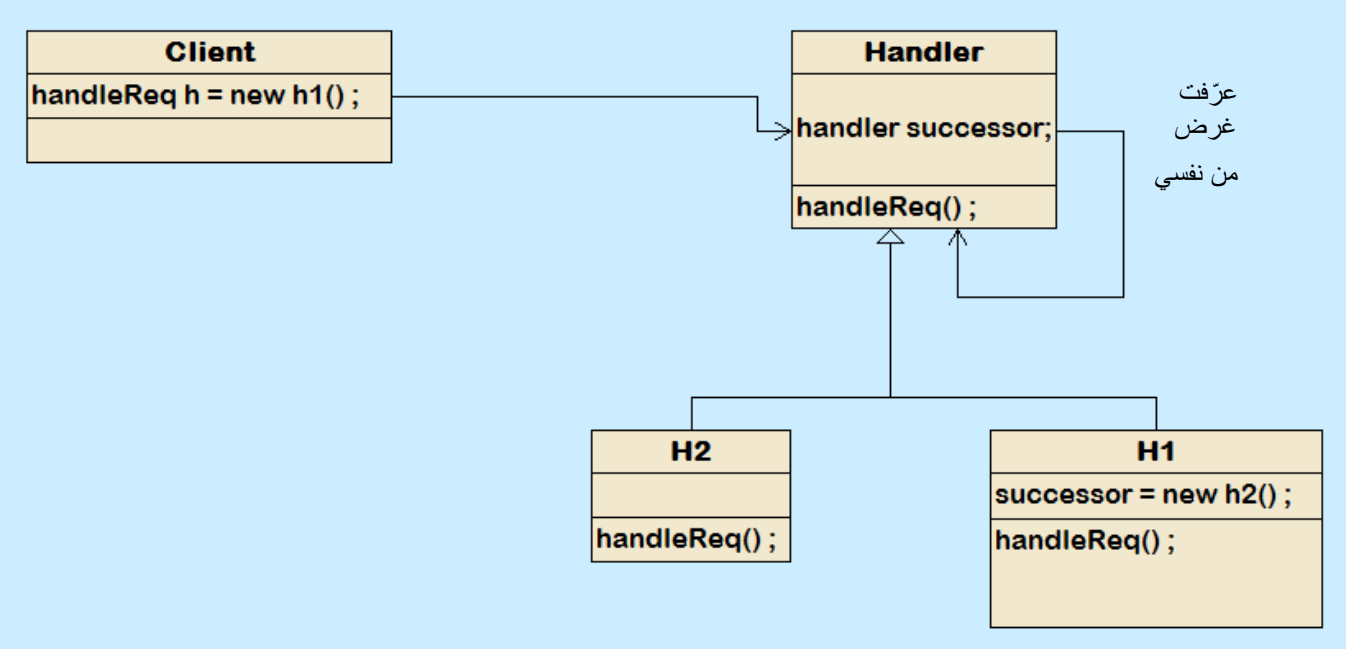
كما اتفقنا في المحاضرة السابقة أن ال Design pattern تساعد في تسريع إيجاد الحل المثالي وليس إيجاد الحل حيث أن الحل يختلف حسب المشكلة ؛ و ال Design Pattern هو ديناميكي أي يقبل الإضافة أو التغيير ..

نكمل أنواع ال Design Pattern :

○ Chain Of Responsibility :

نوعه : Behavioral .

المشكلة : لدي مهمة معينة قد يقتصر تنفيذها على كلاس واحد أو أكثر فنوكل المهمة إلى كلاس من أحد الأبناء(h1) وإن كان هناك حاجة لتابع ما من كلاس ابن اخر يرث من Handler ايضاً, فوجود الوراثة و ال association الذاتي في الأب Handler يتيح له أن يستدعي تابع من هذا الكلاس.



✓ وهي ديناميكية من أجل إمكانية إضافة كلاس جديد (طريقة أخرى) .

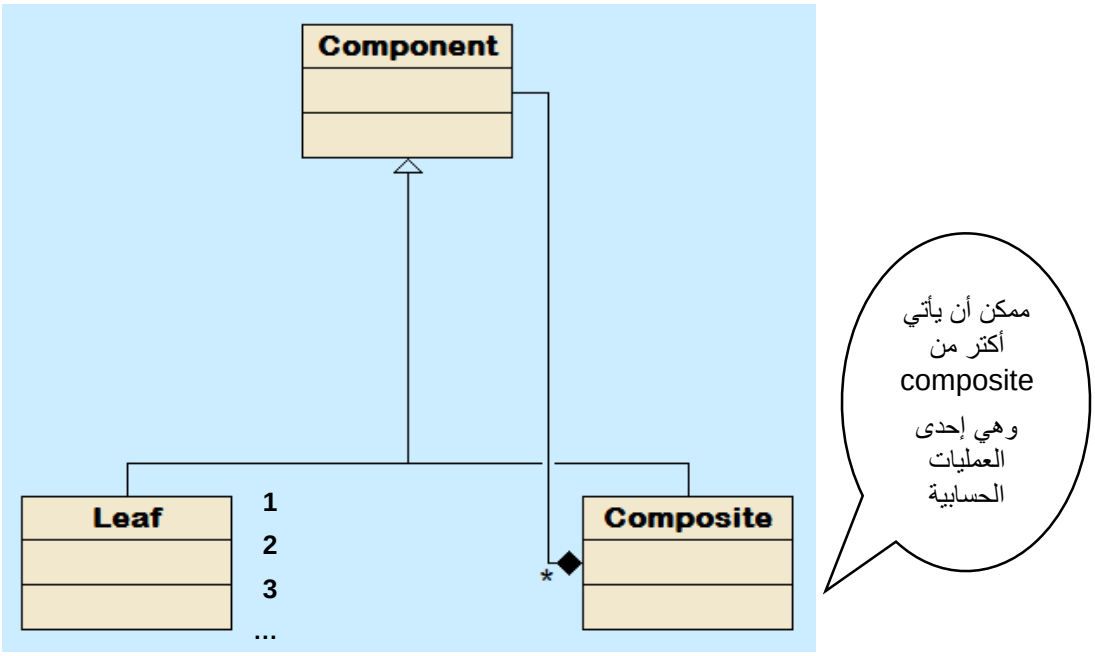
```
Class h1 extends Handler
{
  HandleReq(int x)
    { if ( x==1 )
       \\ مجموعة تعليمات
      Else
       Successor. HandleReq(x)
     }
}
```

نلاحظ وجود تسلسل بالمسؤوليات فمن خلال المثال السابق نجد أن client يعتقد أن h1 هو الذي قام بالمهمة ؛ و لكن من الممكن أن يكون في التابع HandleReq الموجود في h1 استدعاء من الغرض Successor على أي Handler من الأبناء ليقوم بالمهمة .

○ Composite :

نوعه : Structural

تستخدم من أجل أي مشكلة يوجد فيها تراكب مثلا عبارة رياضية هي عبارة عن عملية ورقم (leaf) و هكذا حتى نصل إلى كل الأوراق ؛ وأيضا مجلدات النظام كل مجلد فيه عدة ملفات أو مجلدات .

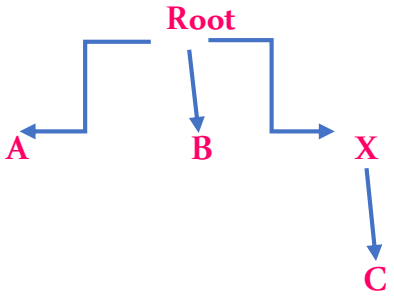


```
class Composite extends Component
{
ArrayList comp = new ArrayList < Component > () ;
Void AddComponent()
{
Comp.add( c) ;
}
```

✓ طبعا يجب الانتباه الى تعريف تابع add في ال Component .

```
Void main()
{
 Composite Root= new Composite ('root') ;
Root.add(new leaf ('A')) ;
Root.add(new leaf ('B')) ;
Composite x = new Composite ('x') ;
x.add(new leaf('c'));
Root.add(x) ;
Root.print() ;
}
```

- **رسمة توضيحية :**



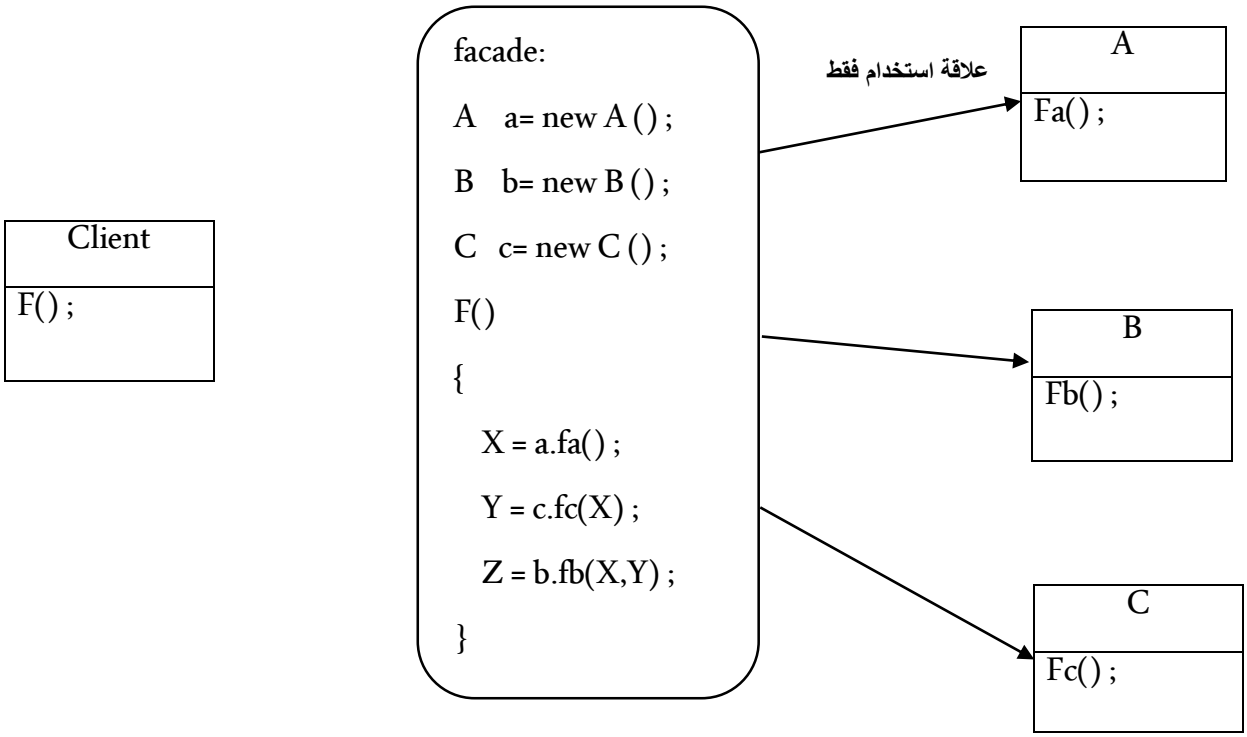
✓ **ملاحظة :**

يمكن الدمج بين أكثر من Design Pattern .

○ Facade :

نوعه : structural.

لدينا عدة توابع من عدة صفوف وبدلاً من التواصل مع جميع الصفوف نتواصل مع صف واحد يرتبط مع هذه الصفوف .



عندما نحتاج إجراء تغيير ترتيب الأحداث أي ترتيب التعليمات للتابع f نغير فقط في الصف facade .

مثلا : في مؤسسة ما يجب أن نوقع أوراق في قسم المحاسبة A ثم الديوان B ثم المدير C ؛ في لحظة ما عند تغيير هذا السيناريو أغير في التابع F فقط وذلك يوفر سهولة وسرعة وديناميكية بالإضافة أنه من الممكن أن يستخدم صف اخر هذا التابع نفسه لذلك لم نضع التابع f () ضمن client .

انتهت المحاضرة

Written by :
Shorouq Abu Hasan
Word press and preparation :
Afaf AlAwam
Reviewed by :
Walaa Jaweesh