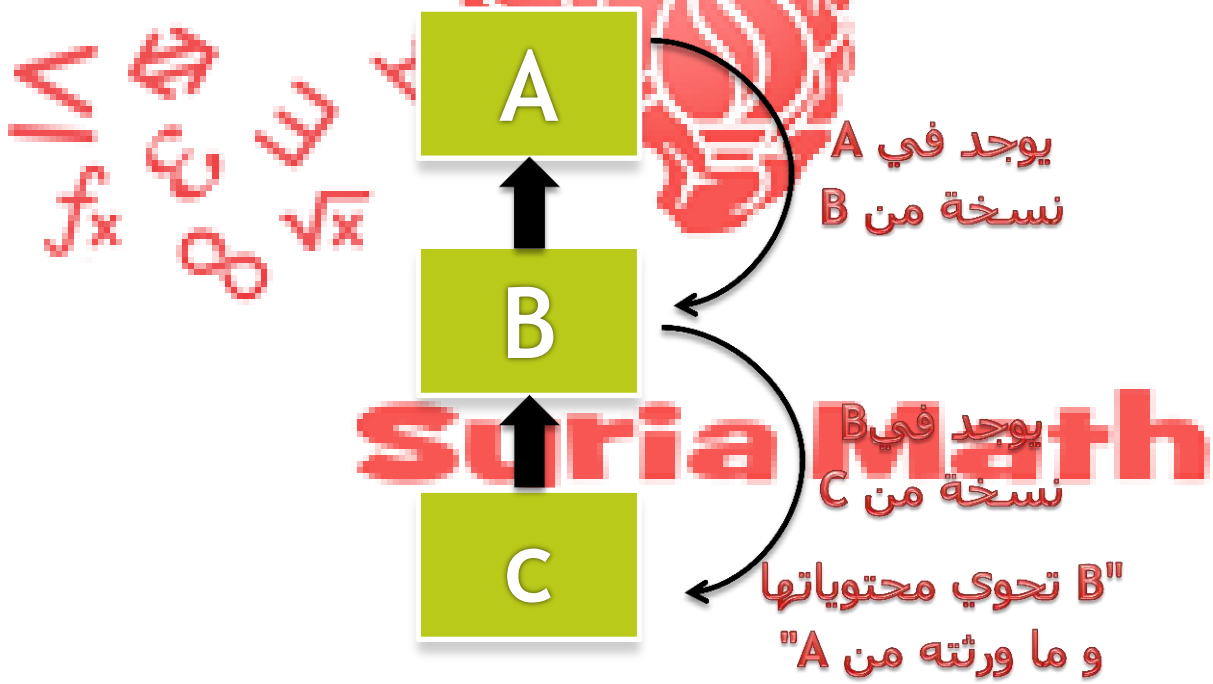
```
class A{
public :
int x;
A(int x){this → x = x;}
};
class B {
public:
char c;
B(char c){this → c = c;}
};
class C : public A, public B{
float f;
```

$$C(float\ f, char\ z, int\ a): A(a), B(z)\{$$
$$this \rightarrow f = f;\}$$
$$\};$$
$$int\ main\ (\ )\{$$
$$C\ a = C(4.1,'w', 11);$$
$$return\ 0;\}$$

ملاحظة : عندما نقوم بإنشاء بانٍ للصف الابن يجب أن يتضمن استدعاءً لباني الأب :

١- أن نترك اللغة تقوم باستدعائه و ذلك عندما يحوي باني الأب بانٍ افتراضي أو خالٍ

٢- أو أن نقوم نحن بكتابته في الباني الابن كما في المثال السابق



و لكن ما الفائدة من مثل هذه الوراثة !!

في الحقيقة هي مفيدة من أجل الانتقال من العام إلى الخاص ،، فيكون الأب هو الحالة الأعم و الابن الأول يقوم بتخصيص بعض الأمور و الابن الثاني يخصص أكثر و هكذا ....

مثلاً:

شكل هندسي -> شكل هندسي منتظم -> شكل هندسي منتظم و هو مربع .....

فمن هذا المثال نستنتج أن الهدف من الوراثة هو التعدية أو الزيادة و قد تعني الزيادة بأنه زيادة في التفاصيل و التخصيص و ليس بالكم.

مثال ٣:

```
class Shape {
public :
char * name ;
int nb ;
void print ( ){
cout << "I am shape " << name;
}
};
```

تمثل سلسلة محارف

انشاء صف مربع :

```
class Square : public shape {
double l;
square (double l){
name = "square";
nb = 4;
this → l = l;
}
void print ( ){
```

ملاحظة: يمكننا بالوراثة فقط اعادة كتابة نفس الدالة او اعادة تعريفها

و تجدر الإشارة هنا إلى أن التحميل الزائد يمكن أن يحدث بالصفوف و لكن يجب أن يكون هذا التحميل ضمن الصف الواحد ولا يمكن تنفيذ التحميل الزائد على دالتين كل منهما في صف مختلف لأننا حينما نكتب في نهاية كل صف " }; " فإننا نعني أن كل ما قبلها هي للصف نفسه و إن كان محدد الوصول PUBLIC:

بالعودة لمثالنا الصف الأخير لديه دالة print() الموروثة من الأب و قمنا نحن بانشاء دالة خاصة به و عن طريق دالته تستدعى دالة الأب.

```
cout << "I am square";<<name;
}
double primetre ( ){
return (nb * l);
}
};
class tringle : public shape {
double x, y, z;
tringle (double x, double y, double z){
this → x = x;
this → y = y;
this → z = z;
}
name = " tringle";
nb = 3; }
void print ( ){
cout << " I am tringle " << name ; }
double primetre ( ){
return (x + y + z);
}
};
double p(Shape s){
return (s.primetre( ) ); }
```

```
int main (){
Shape s = Shape ();
s.print() ;
square s1 = square (2);
cout << s1.primetre ();
s1.print ();
tringle t = tringle (2,3,5);
cout << t.primetre ();
t.print ();
Shape q = square (3.2);
cout << p(q);
```

- الأبن هو أب دائماً .
- الأب ليس أبن دائماً .

مبدأ تعددية الاشكال ((المبدأ الثالث))

أي غرض من الصف الاب يمكن أن يأخذ اي شكل من اشكال الابناء .

و بالتالي في المثال السابق يمكن أن نكتب :

```
Shape S = Tringle (2,3,5);
```

فإن المعنى الحقيقي لهذه الكتابة ما يلي :

فعلياً إن S هي مثلث و لكن كل مثلث هو شكل و بالتالي أي غرض من الصف الأب يمكن أن يأخذ أحد أشكال أبنائه .

بذلك تكون قد نكون أنهينا المحاضرة الأخيرة آملين أن نكون قد قدمنا لكم مادةً مفيدة و راجين الله تعالى أن يوفقكم في جميع المواد و أن يوفقنا في تطوير كل ما نقدمه لكم في سبيل تطوير رياضياتنا .

مع تحيات فريق سيريا ماث التطوعي ^_^