النمذجة ودورها في هندسة البرمجيات

تعاريف:

النموذج: عبارة عن تمثيل مجرد لنظام من وجهة نظر معينة.

النظام: هو مجموعة من المكونات المتفاعلة مع بعضها التي تعمل بالتكامل والتناسق لبلوغ هدف معين.

نظام المعلومات: هو نظام ما يميزه أن الدخل يتم إجراء عمليات عليه من معالجة وتخزين معلومات للحصول على الخرج.

* النظام ليس كتلة واحدة وإنما مجموعة من الأنظمة الجزيئة التي تتفاعل مع بعضها البعض.
* لا يوجد نموذج واحد قادر على تمثيل النظام بشكل كامل وإنما كل نموذج عبارة عن تمثيل من وجهة نظر معينة لتبسيط الواقع فيجب أن يكون بسيط نوعاً ما لنتمكن من فهمه ونحن بحاجة مجموعة من النماذج والمخططات لفهم النظام وكل نموذج يركز على جانب معين.

الخدمات التي تقدمها النمذجة:

1. تبسيط الواقع: وهي أهم خدمة.
2. التوثيق: إحدى طرق التوصيف عبر النماذج.
3. التواصل: يساهم في عملية التواصل بين أعضاء الفريق وبين الفريق المطور والزبائن.
4. يمكن من النموذج الحصول على كود: عن طريق Model driven Engineering (MDE) تعتمد على رسم المخططات وتحويلها إلى كود بشكل مباشر وبالتالي أصبحت النماذج أساس في عملية التطوير.

عبر التاريخ لدينا نوعان من النماذج:

1. المنهجية المهيكلة.
2. المنهجية غرضية التوجه.

المنهجية المهيكلة

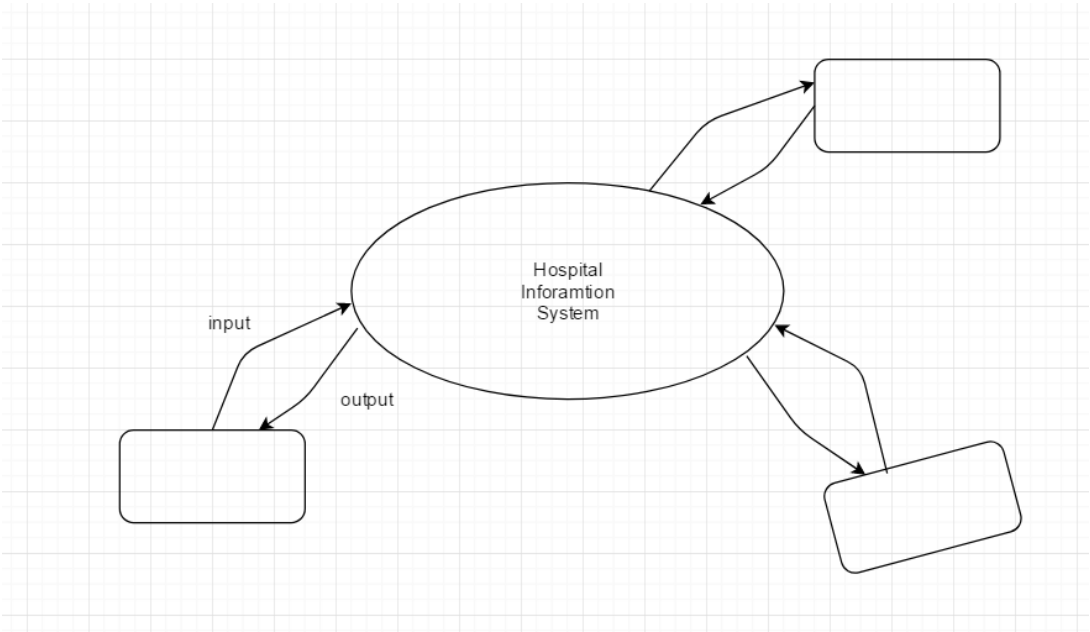
تطوير النظام على شكل توابع وكانت سائدة في السبعينيات قبل ظهور البرمجة غرضية التوجه، في اللغات الإجرائية هناك فصل بين التوابع وال data وكانت المنهجية المستخدمة في النمذجة للإجرائيات (function) عن طريق مخطط Data Flow Diagram DFD أما ال dataعن طريق مخطط ERD وهو مخطط قاعدة البيانات Entity Relation Diagram .

DFD

مخطط دفق البيانات: هو مخطط يتم رسمه على مستويات نبدأ ب DFD level 0 والذي يسمى context Diagram مخطط السياق ويمكن استخدامه لنمذجة scope of System مجال أو نطاق النظام وهو حدود النظام من حيث الخدمات فيحدد ما الذي سيشمله النظام من خدمات والذي سيكون خارجه.

تحديد الخدمات يتم في مرحلة التخطيط وغالباً ما نصل لما يسمى انفجار المدى بسبب كون الزبون متطلب ودائماً يطلب المزيد من الخدمات فنقوم بتقييده (تجميده) بلحظة معينة لكي نستطيع تخمين الوقت اللازم لتطوير النظام والكلفة ... إلخ.

يتم تمثيل المخطط برمزين أساسيين هما الدائرة تحوي اسم النظام ومستطيلات تحوي الكيانات الخارجية وهي الأشخاص والأنظمة التي تتفاعل مع النظام وكلاهما يقدم دخل للنظام وقد يستقبل خرج من النظام.



يأتي بعدها level 1,2,.. وهنا لا نتطرق للكيانات وإنما نمثل الدخل كيف ستتم معالجته وانتقاله وتخزينه واسترجاعه وكل مستوى يعطي تفاصيل أكثر.

مشكلة الDFD أنه يمثل كل التوابع مع بعضها وكل levelيزيد التعقيد.

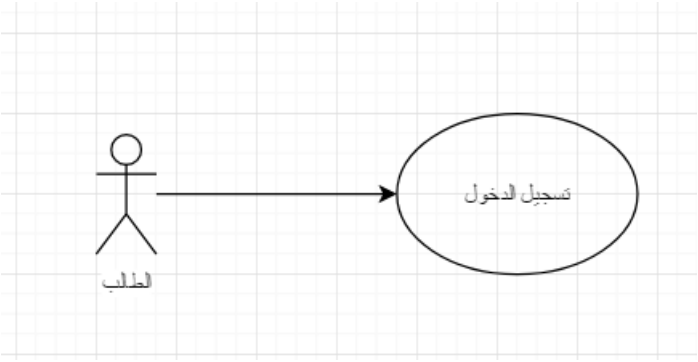
النمذجة في المنهجية غرضية التوجه

النظام عبارة عن مجموعة من الأغراض المتفاعلة مع بعضها تتكامل لتقدم خدمات منذ ظهور هذه المنهجية ظهرت UML (Unified Modeling Language) وهي لغة معيارية لدعم التطوير غرضي التوجه، الهدف من ظهورها هو تقديم طريقة معيارية متفق عليها ومجموعة من المخططات الكاملة عبر جميع مراحل التطوير من تحليل وتصميم وتكويد ...

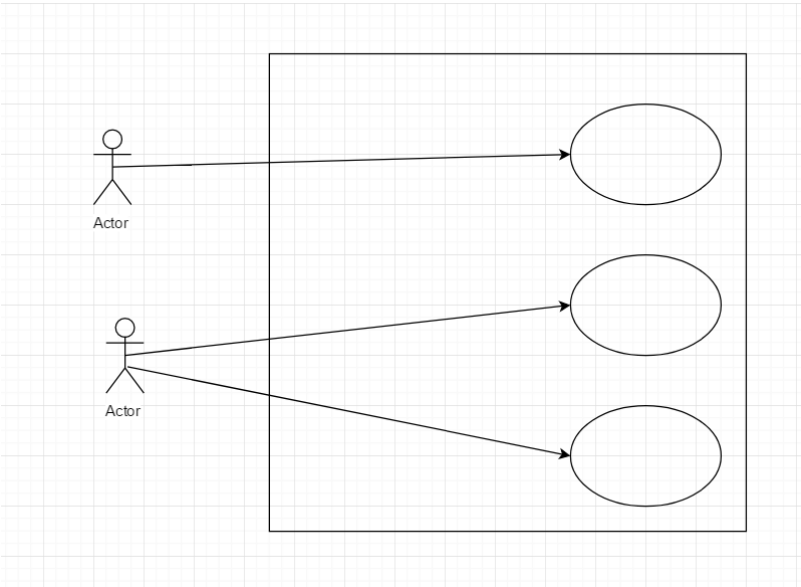
مخططات ال UML

Use Case Diagram: مخطط تحليلي يلخص (ينمذج) مرحلة التحليل بشكل كامل فقط من حيث المتطلبات الوظيفية أما غير الوظيفية فيتم الحديث عنها في الوثائق، أي يقوم بنمذجة مرئية للمتطلبات الوظيفية.

مثال: النظام يسمح للطالب بتسجيل الدخول



شكل ال UCD:



بشكل عام نبدأ متطلب تلو الآخر وفي النهاية نرسم مخطط حالات الاستخدام يتكون من الرموز المعيارية التالية:

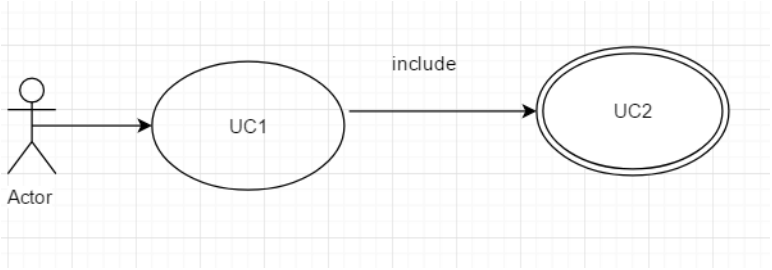
- مستطيل يمثل حدود النظام.

- جميع الactors خارج حدود النظام.
- Use Case بشكل بيضوي داخل حدود النظام.
- أسهم لتمثيل العلاقات.
- العلاقات.

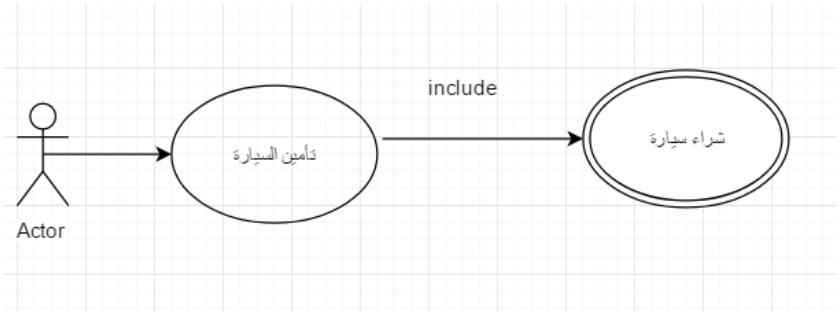
❖ **العلاقات**: لدينا نوعان من العلاقات

1. علاقات بين Actors وهي generation.
2. علاقات داخلية تصل بين ال Use Caseولها 3 أنواع:

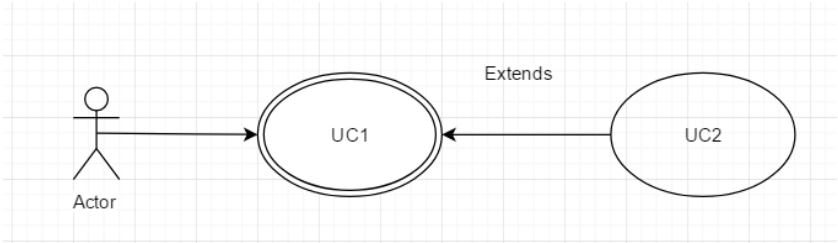
✓ **Include**: UC1تحتوي UC2 عندما تطلب الUC1 فانه لا يمكن إنجاز هذه الخدمة قبل إنجاز UC2



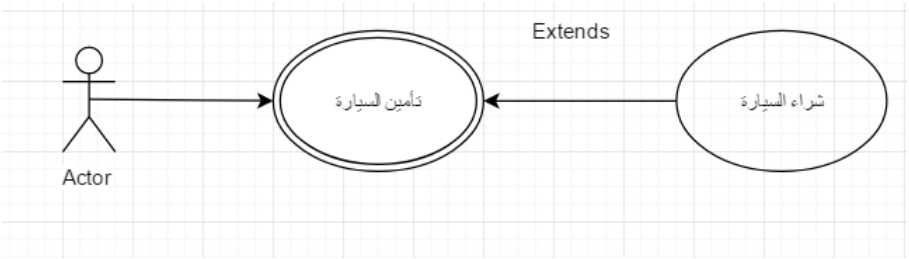
مثال : تأمين سيارة لا يمكن أن يتم بدون أن يكون لدي سيارة



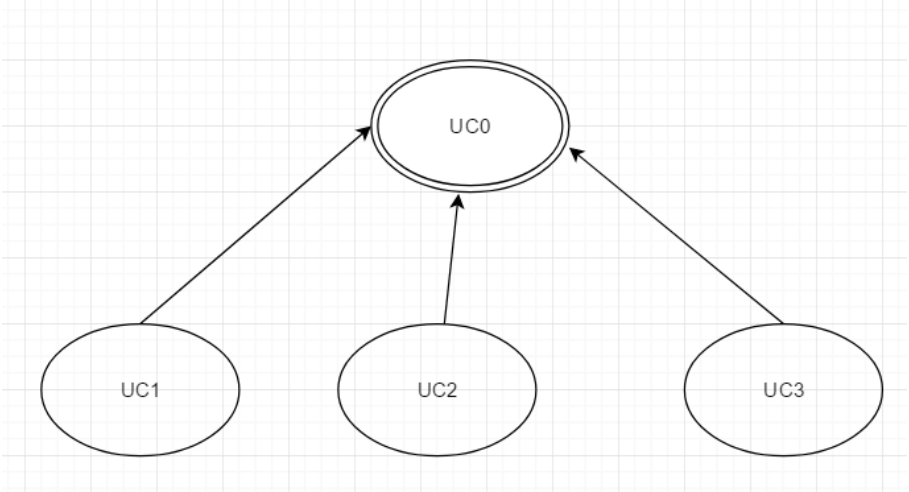
✓ **Extend**: عندما يُطلب UC1 من قبل Actor أقوم ب UC2 بنجاح وبعدها قد أذهب إلى UC1وقد لا أذهب ولكن يجب أن أقوم ب UC2 بنجاح لأستطيع الذهاب ب UC1.



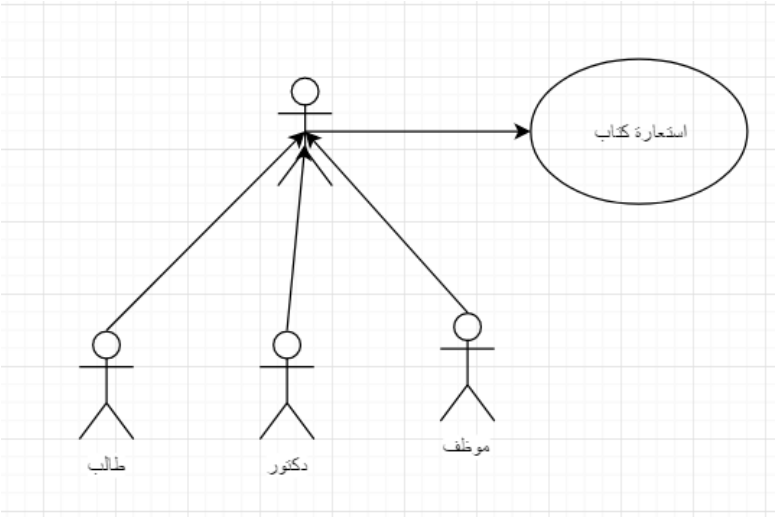
مثلاً: أقوم بشراء السيارة وقد أقوم بعمل تأمين لها أولاً



الوراثة Generalization: عدة حالات استخدم تشترك في صفات معينة نقوم بجمعها تحت حالة استخدام مجردة.



مثال: نظام المكتبة:



كل حالة استخدام تحتاج documentation كاملة وتستخدم هذه العلاقات لتبسيط المخططات وتجنب التكرار في ال documentations

ما هو الفاعل الثانوي؟

هو كيان خارجي يوجد خارج حدود النظام لكنه ليس طالب للخدمة يساعد الفاعل الأولي للحصول على الخدمة.

مثال: في المكتبة يوجد موظف لمساعدة الطالب على استعارة الكتاب ولكنه لا يهتم بحصول الخدمة أو لا لكنه يساعد في إتمامها.

ليس بالضرورة وجود فاعل ثانوي لكن وجود فاعل ثانوي يعني وجود فاعل أولي.

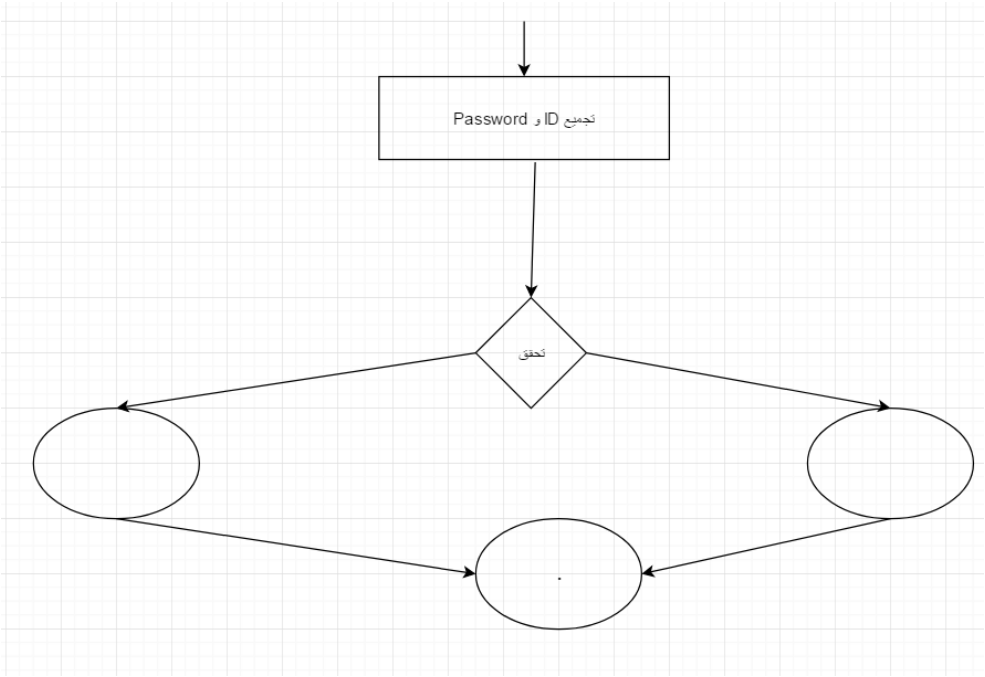
كل حالة استخدام تحتاج توصيف جدولي نذكر فيه اسم حالة الاستخدام وترتيبها حسب أهميتها وفقاً للمتطلبات والشروط البدائية (المسبقة) والتفاعلات بين الactor والنظام، كل التفاعلات التي تؤدي لحصول الactor على خدمة تدعى سيناريوهات أساسية وكل التفاعلات التي تتم ضمن السيناريوهات الأساسية تسمى سيناريوهات ثانوية.

Activity Diagram

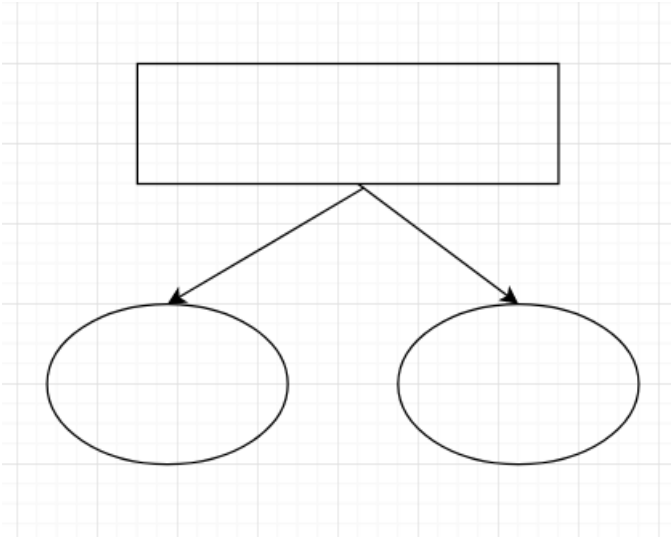
مخطط النشاط: هو مخطط تحليلي يكون جزء من SRS (Service Requirements Specification) وهو مخطط منطقي يقوم بنمذجة السيناريوهات الموجودة بحالات الاستخدام حيث أن مخطط حالات الاستخدام يركز على تفاعل النظام مع الوسط الخارجي أما مخطط النشاط يعطي صورة مرئية للخدمة كيف تتم ولإجرائية العمل في حال النجاح أو الفشل، وهو مجموعة من النشاطات التي يقوم بها النظام من وجهة نظر خارجية وكل نشاط هو مهمة تأخذ وقتاً من النظام، عملية الانتقال بين الأنشطة لا تأخذ وقتاً.

يمثل كل نشاط بمستطيل ونكتب داخله اسم النشاط.

مثال : تسجيل الدخول : نفهم السيناريوهات لنرسم مخطط النشاط ونقسمهم إلى مهام وهنا لدينا البدء بتجميع ID و password المستخدم ثم التحقق منها.



عندما تتم الخدمات على التوازي :



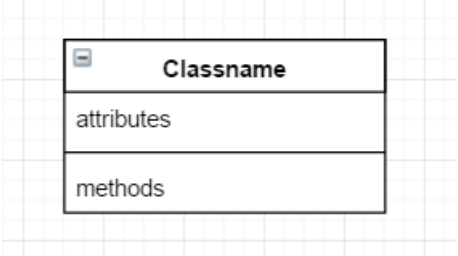
فعلياً مخطط النشاط يعطينا خوارزمية أداء الخدمات وبعض المبرمجين قادرين من مخطط النشاط كتابة الكود بشكل كامل لكننا في هذا المخطط لا نتحدث عن بنية النظام وإنما فقط النشاطات التي يقوم بها النظام فهو فقط مخطط تحليل وليس تصميم، غالباً نرسم مخطط النشاط لحالات الاستخدام المعتمدة والصعبة وليس البسيطة بهدف التبسيط ومنع التكرار وهو مخطط سلوكي يمثل نشاطات النظام من وجهة نظر مجردة.

مخططات التصميم

1. مخطط تسلسلي Sequence Diagram .
2. مخطط تعاوني.
3. مخطط صفوف.
4. مخطط حالة.
5. مخطط الحزم.

أولاً- مخطط الصفوف : هو جوهر عملية التطوير غرضي التوجه وهو آخر مخطط في عملية التصميم، نبدأ فيه من نهاية مرحلة التحليل حتى مرحلة التصميم وهو مخطط تحليلي تصميمي، يعطي وجهة نظر بنيوية للنظام ومم يتكون النظام وما هي الصفوف المكونة للنظام وكيف تتفاعل فيما بينها يبدأ به المحلل ويسلمه للمصمم ليكمله فيتم تطويره بشكل تكراري تزايدي في مرحلة التحليل تحدد الصفوف اللازمة وفي مرحلة التصميم نحدد ال attributes وال methods لهذه الصفوف.

يتم تمثيل الصفوف كما يلي:



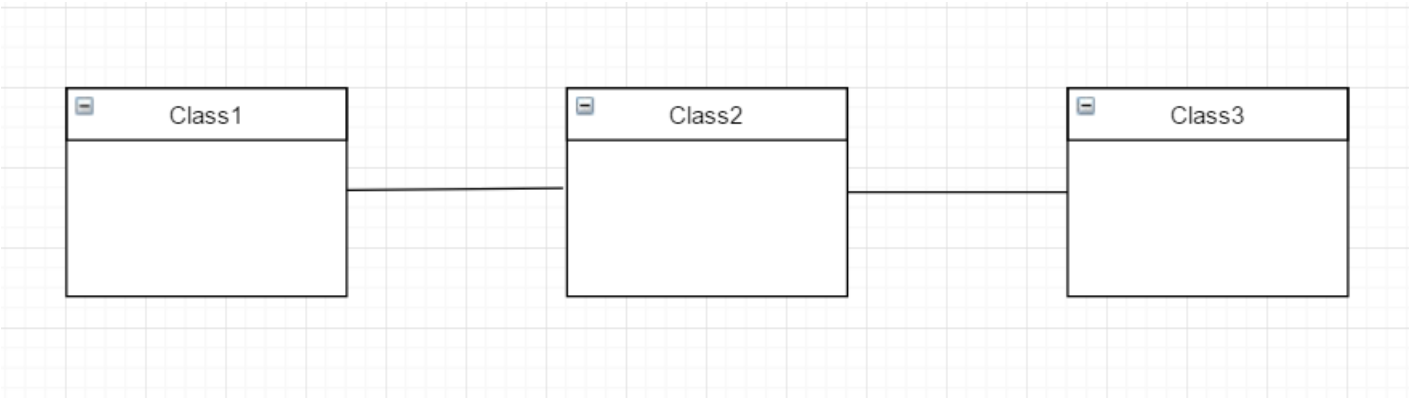
مع بعض الرموز الخاصة مثل + ،- ،#

Private data : -

Public data : +

protected data : #

في النهاية يكون النظام على الشكل التالي :



يحوي المخطط الصفوف والعلاقات فيما بينها ومتحولات كل صف والتوابع الخاصة به فيبقى على المبرمج في مرحلة التكويد تنفيذ كود لهذا المخطط.

ملاحظة: يوجد Tools لتحويل مخطط الصفوف إلى كود.

ما هي آليات اكتشاف الصفوف؟

مخطط الصفوف يجري على التوازي مع مخطط حالات الاستخدام فيساعدنا الUC والسيناريوهات الموجودة أو ما يسمى (منطق الجملة الاسمية) نقوم بالبحث عن أسماء كل هذه الأسماء مرشحة لتكون صفوف لكن ليس جميعها تكون صفوف.

مثلاً: لدي متجر أحذية الحذاء مرشح ليكون صف وكذلك نمرة الحذاء لكن نمرة الحذاء ستكون attribute للصف حذاء وهنا نحتاج للخبرة العملية للتفريق بين أسماء الصفوف والattributes .

مثال: قمنا باستخراج 100 اسم مرشح ليكون أسماء صفوف نقوم بعمل review ويبقى لدينا 10 أسماء صفوف و50 اسم بالتأكيد ليست أسماء صفوف وإنما attributes في النهاية يبقى لدينا مجموعة من الصفوف (الأسماء) حسب خبرة المصمم إما أن يشمل بعضها أو يتركها كما هي.

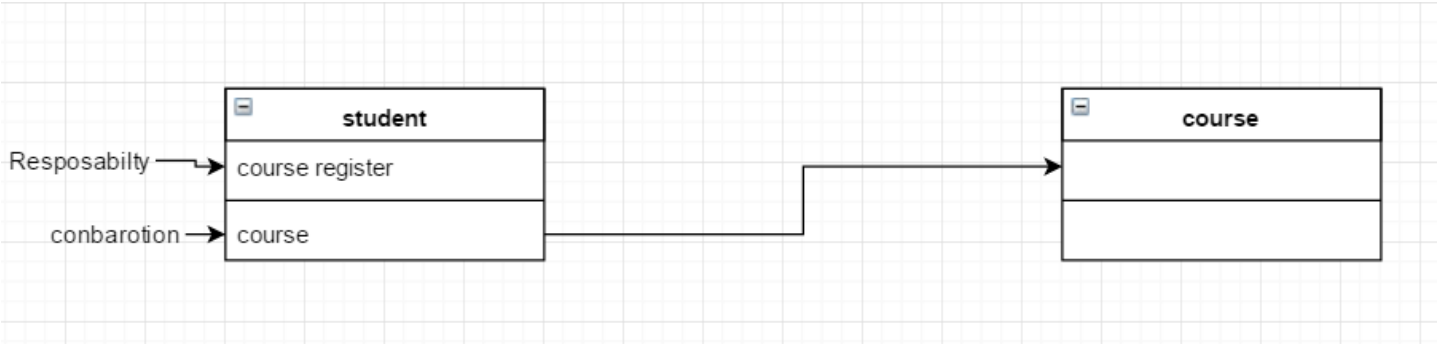
لدينا منهجية أخرى تدعى العينات المشتركة نقسم الصفوف إلى صفوف متعلقة بالأشخاص وصفوف متعلقة بالأمور الفيزيائية ... إلخ.

مثال:

أشخاص	أمور فيزيائية	أمور غير فيزيائية
طيار	طيارة	حجز
مسافر	كرسي	الوقت

هذه المنهجيات فقط للمساعدة لكننا بالنهاية نحتاج للخبرة لكن أفضل طريقة لاكتشاف الصفوف هي الطريقة التالية وهي CRC

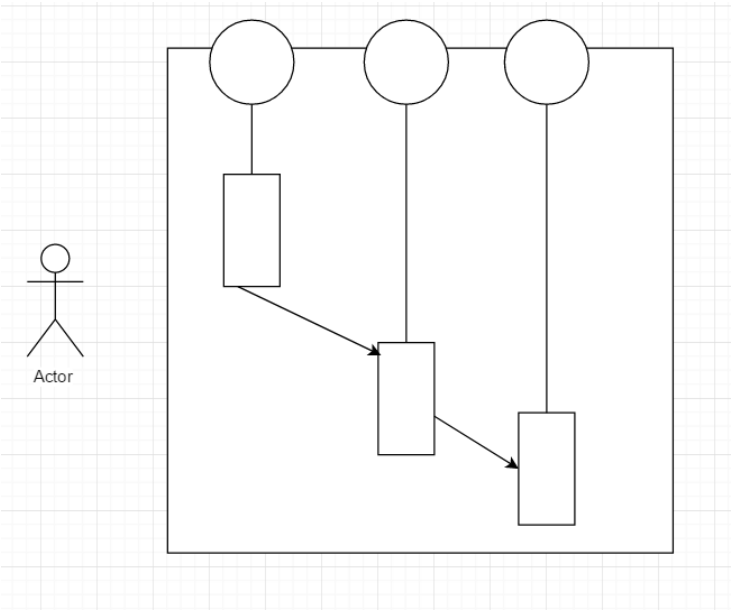
حسب المنهجية غرضية التوجه لكل صف متحولات وخدمات يقدمها ،هذه الخدمات تقدم عن طريق Methods يتم تحديد الصفوف والخدمات عن طريق جلسات وورشات عمل تحتاج عصف ذهني واجتماعات نبدأ بالصفوف الواضحة ونسجلها على بطاقة .



هذه المنهجية مهمة جداً لاكتشاف الصفوف وتصفية الصفوف التي تم اكتشافها بالطرق السابقة.

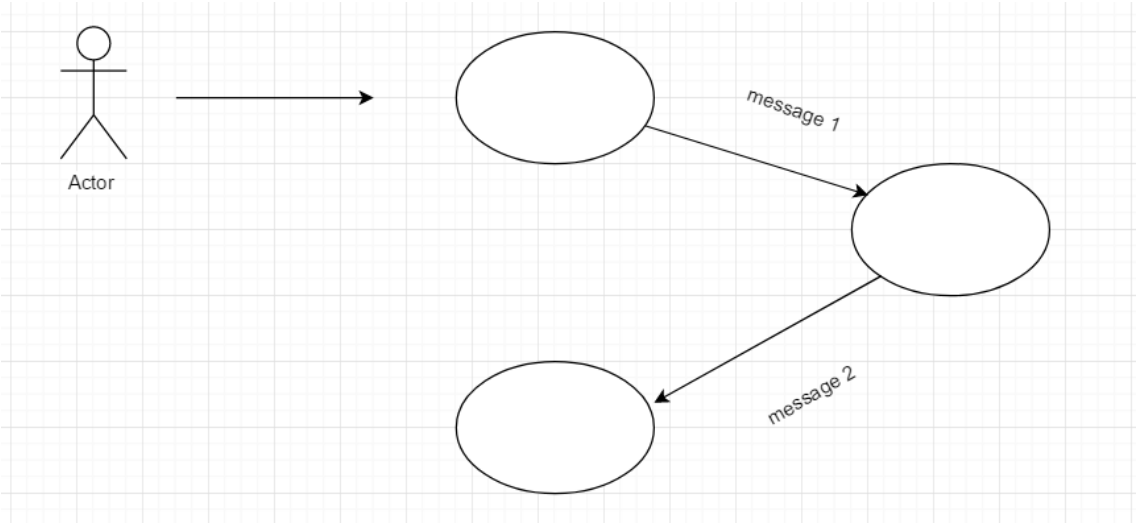
ليس شرط أن تكون جميع الصفوف متواصلة مع بعضها ولكنها بشكل أو بآخر يوجد تواصل بينها ولو كان هناك صف غير مرتبط مع غيره من الصفوف فيمكن الاستغناء عنه.

ثانياً- المخطط التسلسلي : يمثل كيف يتفاعل الفاعل (Actor) مع النظام وكيف تتفاعل أجزاء النظام مع بعضها لتأدية خدمة فمن أجل كل حالة استخدام يمكننا رسم مخطط تسلسلي، يحوي بعدين بعد أفقي تمثل عليه الأغراض (objects) وكيف تتواصل مع بعضها والبعد الشاقولي يمثل activation time لكل object



ملاحظة: يجب تحقيق التكامل بين المخططات عند وجود **object** في المخطط التسلسلي يعني وجود **class** مقابل في مخطط الصفوف وعند وجود علاقة بين غرضيين في المخطط التسلسلي يجب وجود علاقة بين الصفين المقابلين في مخطط الصفوف.

ثالثاً- المخطط التعاوني **Collaboration Diagram** : يشبه المخطط التسلسلي يركز على التفاعلات بين ال (**objects**) دون التطرق للبعد الزمني.



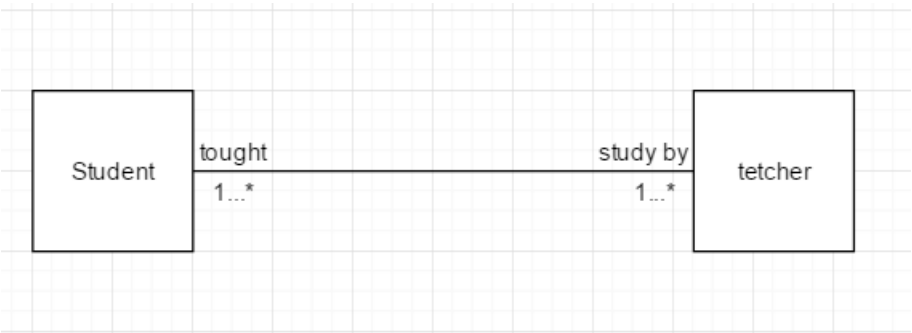
كل رسالة تمثل استدعاء تابع في **object** ويركز المخطط على الرسائل المتبادلة بين ال (**objects**) أي كيف تتفاعل مكونات النظام لتأدية خدمة معينة ومن خلال ترقيم الرسائل نأخذ فكرة كيف تتم هذه العملية.

المخططات التسلسلي والتعاوني يقدمان نفس المعلومات وغالباً أدوات رسم المخططات يمكنها تحويل المخطط التسلسلي إلى تعاوني وكلا المخططان يعطيان وجهة نظر ديناميكية لكيفية التفاعل بين الأغراض.

العلاقات في مخطط الصفوف :

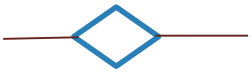
- الوراثة : أي صف أعم واشمل من صف آخر يكون أبوه وهي سهلة الاكتشاف وتكون علاقة بين صفوف.
- الاقتران : هي علاقة بين أغراض، نقول عن صفين متقارنين إذا كان وجود أحدهما يحتاج لوجود الآخر مثلاً : وجود Course يتطلب وجود teacher ووجود student فالعلاقة هنا علاقة اقتران ولكن قد يكون هناك teacher واحد وأكثر من studentوهي تعددية العلاقة.

نوع العلاقة في الاقتران (إما تجميع أو تركيب) عندما يكون لدي علاقة كل أو جزء عندها تقسم العلاقة إلى تجميع أو تركيب



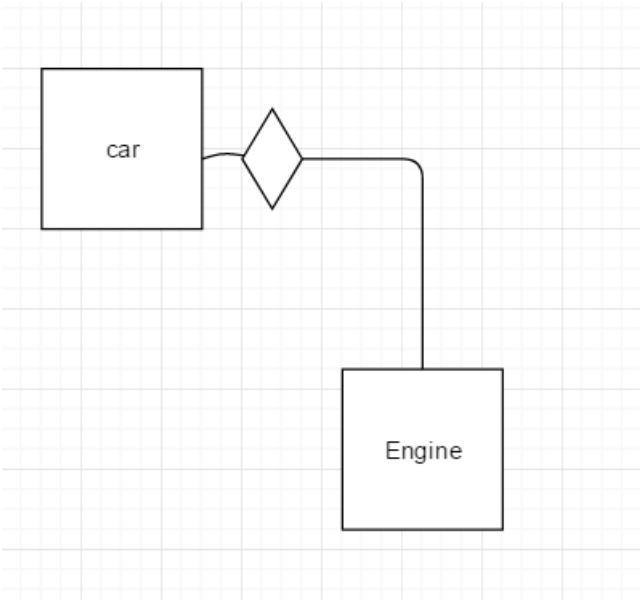
... 0 : تعني صفر أو أكثر.

... 1: تعني واحد أو أكثر.

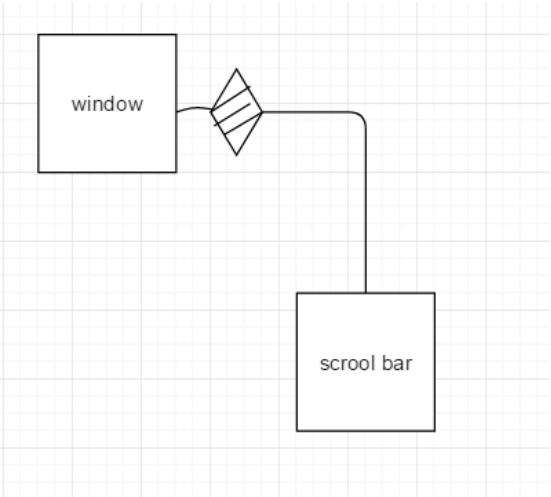
في حال كان الاقتران ضعيف أي أن الجزء يستطيع أن يعيش بمعزل عن الكل تكون علاقة ضعيفة أي تجميع Aggregationويرمز لها ب  عند جهة الكل.

أما إذا كان الجزء لا يستطيع أن يعيش بدون الكل تكون علاقة اقتران قوي أي تركيب ويرمز لها ب

مثال : المحرك يمكن أن يعيش بمعزل عن السيارة

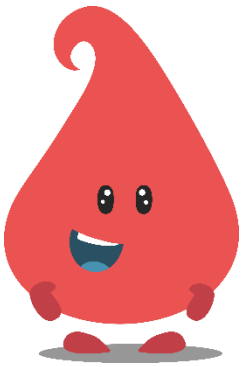


مثال عن علاقة قوية



لا يمكن لشريط السحاب أن يعيش بدون النافذة.

رابعاً: مخطط التحزيم: بينة مهيكلة عن النظام يعتمد على تقسيم النظام بحزم (packages) وكيف تتواصل فيما بينها.



نتمنى لكم دوام التوفيق ^^