جامعة دمشق – كلية الهندسة المعلوماتية

السنة الخامسة – قسم البرمجيات

هندسة برمجيات (3)

Design Patterns

المهندس المشرف : روان قرعوني

إعداد الطلاب :

حنين فخر الدين الشعراني

وليم الندفة

مرح جلبوط

أيهم المصري

الطلب الأول :

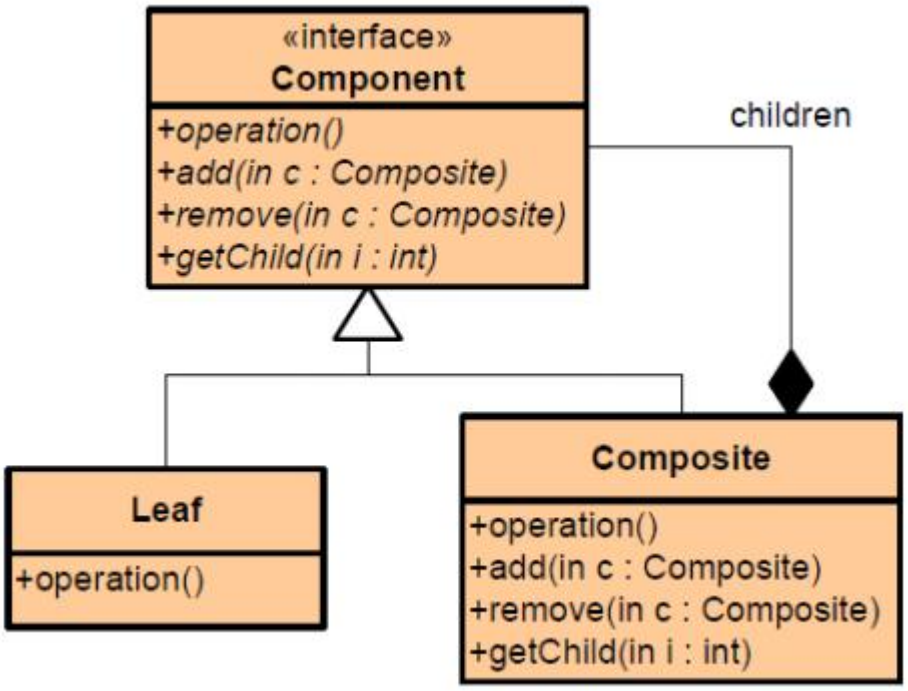
التصميم المستخدم : Composite

نوعه: Structural

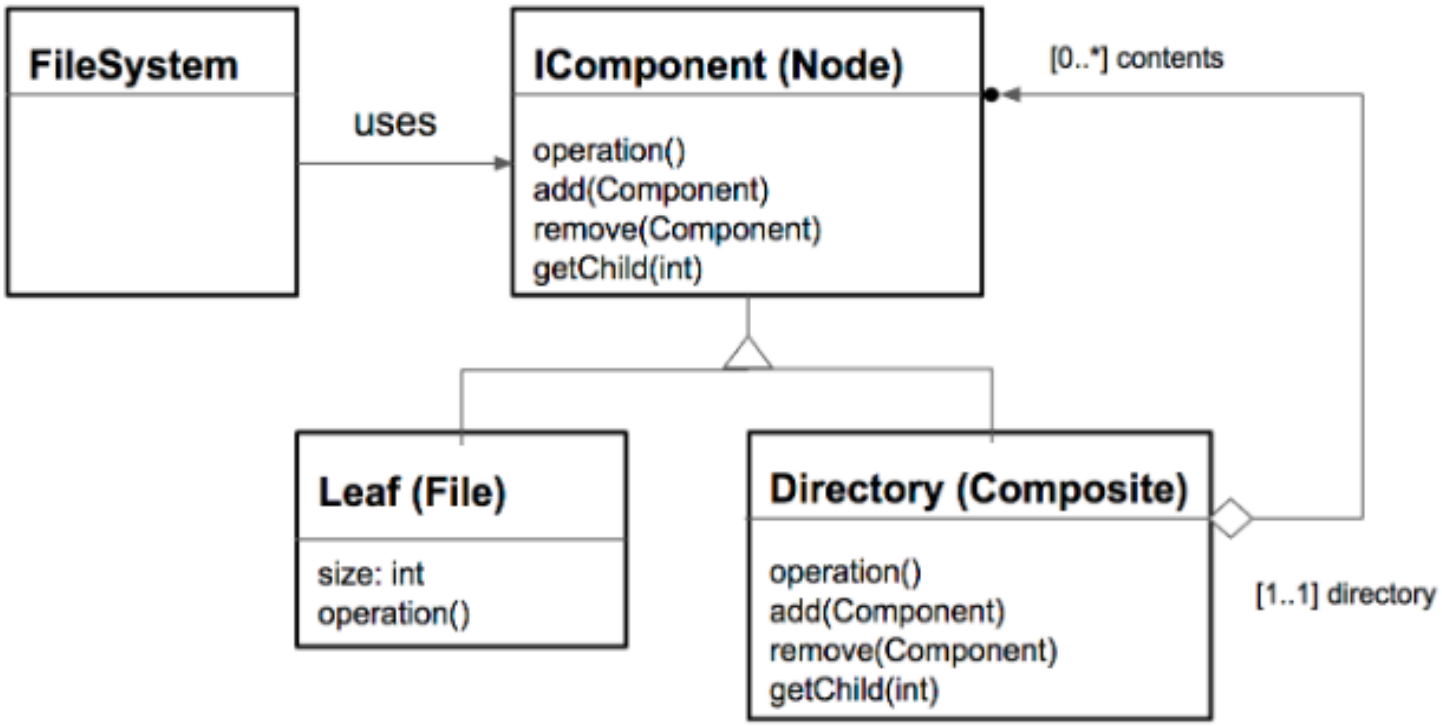
المشكلة : لكل مريض يوجد مجلد في النظام و كل مجلد مكون من مجموعة من المجلدات أو الملفات الأخرى المرتبطة مما أدى للتراكب بين المستويات .

يسمح لنا هذا الصميم بتعريف العملية الخاصة بالملف و ضمن المجلد نعرف العمليات الأساسية الأخرى و يمكننا من بناء بنية هيكلية تتيح لنا معاملة كل جزء من الأجزاء المختلفة بشكل متماثل .

الشكل العام لنموذج Composite:



مخطط الصفوف الموافق للمطلوب :



كود توضيحي :

```
class Directory extends IComponent{

ArrayList comp = new ArrayList < IComponent > () ;

Void add(Component)

{

Comp.add( Component) ;

} ……

 Void main(){

Directory Root= new Directory () ;

Root.add(new  File) ;}}
```

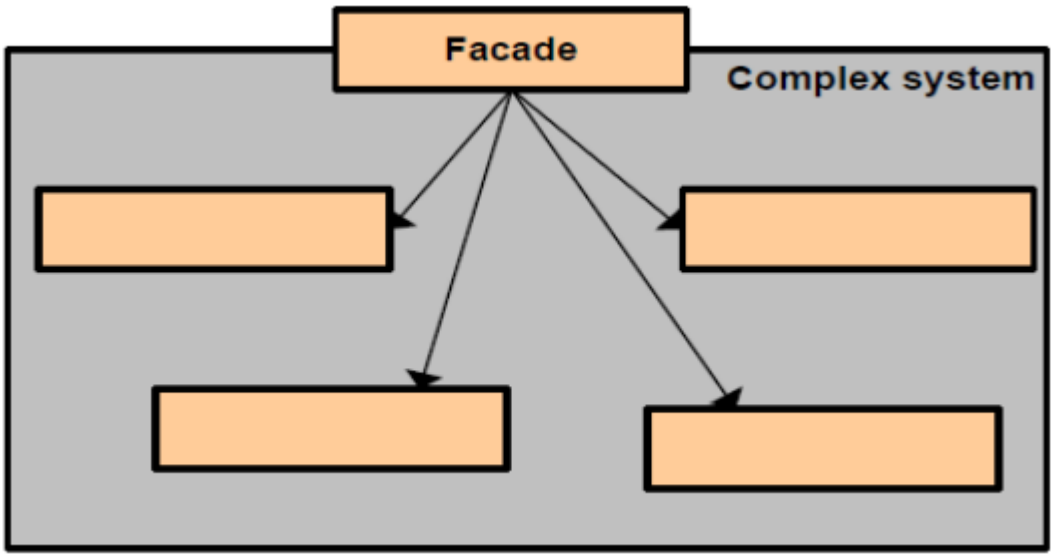
الطلب الثاني :

التصميم المستخدم :Façade

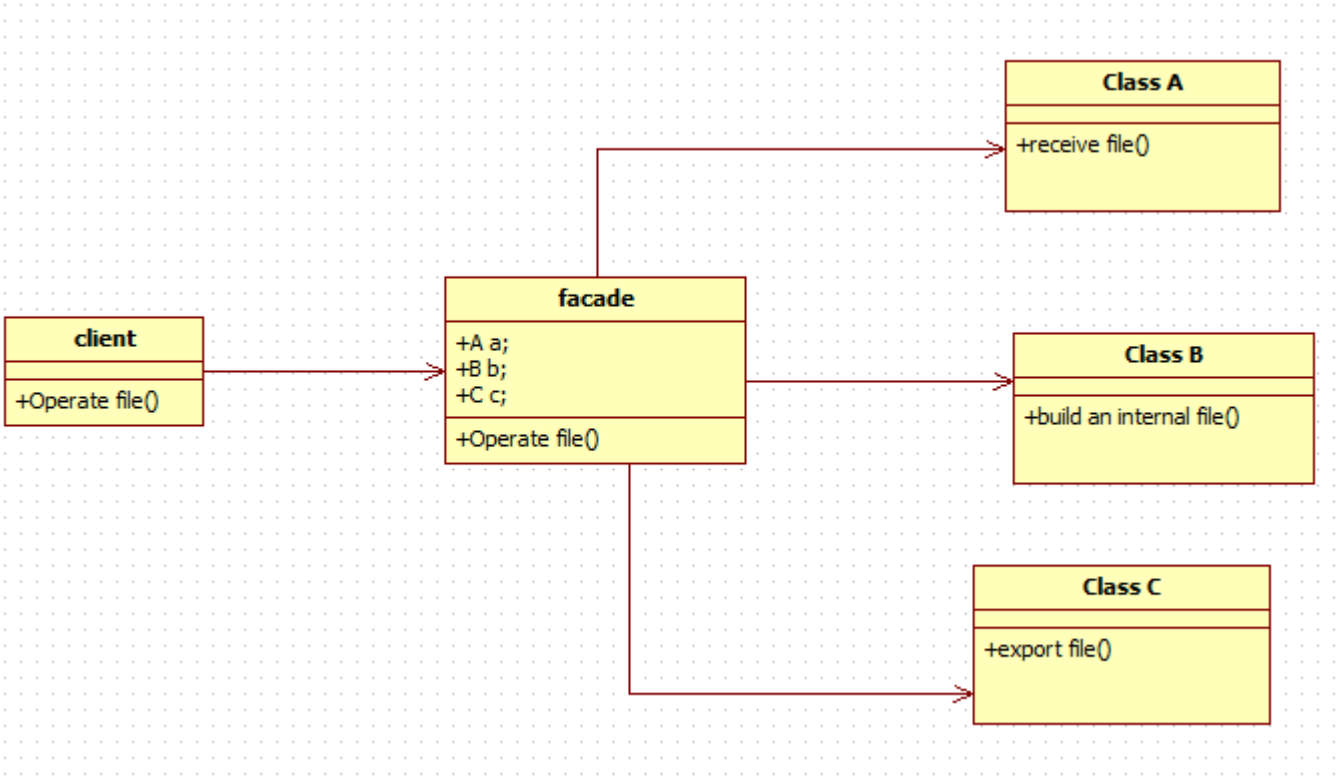
نوعه : Structural

المشكلة : نستخدم هذا النموذج لتعريف واجهة موحدة لمجموعة من الواجهات في النظام ، أي أنه يشكل واجهة عالية المستوى للتخاطب مع النظام بشكل أسهل و هكذا يستطيع الموظف القيام بالعمليات المختلفة على ملف المريض من استقبال و بناء ملف داخلي و تصدير للخارج من خلال أمر محدد في النظام دون التقليل من الإمكانيات المتاحة و دون التعامل م تفاصيل المكونات و تعقيدات النظام.

مخطط الصفوف العام لنموذج Façade:



مخطط الصفوف الموافق لمعالجة ملف المريض :



كود توضيحي:

```
Class A{

Public void receive file();}

Class B{ Public void build an internal file();}

Class C{ Public void export file();}

Class Façade{

A a ;  B b;  C c;

Public void Operate file(){

a.receive file();

b.build an internal file();

c.export file();

}}
```

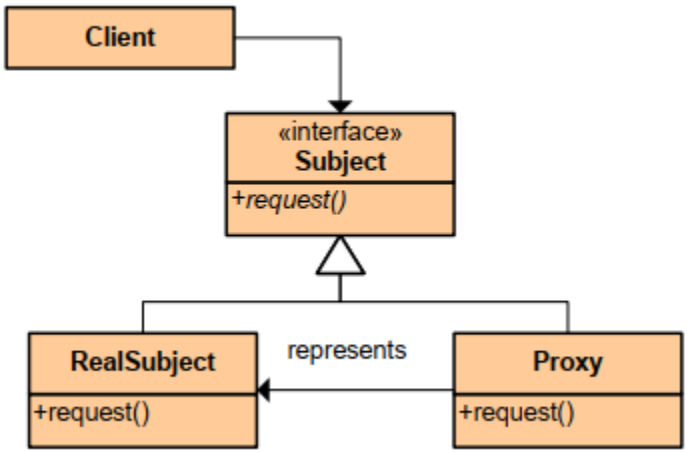
الطلب الثالث :

التصميم المستخدم : Proxy

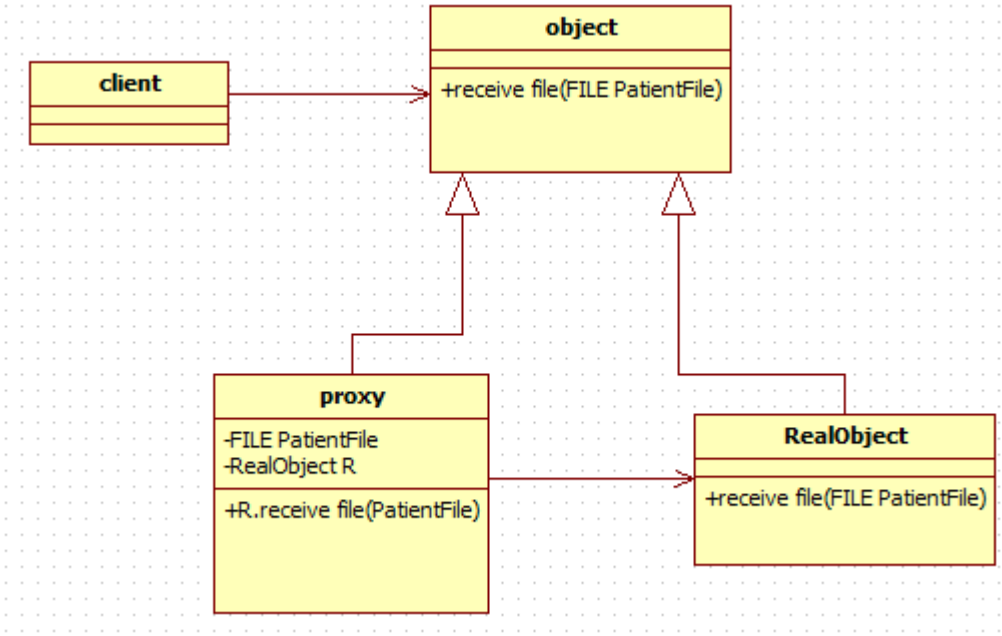
نوعه : Structural

المشكلة : نريد منع موظفو المشفى(عدا موظفو الديوان) من القيام باستقبال ملف قبل التحقق من صلاحياته أي يوجد شروط معينة يجب تحقيقها قبل تنفيذ التابع المطلوب(استقبال الملف) و هذا التصميم يتيح سماحيات مختلفة للمستخدم .

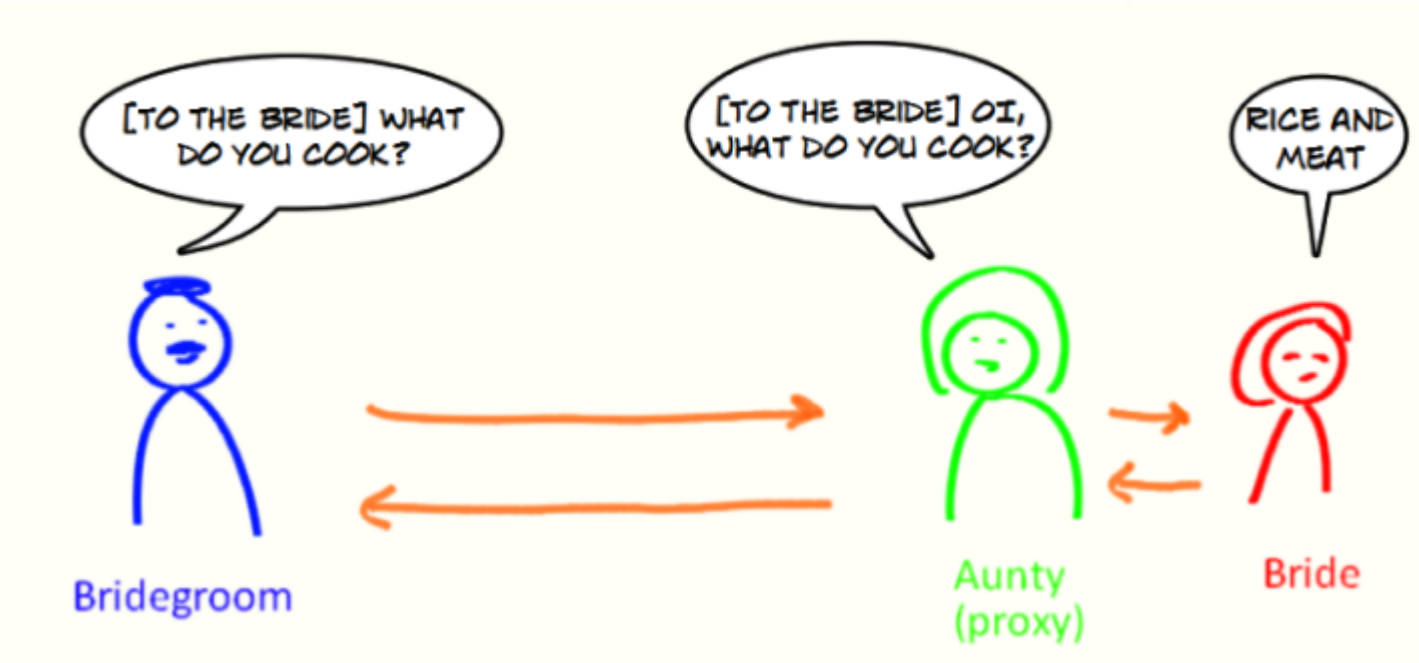
مخطط الصفوف العام لنموذج Proxy:



مخطط الصفوف الموافق للمطلوب:



(;



کود توضیحی:

```
public interface object {
 public void receive file(FILE PatientFile);
}
public class Proxy implements object {
private FILE PatientFile;
private RealObject R = new RealObject();
@override
Public void receive file(FILE PatientFile){
R.receive file(PatientFile);
}}
Public class RealObject implements object{
@override
Public void receive file(FILE PatientFile){
receive file(PatientFile);}
}
```

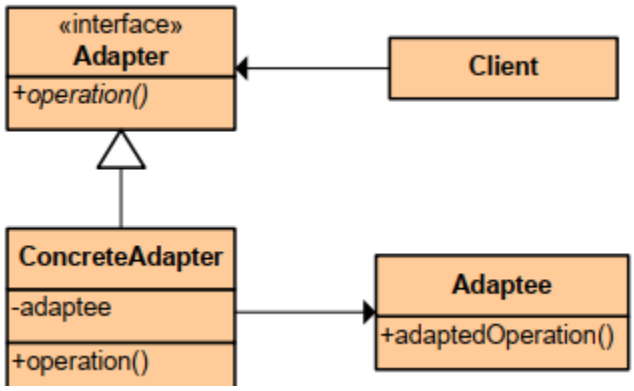
الطلب الرابع:

التصميم المستخدم : Adapter

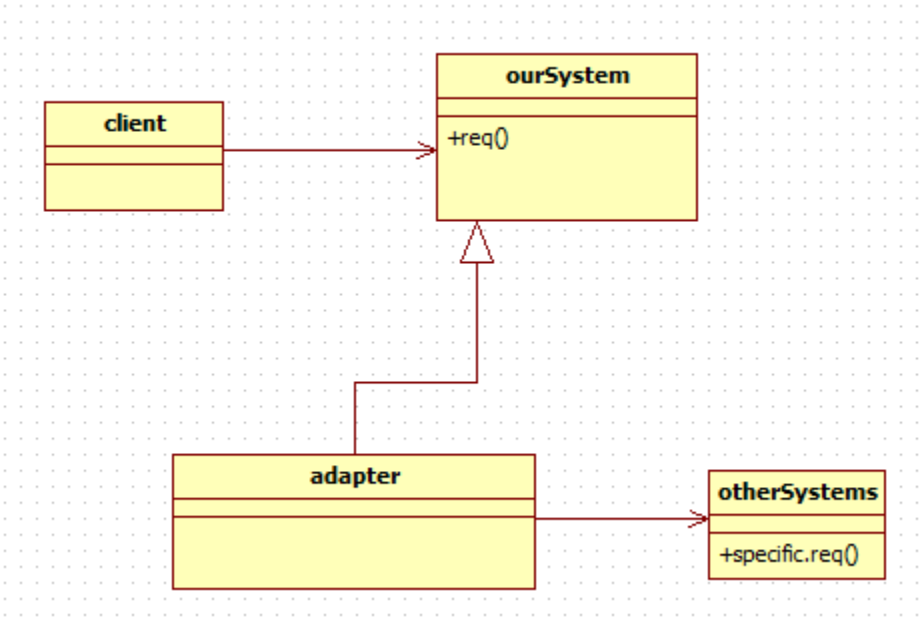
نوعه: Structural

المشكلة : الأنظمة التي يتعامل معها نظامنا تحوي برامج خاصة لا نستطيع تعديلها و هذا النموذج يقوم بتحقيق التوافق بينهما لذلك قمنا باختياره.

مخطط الصفوف العام لنموذج Adapter:



مخطط الصفوف الموافق للمطلوب:



کود توضیحی:

```
Class client {
ourSystem O = new ourSystem();
void F(){
O.req();
}}
Class adapter extends ourSystem{
otherSystems S = new otherSystems();
req();
S.specific.req();
}
```

الطلب الخامس :

التصميم المستخدم : Template

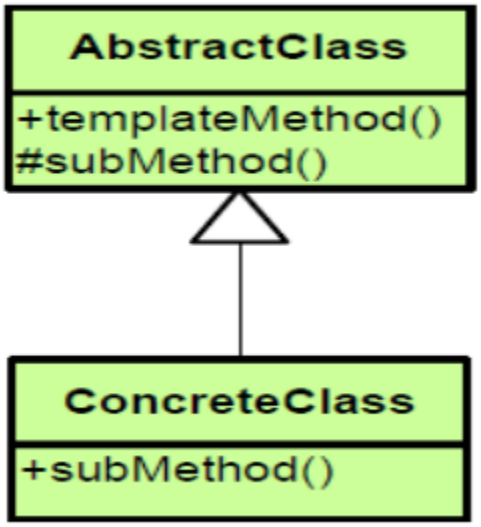
نوعه : Behavioral

المشكلة : يوجد أكثر من خطوة لأرشفة ملف ولكن أحد الخطوات تتغير حسب نوع الملف (وارد / صادر / داخلي) .

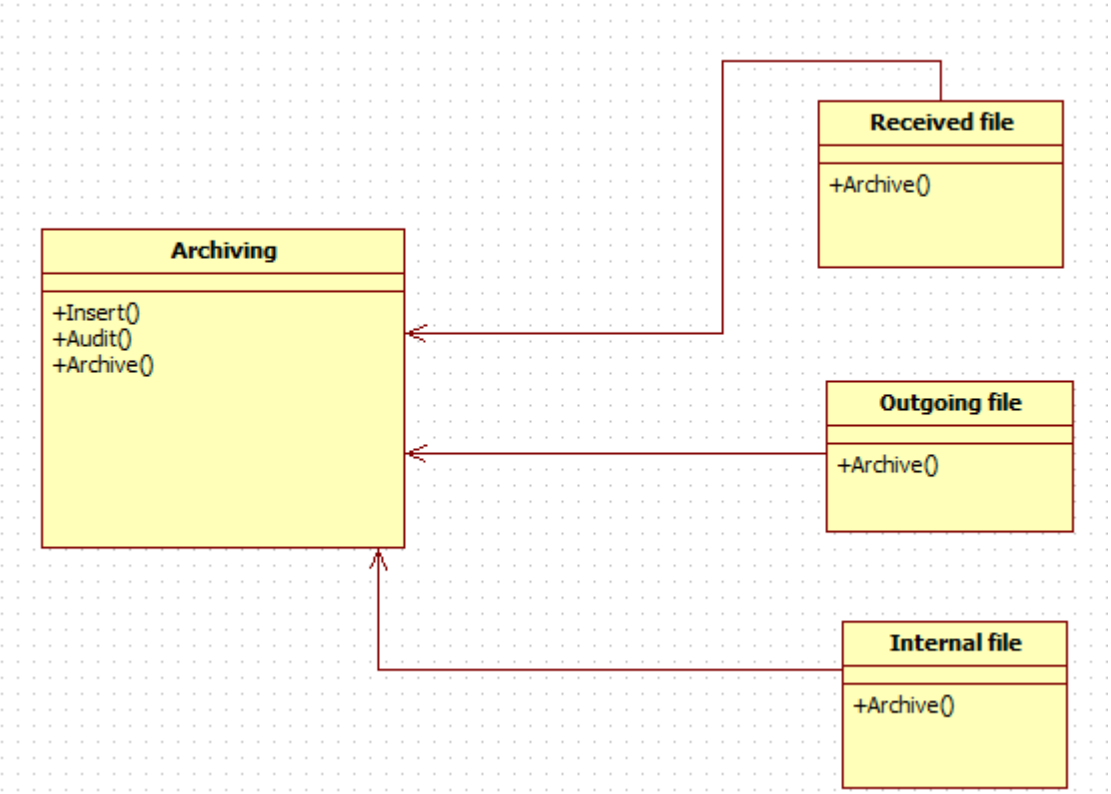
نموذج (Template) هو النموذج الذي نستخدمه عند وجود خوارزمية تتألف من مجموعة من الخطوات

و التي تستخدمها و بنفس الترتيب مجموعة من الصفوف حيث يكون عدد الخطوات مشتركة بين جميع الصفوف بينما الخطوات المتبقية لكل صف سلوكه الخاص .

مخطط الصفوف العام لنموذج Template:



مخطط الصفوف الموافق لخطوات أرشفة ملف :



كود توضيحي:

```
Public abstract class Archiving{

Abstract void Insert();

Abstract void Audit();

Public final  void Archive(){}}

Public class Received file{

@override

Void Archive()}

Public class Outgoing file{

@override
```

Void Archive()}

Public class Internal file{

@override

Void Archive()}

The End ☺