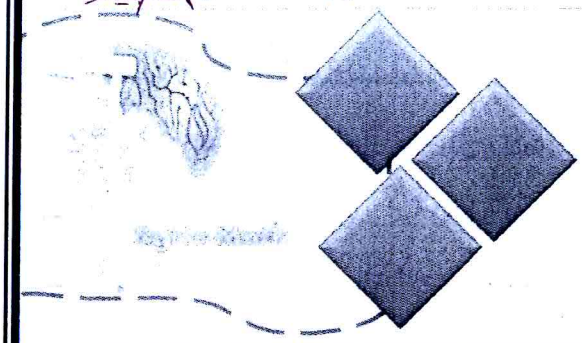
مقرر : البرمجة والخوارزميات المتقدمة السنة الرابعة اختصاص التطبيقية



المحاضرة العاشرة

نظري ☑ عملي ☐

◄ دكتور المادة: سمير جعفر

◄ عنوان المحاضرة: الوراثة

أهلاً بكم أصدقائي في المحاضرة العاشرة من مقرر البرمجة المتقدمة والتي سنتابع فيها بمفهوم الوراثة في لغة جافا ...

مقدمة :

- إن مفهوم الصنف يذهب إلى أبعد من تعريف أنواع جديدة فقط ، حيث تسمح لغة Java بإعادة استخدام أي صنف عن طريق توسيعه ، أي نحصل منه على صنف موسع وبقدرات (عمليات وخصائص) إضافية.
- نسمي عملية إعادة استخدام الصنف عن طريق توسيعه بالوراثة من الصنف (أو اختصاراً الوراثة).
- إذا أردنا لصنف وراثة صنف آخر :

نسمي الصنف الجديد صنف وارث أو صنف مشتق أو صنف ابن .

والصنف القديم صنف موروث أو صنف قاعدة أو صنف أب .

- تستخدم الكلمة المحجوزة extends للتعبير عن وراثة صنف لصنف آخر في لغة Java
- الصنف الابن يرث من الصنف الأب جميع مكوناته (دوال ومتغيرات) العامة والمحمية ، أما الخاصة فهي خاصة بالصنف الأب ولا يرثها الصنف الابن .
- توضيح: ليكن A صنفاً «الأب» :

Private	Public	Protected

A (صنف)

↑ (يرث)

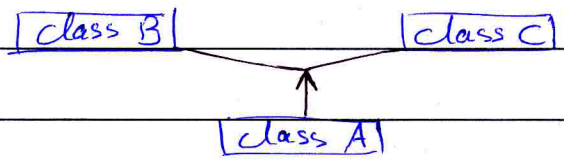
المكونات الموروثة

B (صنف ابن)

الجزء الموروث		الجزء الجديد (جزء ابن)		
Protected	Public	Private	Public	Protected

- بالنسبة للجزء الخاص بالابن فنحن نعرفه ونكتبه، أما الجزء الموروث من الأب (أي المكونات العامة والمحمية) فيستقل بشكل تلقائي عند إنشاء علاقة الوراثة.
- المكونات الخاصة private الموجودة في الأب لا ترث للابن فهي خاصة بالأب.
- لغة جافا لا تسمح بالوراثة المتعددة، أي أنه لا يسمح للصف الواحد أن يرث أكثر من أب واحد مباشر.

مثلاً: التوريث التالي خاطئ، لأن الصف A له أبوان ((يرث من أبوين)) مباشرة B و C



- لكن على الصف الواحد أن يمتلك عدداً من الأبناء ((أي يمكن أن يكون هناك أكثر من صف يرث الصنف ذاته)).
- إذاً الوراثة تمكننا من تعديل وإيقوم به الصف الأب دون أن نخسر كود الأب - فنقوم بتوسيع الصف الأب ((بإضافة متغيرات ودوال)) لنقوم بعمل إضافي لعمله الأساسي.
- يقوم الصنف الابن بتعديل التعليمات البرمجية للدوال الموروثة من الصف الأب، وندعو ذلك بإعادة تعريف للدوال (أو الطرق).

والقدرة على إعادة تعريف طرق الصف الأب في الصف الابن واحدة من الإمكانيات الكبيرة التي توفرها البرمجة غرضية التوجه.

- يكفي التصريح عن صف جديد (ابن) يرث من الصف الأب، ثم يقوم الصف الابن بإعادة كتابة التعليمات البرمجية لبعض أو جميع الطرق التي ورثها بحيث يحتفظ الصف الأب بالطرق الأصلية. ويمتلك الصف الابن طرقاً موروثة ومعدلة.
- نه نستطيع القول أن إعادة تعريف الدوال: هي كتابة دوال ضمن الصف الابن بحيث يكون لها نفس الاسم ونفس قائمة الوسطاء، ولكن تقوم بعمل مشابه لعمل دالة الأب أو مختلف عنه.

مثال: ((في مثالنا الصف الأب A والصف B يرث أي

))

```
public class A{
int i=0;
void something(int k){
    i=k;}
}
class B extends A{
 int j=0;
void something (int k){
    j=10;
   i= 2*k;}
}
```

- لاحظ كيف أعاد الصنف الابن B تعريف الطريقة الموروثة something من الأب A.
- عند تعريف غرض من الصنف B واستدعاء الدالة something على هذا الغرض يتم تنفيذ تعليمات الدالة الخاصة بالصنف B ((وليس تنفيذ تعليمات دالة A))
- عند استدعاء دالة غير موجودة في الصنف الابن ((أي موروثة من الصنف الأب)) فإنه يقوم باستدعاء الدالة من الصنف الأب.

- حتى نتمكن من إنشاء غرض من الصنف الابن يوجب علينا هنا إنشاء غرض من الصنف الأب. ومنه يجب على باني الصنف B استدعاء باني الصنف A، لذلك يجب أن تكون أول تعليمة في الصنف الابن استدعاء لباني الصنف الأب، وإذا لم تكن كذلك ((إذا لم نكتب صراحةً أو لم يوجد باني في الصنف الابن مثلاً)) فإن لغة جافا سوف تستدعي الباني الافتراضي أو الخالي للصنف الأب، لذلك يجب كتابة باني خالي بأي خالي في الصنف الأب هنا لأنه في حال لم نزع باني الأب استدعى في باني الابن ولم يكن هناك باني خالي في الأب فإن هذا سيولد خطأ.

مثال:

```
class A{
public int x;
protected int y;
private double d;
A(int x, int y, double d){          (1)
    this.x = x;                     (2)
    this.y = y;
    this.d = d; }
public void print(){
System.out.print(x + "," + y + "," + d); }
}

class B extends A{                  (3)
private char c;
(*) { B(char c){ this.c = c; }      (4)
    void print(){ System.out.print(x + "," + y);   (5)
    System.out.print(c); }
}
```

(1) باني الصنف A له اسم الصنف ووسطاؤه (x, y, d).

(2) this هو اسم آخر للصنف داخل الصنف نفسه

(نستخدمه لتجنب اختيار أسماء كثيرة)

ويعني أن (this.x) هو المتحول x الخاص بالصنف، و (x) هو الوسيط

(3) صنف B ورث من الصنف A.

(4) باني الصف الابن وسيطه (c).

(5) دالة print() نوع إرجاعها فارغ، تقوم بطباعة قيمة x وحرف (و) وقيمة y ثم تطبع قيمة c.

لكن هنا لم نستدع باني الصنف الأب في الصنف الابن، لذلك سوف نستدعي اللغة باني الأب الافتراضي، لكن نلاحظ هنا أنه في الصنف الأب لا يوجد باني افتراضي (لأننا قمنا بكتابة باني له) كما أننا لم نكتب باني خالي أيضاً، وهنا سيولد خطأ.

لتجنب هذه المشكلة يجب استدعاء باني الأب في باني الابن كما يلي:

نغير كتابة الجزء (*) ونستبدله بالتالي:

```
B(char c){
  super(0,0,0); ((<-- استدعاء باني الأب))
  this.c = c; }
void print(){
  System.out.print(x + "," + y);
  System.out.print(c); }
}
```

ملاحظة: super هو اسم الصنف الأب في الصنف الابن، فعند أي استدعاء للصنف الأب في الصنف الابن نستخدم كلمة super وليس اسم الصنف الأب، ولو استخدمنا اسم الصنف الأب سيعطي خطأ.

تمرين: (سؤال دورة)

ليكن لدينا الملف A.java والذي يحتوي البرنامج التالي:

```
class A{
  private int x,y;
  public A(int x, int y){
      this.x = x;
      this.y = y; }
}
```

```
public class B extends A {
    int z;
  public B(int z) {
    this.z = z; }
 int alt_odd() {
    int sum = x + y + z;
    return sum; }
                }
```

عند ترجمة الملف السابق نحصل على أربعة أخطاء، حددها مع الشرح.

الحل:

1- اسم الملف: يجب أن يكون B.java لأننا نسمي الملف باسم الصف العام فيه و B هو الصف العام في هذا الملف.

2- عدم استدعاء بانى الأب في أول تعليمة من بانى الابن: لم نقم باستدعاء مرجع لبانى الأب، وبالتالي سيتم استدعاء بانى خالي من الأب، وبسبب عدم وجوده في الأب سيعطي خطأ.

3- x عضو بياني خاص في الصف A، لا يمكن الوصول إليه خارج الصف.

4- y عضو بياني خاص في الصف A، لا يمكن الوصول إليه خارج الصف.

(في السطر قبل الأخير من البرنامج).

إلى اللقاء في محاضرة قادمة...

انتهت المحاضرة.