Ch.5

Software Engineering

Mohammad Jadallah

Activity diagram

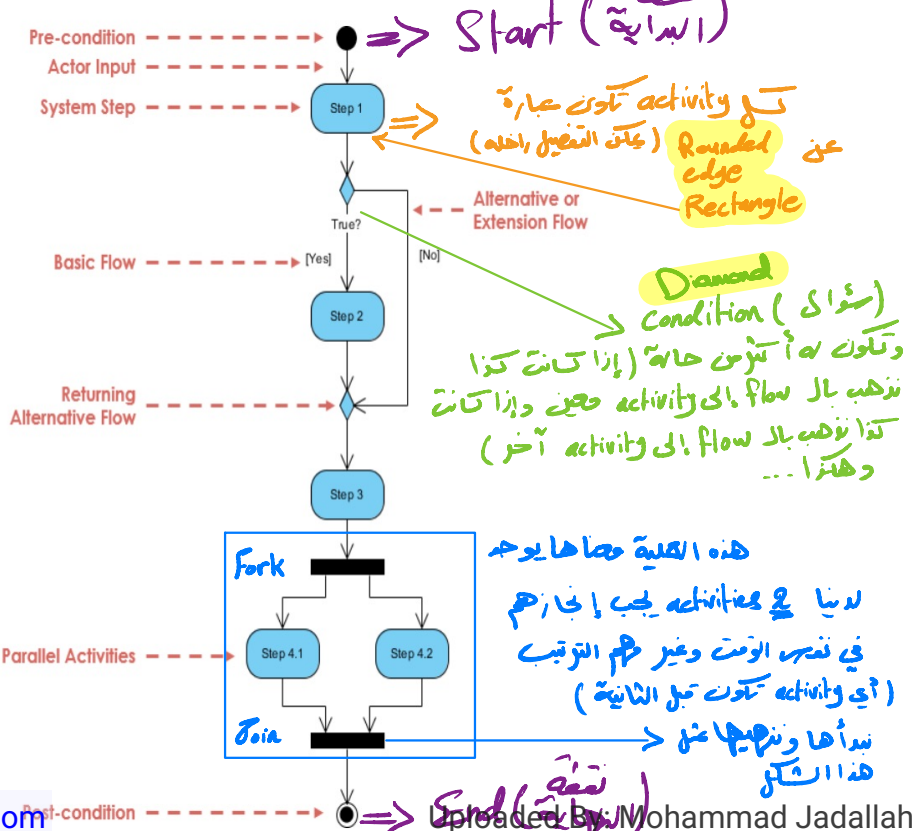
⇐ توضيح من خلاله تسلسل الخطوات لـ process معينة

⇐ يمكن استخدامه في جميع المجالات ليس فقط Software Engineering

⇐ في هذا المجال نستخدمه لوصف إما

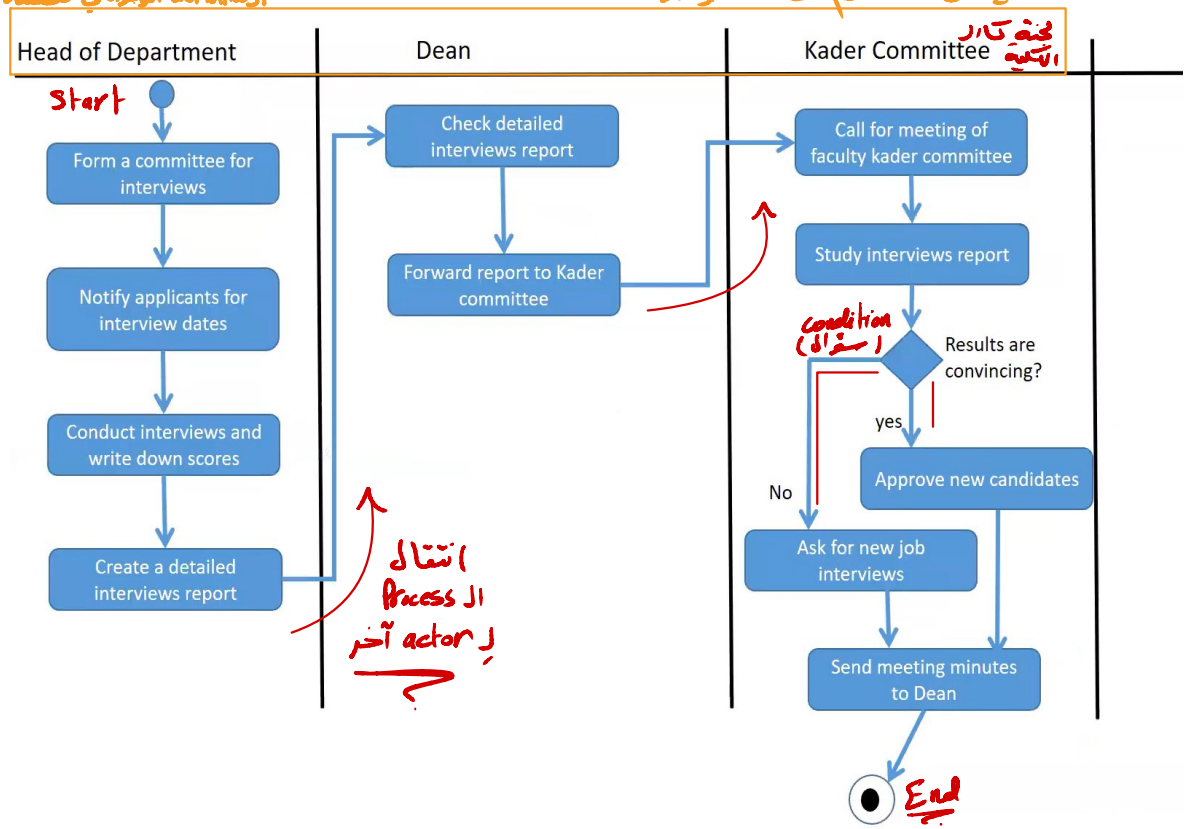
1. process للنظام الحالي الذي يتم دراسته
2. أو process للنظام المراد إنشاؤه

ملاحظة: من المحاسن أن تكون لا أكثر من use case ليس شرطاً أن تكون لواحدة فقط



مثال على activity diagram في توظيف موظفين جدد في جامعة بيرزيت

بداية بتقسيم الـ activity diagram الى 3 أقسام حسب الـ actor المسؤول عن الـ activities الموجودة في منطقته



بما معناه أن هذه المنطقة تحوي على الـ activities التي يقوم بها الـ Kader committee في الـ Process التوظيف وهكذا في كل منطقة actor مختلف

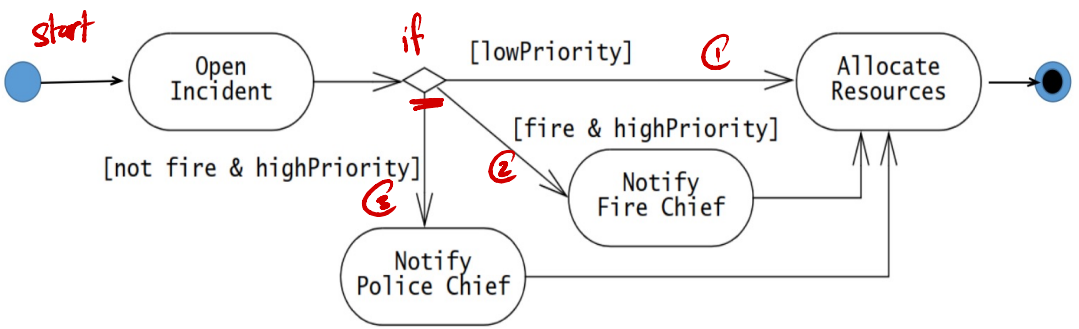


Figure 2-42 Example of decision in the OpenIncident process. If the Incident is a fire and is of high priority, the Dispatcher notifies the FireChief. If it is a high-priority Incident that is not a fire, the PoliceChief is notified. In all cases, the Dispatcher allocates resources to deal with the Incident.

* UML Class diagram

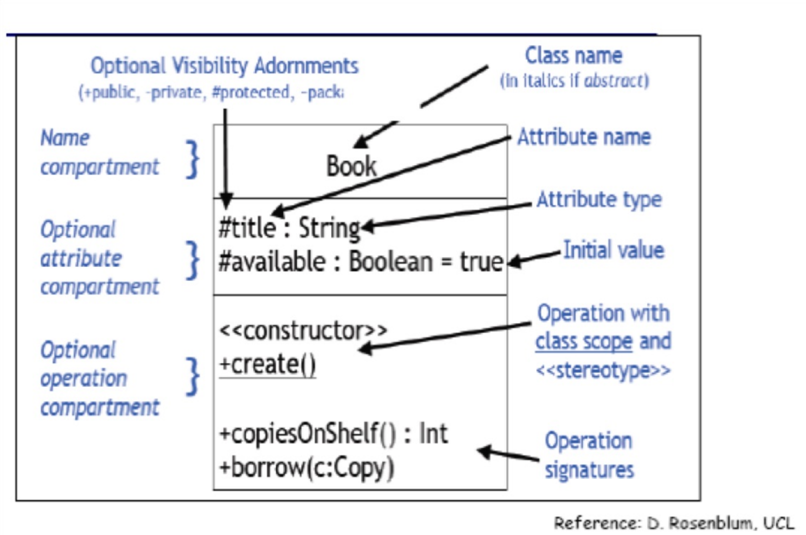
CLASS DIAGRAM

- Class diagram is also known as static system modeling
- It addresses the structural view of a problem.
- It defines the:
  - classes (concepts) in the system,
  - their attributes,
  - operations,
  - and the relationships between these classes.

* من الأكثر أهمية في عالم ال Software (يمثل ال Structure في System)

* هنالك tools موجودة حالياً إذا قمت بإعطاءها ال class diagram تقوم بعمل generate للكود مثل عمل ال data base وغيرها

* في ال class diagram لا نضع جميع ال classes الموجودة في النظام وانما نضع فقط ال entities التي سنقوم بعمل save لل data الخاصة بها (main entities saved in database)



* بنفس الطريقة التي طبقناها في Java .

DATA DRIVEN APPROACH

- **Some heuristics/hints of what kind of things are classes [Shlaer and Mellor; Booch]:**
- Tangible or "real-world" things – e.g. book, copy,
- course;
- Roles- e.g. library member, student, director of studies,
- Events- e.g. arrival, leaving, request;
- Interactions- e.g. meeting, intersection

* الاستدلال ع ال classes من خلال العلاقات والحاجة لها
مثلاً المؤلف للكتاب في نظام مكتبة هل هو class أو attribute في book class؟
⟸ المؤلف ممكن أن يؤلف أكثر من كتاب، كذلك الكتاب من الممكن أن يكون له أكثر من مؤلف بالتالي هناك علاقة واضحة بين المؤلف والكتاب
⟸ ثانياً ما هي المعلومات التي سيتم تخزينها للمؤلف هل اسمه فقط؟ الاسم، صورة، ... ؟ بالتالي المؤلف هو entity لوحده وليس attribute فقط في book class

* قاعدة من الدكتور سامر حشو منيح : إذا كان في شك عن إشي معين هو attribute ولا Entity اعرف شغلة إنو Entity لأنو الـ attribute بكون واضح

مثال آخر : الـ address في أي نقطة بشكل عام يتم التعامل معه على أنه Entity لأن الـ address بمثل تكون من city , country --- Telephone

* مثال من سلايدات دكتور سامر

EXAMPLE DATA DRIVEN APPROACH NOUN/VERB ANALYSIS

Books and journals:
The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

Borrowing:
The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

* بداية نقوم بتعيين الـ Nouns (الأسماء) الموجودة عشان نعرف شو الـ Entities الموجودة

① هون حددنا ال nouns الموجودة عنا 8

EXAMPLE (DATA DRIVEN APPROACH NOUN/VERB ANALYSIS)

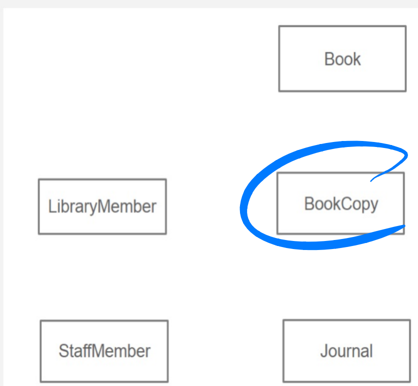
أهم هاي العملية

Books and journals:
The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

Borrowing:
The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

بعد ما حددنا ال nouns الي عنا صورنا نعرف الentities منهن

EXAMPLE DATA DRIVEN APPROACH: NOUN/VERB ANALYSIS



عملنا ال book copy entity لأنو الكتاب إلو أكثر من copy بالتالي في علاقة إذا entity

ثانيا بنحدد الـ Verbs الموجودة عنا عشان نعرف العلاقات بين الـ Entieties

هون حددنا الـ Verbs الموجودة عنا

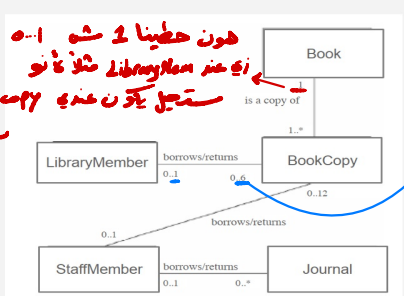
EXAMPLE DATA DRIVEN APPROACH: NOUN/VERB ANALYSIS

Books and journals:
The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

Borrowing:
The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

بهيك منحدد الـ Verbs منعرف العلاقات بين الـ Entieties

EXAMPLE DATA DRIVEN APPROACH: NOUN/VERB ANALYSIS



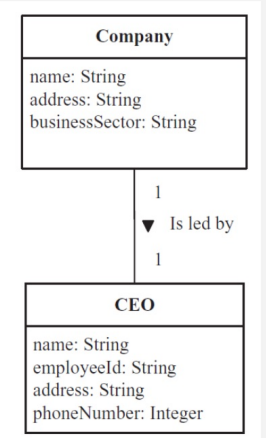
هون حطينا 1 شو 1..* يعني LibraryMem مثلا نو ستحيل يكون عندي Bookcopy بدون ما يكون عندي Book بس عادي ما يكون LibraryMem عاملها borrow

حسب الـ use case كل Library Member بقدر يستعير كم كتب بالتالي العلاقة 0...1

عن العلاقات بين الـ Entities

ASSOCIATIONS..2

- On class diagrams, an association is shown as an arc joining the two class boxes, with the name of the association next to the arc.
- In class diagrams, association names usually read from left to right and top to bottom.
- However, on a large class diagram with many classes, classes are usually in different positions relative to each other.
- To avoid ambiguity when reading UML class diagrams, COMET uses the UML arrowhead notation to point in the direction



MULTIPLICITY OF ASSOCIATIONS

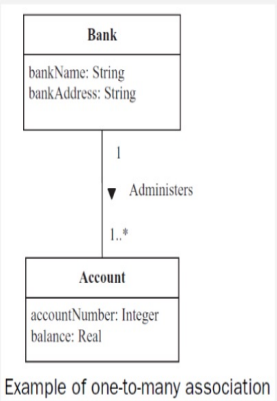
- The *multiplicity* of an association specifies how many instances of one class can
- relate to a single instance of another class.
- The multiplicity of an association can be as follows:
  - **One-to-one association.**
  - **One-to-many association**
  - **Numerically specified association**
  - **Optional association:** In an optional association, there might not always be a link from an object in one class to an object in the other class.
  - **Many to many associations**

1, 6 تعني إما 1 أو 6 لا يوجد خيار آخر

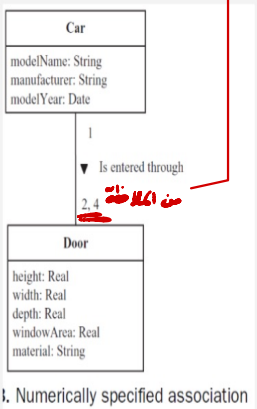
1-6 تعني من 1 إلى 6 (1,2,3,4,5,6) أو 6...1

يملا فقاعة

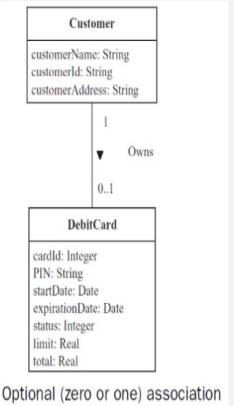
أمثلة على العلاقات السابقة



Example of one-to-many association



I. Numerically specified association



Optional (zero or one) association

في هذا المثال حسب اتجاه قراءتنا نستعمل الكلمة المناسبة

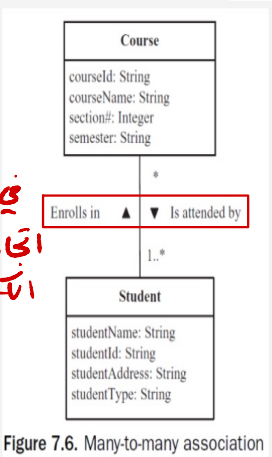
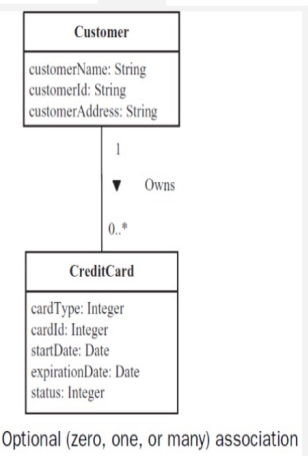


Figure 7.6. Many-to-many association



Optional (zero, one, or many) association

كل Entity مربوط مع Entity ثاني لازم نكتبلوا العلاقة اللي بينهم من هاي العلاقات.

EXAMPLE CLASS DIAGRAM

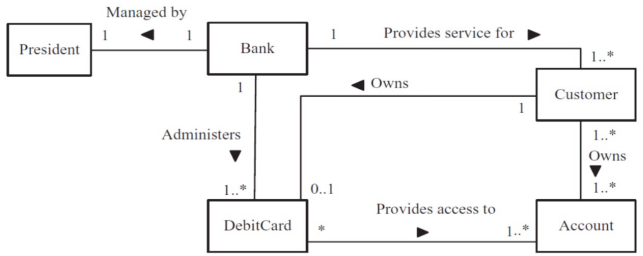


Figure 7.7. Example of associations on a class diagram

هي not-detailed class diagram يكون فقط اسم الـ class عادة متحان محكن يطلب detailed أو not يعني حسب السؤال شو بدو يدخل

TRINARY ASSOCIATION

- A ternary association is a three-way association among classes.
- An example of a ternary association is among the classes Buyer, Seller, and Agent.
- The association is that the Buyer negotiates a price with the Seller through an Agent.

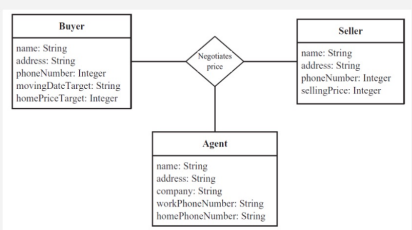


Figure 7.9. Example of ternary association

هي علاقة ثلاثية وهي تقوم بربط الـ entities مع بعض عن طريق ◇ فالمثال كل عملية بيع للعقار رها في أطراف بائع، مشتري مشرف

UNARY ASSOCIATIONS

- A unary association, also referred to as a self-association, is an association between an object of one class and another object in the same class

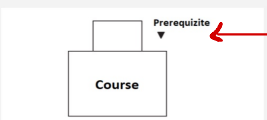


Figure 7.10. Example of unary association

هي علاقة بين الـ class ونفسه مثال عم ذاك الـ course يكون متطلب لـ courses أخرى (Prerequizite)

ASSOCIATION CLASSES

- An *association class* is a class that models an association between two or more
- classes.
- The attributes of the association class are the attributes of the association.

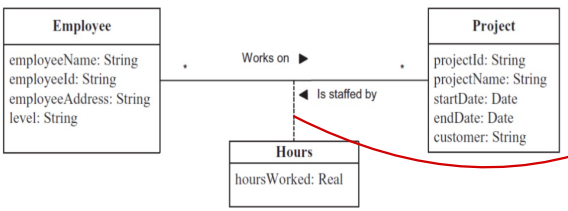


Figure 7.11. Example of association class

COMPOSITION AND AGGREGATION

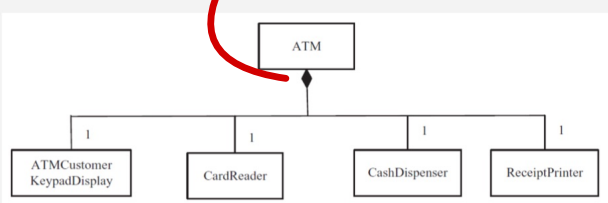


Figure 7.12. Example of composition hierarchy

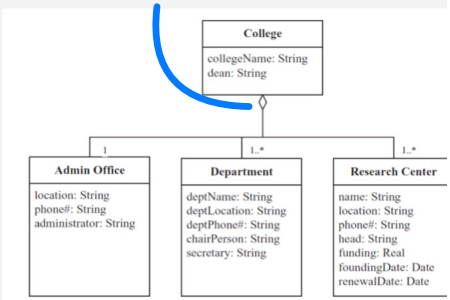


Figure 7.13. Example of aggregation hierarchy

Composition	Aggregation
* في حال تم حذف العنصر الرئيسي يتم حذف العناصر الفرعية (الرئيسية: ATM)	* في حال تم حذف العنصر الرئيسي لا يتم حذف العناصر الفرعية (الرئيسي: College)

مثال آخر على Composition إذا حذفت الـFolder يتم حذف جميع الملفات الموجودة في داخله.

GENERALIZATION/SPECIALIZATION HIERARCHY

- Some classes are similar but not identical.
- They have some attributes in common and others that are different.
- In a **generalization/specialization hierarchy**, common attributes are abstracted into a generalized class, which is referred to as a *superclass*.
- The different attributes are properties of the specialized class, which is referred to as a *subclass*.
- There is an *Is a* relationship between the subclass and the superclass.
- The superclass is also referred to as a parent class or ancestor class.
- The subclass is also referred to as a child class or descendent class

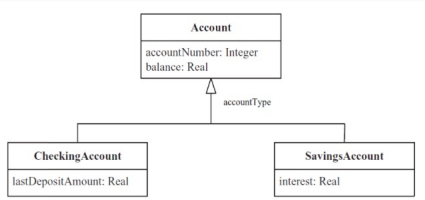


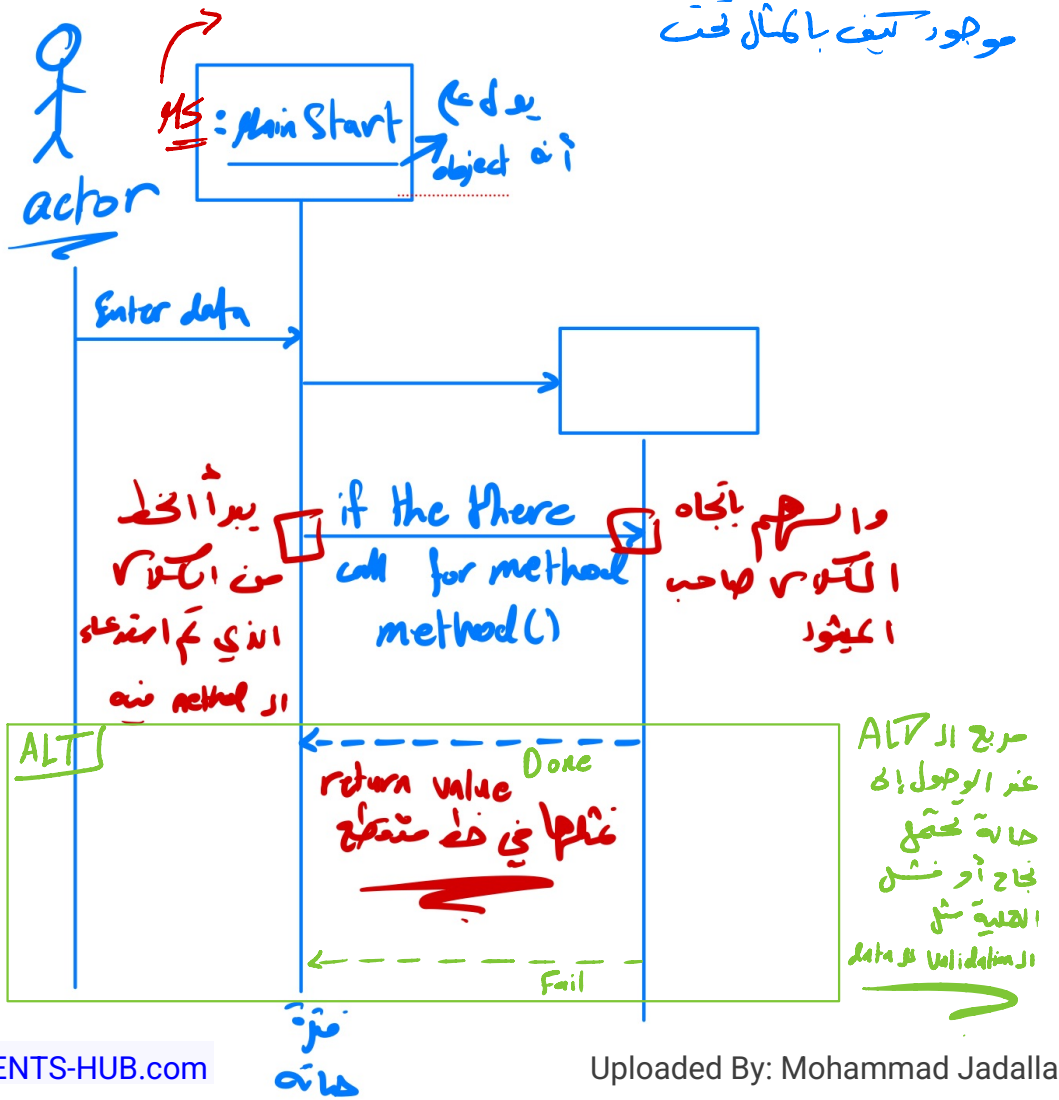
Figure 7.15. Discriminator in generalization/specialization

علاقة الـ inheritance التي سنتعلمها في حال كان هناك attributes مشتركة بين الـ entities وتدعى أخرى مختلفة مثل:

Employee
- salaryEmployee
- hourlyEmployee

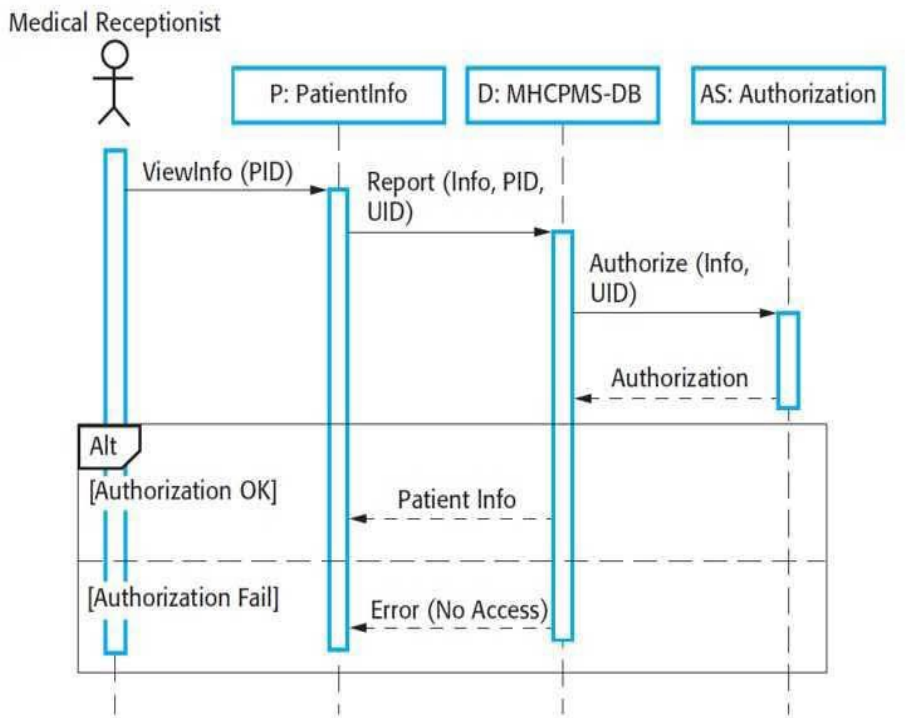
Sequence diagram

=> How the flow will work and show interactions between components



مثال على Sequence diagram من سلايدات دكتور سامر ونه

Higher level Sequence Diagram



Ex from dr. Samar lecture

=> The new BLS (Book loan System) shall allow members to log into the system using their personal Card.

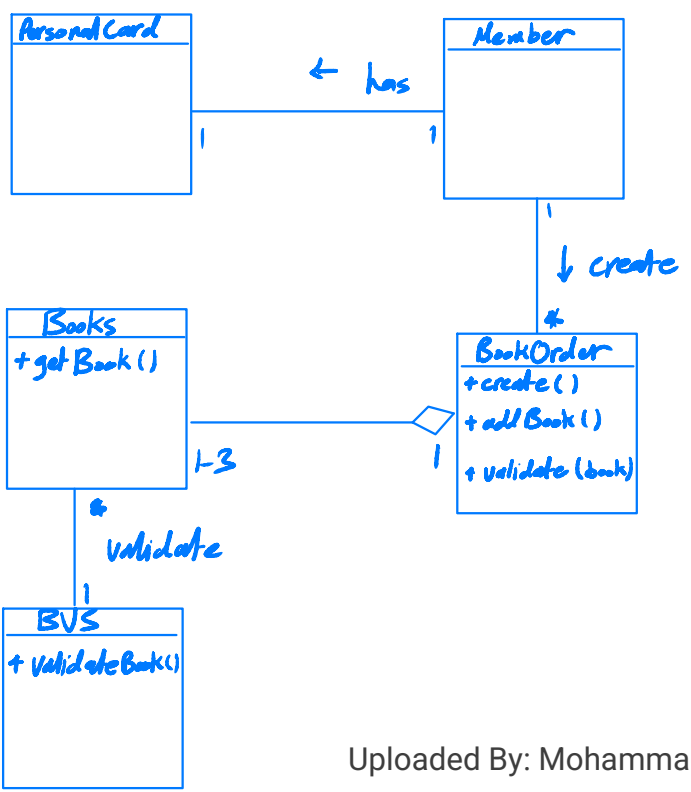
The member can then create new book order which will allow them to add up to 3 books. Before any book is added it should be validated by the book validation service.

Member can also print order information

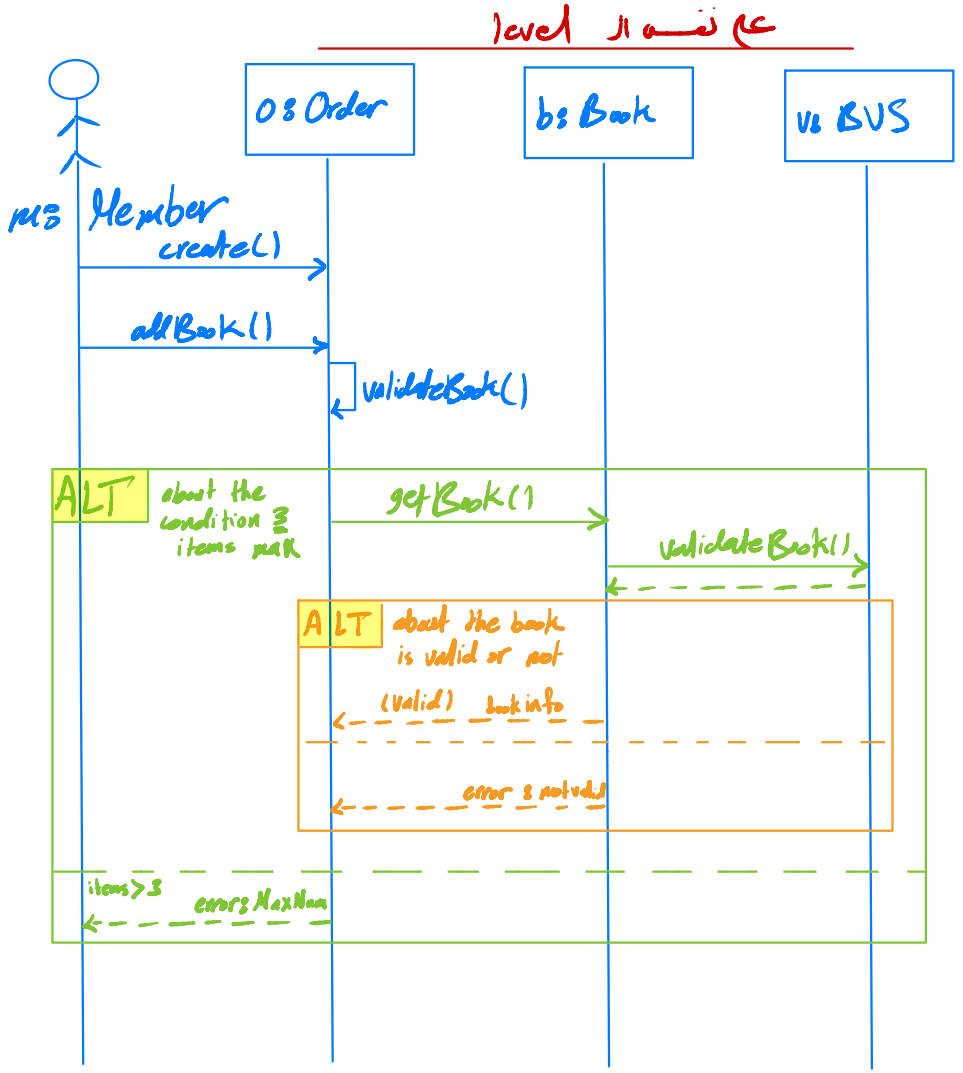
Q1) Draw UML class diagram

Q2) Draw UML Sequence diagram for creat new order use case

Q1:



Q2:



ملاحظة: الحل حل الدكتور

Components Diagram

Subsystems and Components

- **Components** are depicted as rectangles with the component icon in the upper right corner.
- Dependencies among components can be depicted with dashed stick arrows.
- In UML, components can represent both logical and physical components.
- A **logical component** corresponds to a subsystem that has no explicit run-time equivalent, for example, individual business components that are composed together into a single run-time application logic layer.
- A **physical component** corresponds to a subsystem that as an explicit run-time

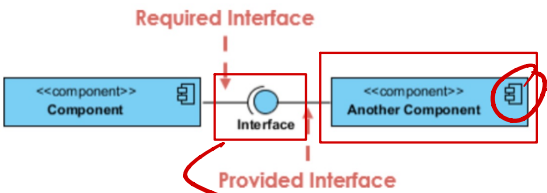
=> Components & Subsystem => مجموعة من الـ classes في نفس البكج و جميع هذه الـ classes محايدون خدمة معينة.

=> 2 types of components physical or logical

logical => Packages in Project

physical => like dataBase jar file wich is connect java with database

Required/provided interfaces



Coupling and Cohesion as design goals

- When writing a subsystem interface, one should strive to **minimize** the amount of information provided about the implementation.
- For example, a subsystem interface should not refer to **internal data structures**, such as linked lists, arrays, or hash tables.
- This allows us to minimize the **impact of change** when we revise the implementation of a subsystem

components يجب الوصول إلى حالة يكون فيها الترابط بين ال

high Cohesion + Low Coupling

Coupling & components صدى اعتماد ال ⇒ على بعضها البعض

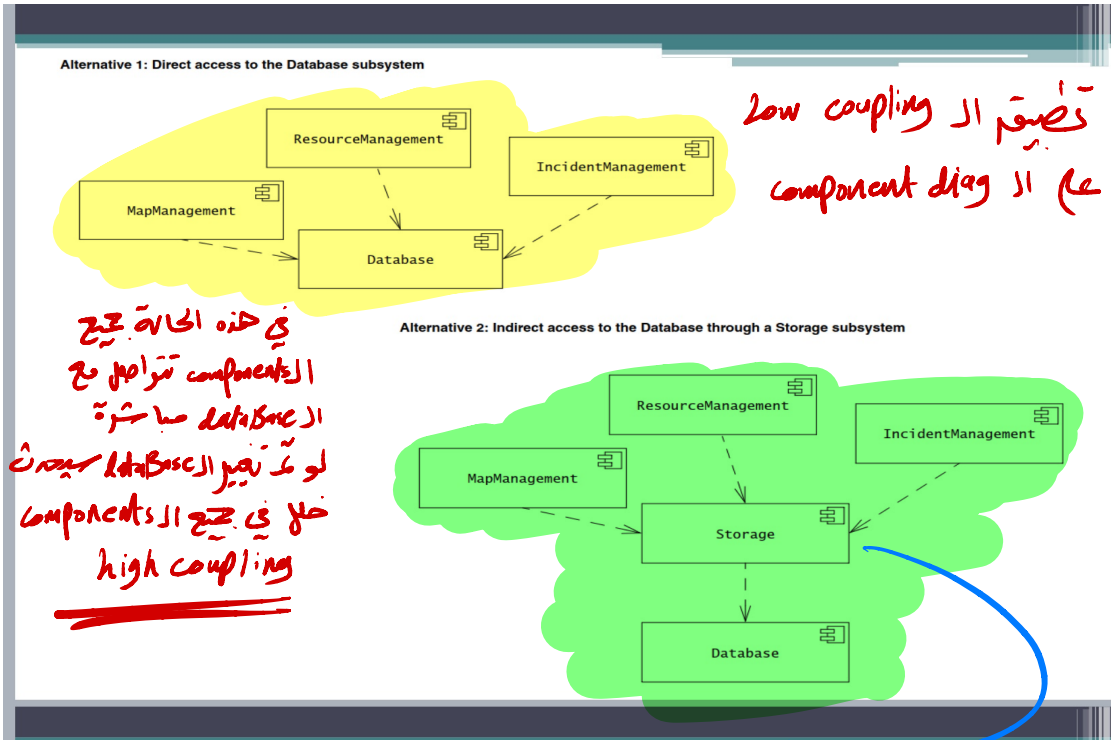
Cohesion ⇒ صدى الترابط بين محتوى ال component الواحد

تلاهما متكملان لبعضهم البعض

Coupling

- Coupling is the number of **dependencies** between two subsystems.
- If two subsystems are **loosely coupled**, they are relatively independent, so modifications to one of the subsystems will have **little impact** on the other.

شرح من سلايدات Dr. Samer عن ال coupling



تضيق الـ Low coupling على الـ component diag

في هذه الحالة جميع الـ components تتواصل مع الـ dataBase مباشرة لو تم تغيير الـ dataBase سيحدث خلل في جميع الـ components high coupling

هنا تم إنشاء component خاص للتواصل مع الـ dataBase بحيث لو تم تغيير الـ dataBase سيتم التعديل فقط على هذا الـ component وهو الأطول Low coupling

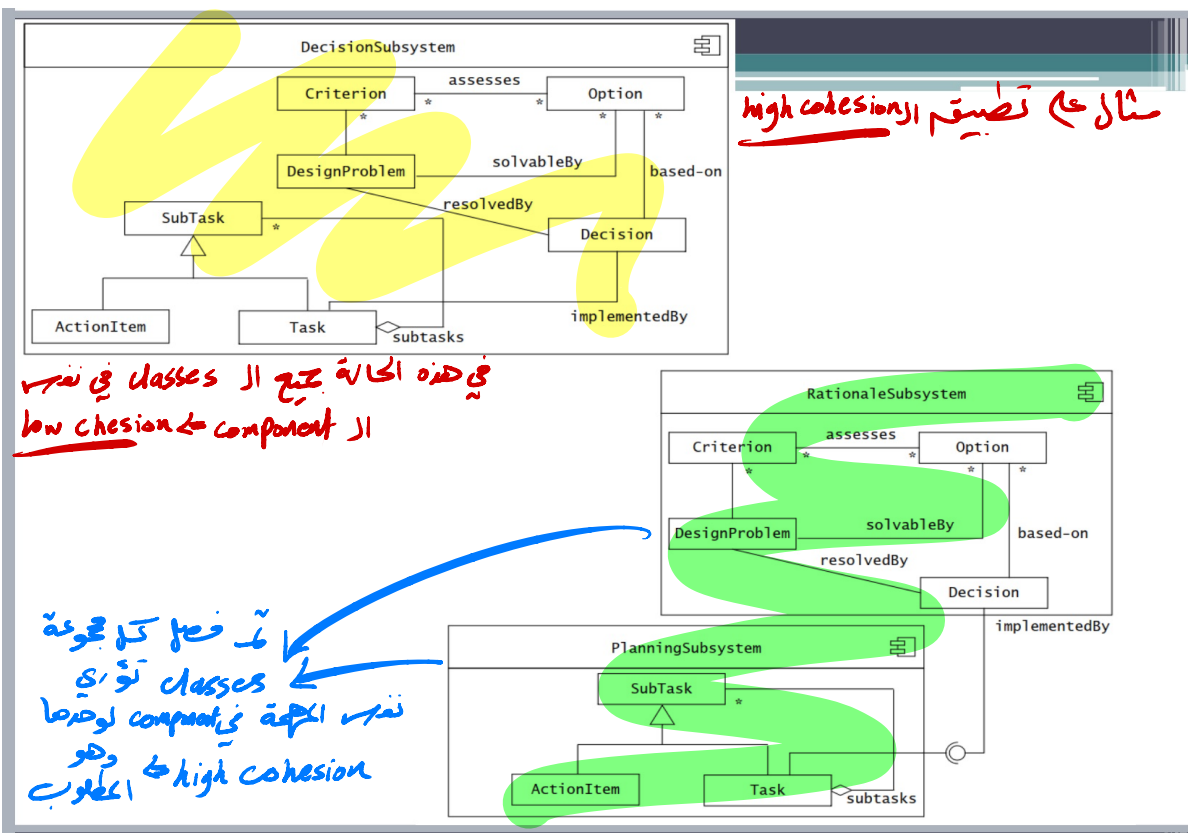
Cohesion

شرح من سلايدات dr.Samer عن الـ cohesion

- Cohesion is the number of dependencies within a subsystem. نفس الـ component
- If a subsystem contains many objects that are related to each other and perform similar tasks, its cohesion is high.

Why high cohesion?

- increases the clarity and ease of comprehension of the design
- simplifies maintenance and future enhancements
- often supports low coupling
- supports increased reuse
- a highly cohesive (i.e. a highly related functionality)
  - component can be re-used for the same specific purpose!



UML Component diagram for supermarket system

design goals come before draw component diagram

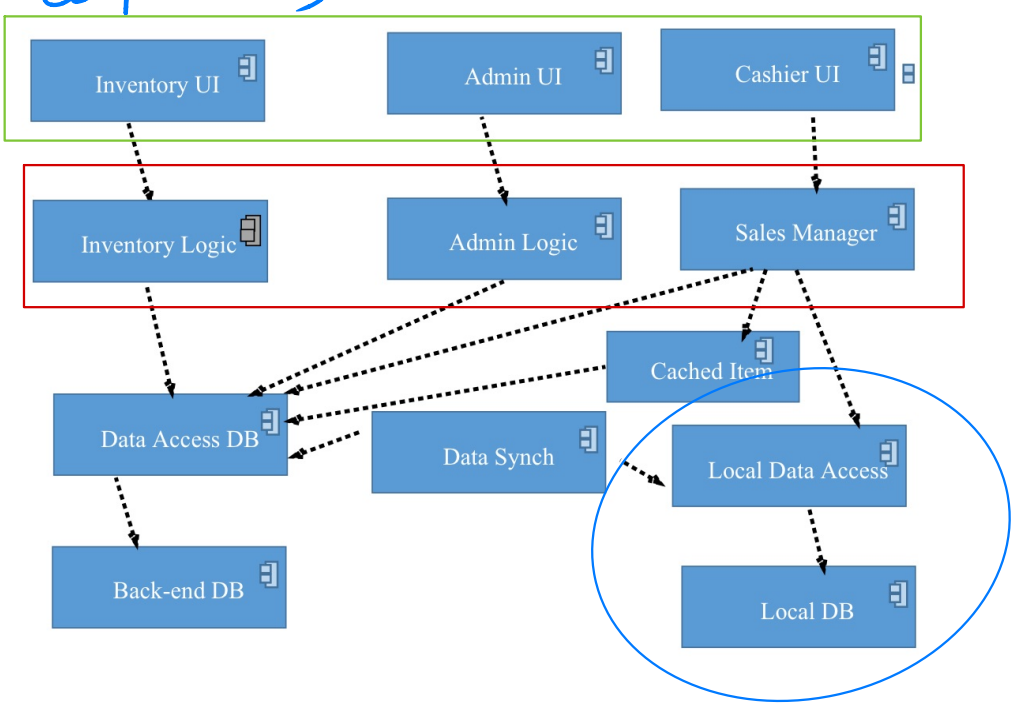
- Design goals:
  - High cohesion
  - Low coupling

→ general goals

- High reliability: the cashier should be able to continue working even if network/back-end server is down.
- Performance: the cashier should be able to retrieve every item information after scanning it with barcode reader **instantly!**

→ non functional Requirements goal

Component diagram



for high cohesion

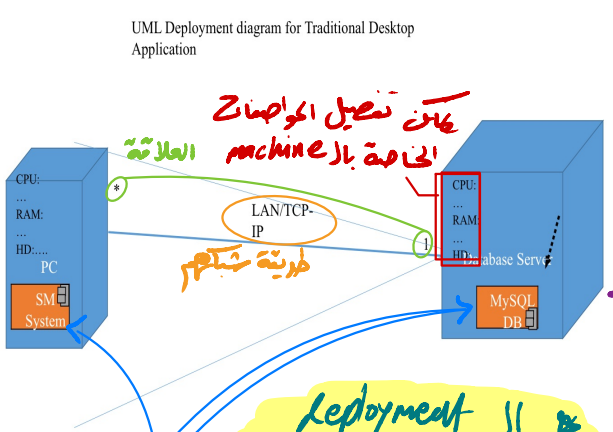
for low cupling

for non-functional Requirements goal

UML Deployment diagram

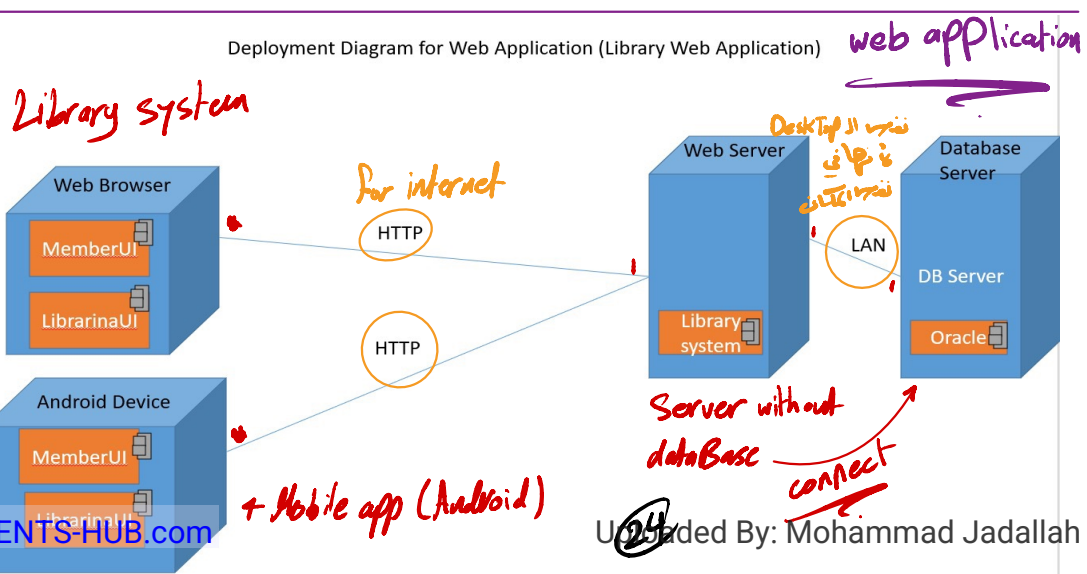
=> How the Software will install in our machine

* in deployment diag every machine is a 3D box



=> Desktop الـ Application

* الـ Deployment يوضح كيف توزيع الـ components على الـ hardware machines اللي عنا



State diagram عن

⇐ It's for core objects in the system
مثلاً: في نظام مدرسة: الطالب والحصة، المواد
وهكذا...

يوضح حالة هذا الـ object والحالات التي سينتقل إليها عند
حدوث activity معينة على هذا الـ object

مثال على state diagram
كتاب في نظام مكتبة