البرمجة و الخوارزميات 2

الدكتور: سمير جعفر

المحاضرة : الحادية عشر

التاريخ :١٤/١٢/٢٠١٦

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

البرمجة غرضية التوجه تقوم على ثلاث مبادئ :

١- التجريد (الصفوف والأغراض) ، يعني ذلك أننا نقوم بتمثيل النوع الجديد بشكل مجرد و بعد ذلك نستخدم هذا النوع الجديد من خلال الأغراض(أي المتحولات من هذا الصف)
٢- الوراثة (موضوع محاضرتنا هذه)
٣- تعددية الأشكال .

سنتكلم في محاضرتنا عن الوراثة والتي لها دور هام في تطور البرمجة .

الوراثة:

المبدأ الوراثي : يحاكي نوعاً ما الواقع إذ سيكون لدينا صفاً يرث من صفٍ آخر فيكون الصف المورث بمثابة أب، و الصف الوارث بمثابة ابن.

◄في C++: لنأخذ :

Class A صف ما (أب – قاعدة –مورث)مكون من ثلاث اقسام رئيسية:

عام	محمي	خاص
معطيات اعضاء طرق	معطيات اعضاء طرق	معطيات اعضاء طرق

وراثة من نوع معين
(محدد الوصول) "محمي – عام – خاص"

Class B صف جديد (ابن-مشتق-وارث)

جزء جديد (الابن)			جزء موروث من الأب
عام	محمي	خاص	الجزء المحمي من الأب الجزء العام من الأب

- يسمى الصف الموروث منه بالصف الأب أو قاعدة أو مورث .(الصف الاساسي)
- ويسمى الصف الوارث بالصف الابن أو مشتق أو وارث.(الصف الجديد)
- كل ما هو **عام و محمي** في الصف الأب سينتقل تلقائياً إلى الصف الابن بالإضافة إلى أمور جديدة (محتويات الصف الابن)
- كل شيء خاص لا ينتقل الى الصف الابن .
- نوع الوراثة يحدد محددات الوصول في الجزء الموروث ، بمعنى أن المحمي و العام سينتقلان من الصف الأب إلى الابن ، لكن ليس بالضرورة ما كان محمياً في الأب أن ينتقل على أنه محمي في الابن و كذلك العام ، فالمعلومات المورثة ستصبح من نوع الوراثة (و انظر الملاحظة (*))
- الوراثة ثلاث أنواع (عامة ، خاصة ، محمية).
- نرمز للوراثة بـ : (نقطتين) ونوعها يكون احد محددات الوصول .

ملاحظة (*) : في لغة $c++$ يحق لنا مرة واحدة فقط اختيار نوع الوراثة من الصف الاساسي (عام- خاص - محمي) و أن التعليمات الموروثة يتم التعامل معها ككتلة واحدة فإما كلها خاص أو كلها محمي أو كلها عام .

عندما نقول الوراثة من نوع عام فإننا نعني أن كل ما هو موروث من الأب سيصبح عام في الابن و كذلك الأمر من أجل المحمي و الخاص

نخلص إلى أنه في الوراثة يجب :

أولاً تحديد الصف الابن الذي سيرث من الصف الأب و تحديد الصف الأب .
ثانياً تحديداً نوع الوراثة فبالتالي كل ما سينتقل من الأب إلى الابن سيكون محدد الوصول إليه هو نوع الوراثة.

ملاحظة :سيكون الصف البن هو الصف الأكبر لأنه سيحتوي المعلومات التي سيرثها من الصف الأب بالإضافة إلى معلوماته هو.

كيف نقوم بكتابة وراثة في الكود البرمجي

$$\underbrace{class\ A}_{\text{الصف الأب}}\{\dots\dots\dots\};$$

$$\underbrace{class\ B}_{\text{الصف الابن}}\ :\ \underbrace{public}_{\text{نوع الوراثة}}\quad \underbrace{A}_{\text{الصف الأب}}\quad \{\dots\dots\dots\};$$

$$class\ C : public\ B\{\dots\dots\dots\};$$

و هنا سيكون الصف C على الشكل التالي :

س				خاص بالصف C		
موروث من الأب A	خاص بالصف B			خاص	محمي	عام
A	خاص	محمي	عام			

#تمهيد : برنامج يقوم بجمع عددين ثم جمع عدد ثالث باستخدام الوراثة

```
#include < iostream.h >
class A{
private:
int x;
public:
int y;
A(int x, int y){
this → x = x;
this → y = y;}
int sum( ){
return (x + y);}
};
```

class A يقوم بجمع عددين (x,y)

```
class B : public A{
public :
int z;
B(int z){
this → z = z;}
int sum1( ){
return (sum( ) + z);}
};
int main ( ){
A a = A(2,3);
cout << a.sum( ) << endl;
B b = B(5);
cout << b.sum ( );
return 0;
}
```

class B يقوم بجمع العدد z إلى مجموع (x,y) و هو وارث من A

إن هذا السطر يعني أن الصف B سيرث كل ما هو عام و محمي في الصف A و قد حددنا أن المعلومات الموروثة ستكون في B عامة و ذلك عندما قلنا أن نوع الوراثة public

قمنا ببناء غرض من الصف A و أسميناه a

هنا سيحدث **خطأ** لأن أي غرض من الابن يجب أولاً أن يبنى غرض من الأب ضمنه و إذا لم يكن في باني B تعليمة بناء لـ A فإنه سيستدعي الباني لوحده و لكون لا يوجد باني افتراضي لـ A ستحدث مشكلة

فما الحل !!

١- أن نقوم بكتابة باني خالي ضمن الصف الأب A و من ثم نقوم باستدعاء الباني الخالي الموجود في الأب.

٢- أن نقوم بكتابة باني الصف الابن (داخل الصف) كالتالي :

$$B(int\ x, int\ y\ , int\ z): A(x,y)\{this \rightarrow z = z;\}$$

عندئذٍ في الدالة الرئيسية بدلاً من B b=B(5); سنكتب :

$$B\ b = (3,4,5);$$

ملاحظة: أول تعليمة في باني الأبن هي استدعاء لباني الأب .

ملاحظة: لو قمنا في الرنامج السابق بكتابة التعليمة :

$$cout \ll a.x;$$

فإن ذلك خطأ لأن x نوع محدد الوصول لها هو *private* و لا يمكنني التعامل مباشرة مع أي شيء محدد الوصول له خاص إلا ضمن الصف نفسه .

تمهيد ٢:

```
class T{
private:
int x;
};
int main (){
T t = T();
t.x = 5;
cout << t.x;
```

إن آخر تعليمتين **خاطئتين** لأن x خاصة ولا يمكن الوصول اليها بهذه الطريقة واذا اردنا الوصول اليها فيجب اجراء التغيير التالي :

```
class T{
private :
int x;
public:
void set_x(int x){
this → x = x;}
int get_x( ){
return x;}};
int main(){
T t = T();
t.set_x(5);
cout << t.get_x( );
return 0;}
```

و قد يسأل سائل ، إذا كنا نستطيع الوصول إلى كل ما هو خاص كما في المثال السابق ، فما الفائدة من كونه خاصاً !!!

الفكرة أننا لا نستطيع الوصول إلى ما هو خاص إلا عن طريق الدوال و هذا يمكننا من وضع العديد من الشروط على الوصول و التعامل مع ما هو خاص.

مع تحيات فريق سيريا ماث التطوعي ^_^

"انتهت المحاضرة"

Syria Math