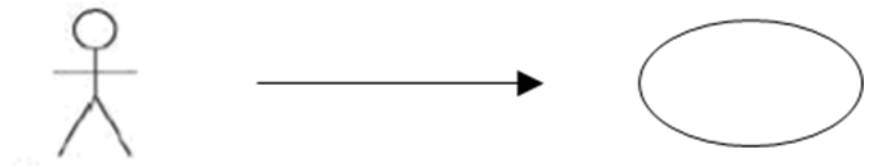
UML Diagrams

1- Use Case Diagrams

- Use cases represent ***system functionality***, the requirements of the system from the ***user's perspective***
- Use cases just focus on ***automated processes***
- Use Case diagrams show the *interactions between*
    - ***Use cases****: what* the system should do*, and*
    - ***Actors****:* anyone or anything that *interacts* with the system (Individual, group, company, …)

ال use case ودى اول رسمه فى الdesign بتعبر عن ال functionality اللى ال system بتقوم بيها واللى بطلعها من ال requirements

ال use case diagram بيتكون من حاجتين ال use case وهى على شكل oval وهى ال main functions اللى ال system بيعملها وال actors وهما الاشخاص او الاشياء اللى بت interact مع ال functions لان ال actors مش بس اشخاص لأ ممكن تكون organization او ممكن تكون function بتنده function تانيه وطبعا relationships



نشوف ايه بقى ال relationships اللى عندنا وهى انواع

Relationships: *Association:*

اول نوع هى relation بين ال actor وال use case ودى اسمها association ومعناها ان ال actor ده بي interact مع ال use case دى ولازم كل use case يبقى ليها relation سواء بينها وبين الactors او بينها وبين use case تانيه يعنى متبقاش لوحدها فى الرسمه ومفيش حاجه رايحالها وخارجه منها

- ***Association relationship*** is used to show the relationship between a use case and an actor
- Every use case ***must be initiated*** by an actor, With the *exception* of use cases in includes and extends relationships

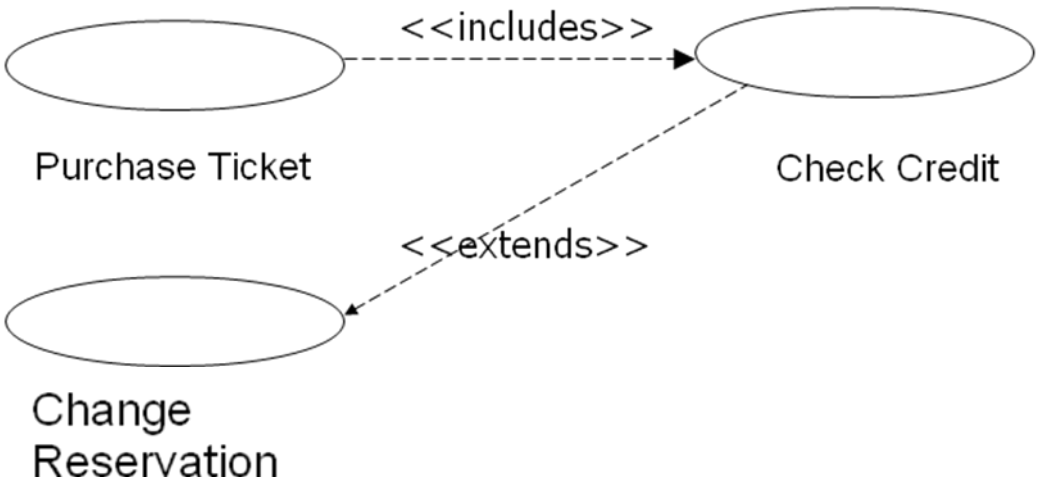
Relationships: *Includes and Extends:*

تانى نوع ال includes ,extends ودى بين الuse case وبعضها بس هو ايه الفرق بينهم ؟؟

- ***Includes relationship*** allows one use case to use the functionality provided by another use case. ***Extends relationship*** allows one use case ***the option*** to extend the functionality provided by another use case
- This is useful if two or more use cases have a large piece of functionality that is identical

ال includes معناها ان لازم ال function دى لما اجى اعملها انده على ال function دى يعنى مثلا لازم لما احجز تذكره اعمل check لل credit لكن extends يعنى ال function دى ممكن تتعمل وممكن لأ يعنى مثلا لو غيرت الحجز ممكن اعمل check لل credit او لأ

ملحوظه :فى ال includes السهم اتجاهه ناحية ال called function لكن فى الextends السهم اتجاهه ناحية ال parent function ولازم اكتب على السهم ان كان includes ولا extends

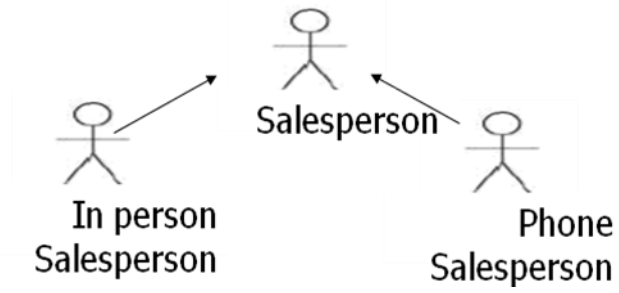


Relationships: *Generalization* :

اخر نوع وهو ال generalization وهو بين ال actors وبعض يعنى ممكن يبقى فى actors بينهم صفات مشتركه ساعتها بقول ان فى general actor بجمعهم هما الاتنين يعنى ايه؟؟ ناخد مثال

- ***Generalization relationship*** is used to show that several actors have some commonality
- For example, you may have two types of customers. If the type A customers will be initiating some use cases that type B customers will not, it's probably worth including the actor generalizations. If both types of customers use the same use cases, it's probably not necessary to show an actor generalization

لو عندى نوعين من ال salespersons الاول in sales person يعنى اللى بيقابل الناس والتانى phone sales person وهو اللى بتعامل مع الناس من خلال التليفون هما نوعين بس هما الاتنين فى الاخر salesperson



Flow of Events:

- To actually build the system, though, you'll need more ***specific details***. These details are written as the flow of events. ***The purpose of the flow of events is to document the flow of logic through the use case***
- Although it is detailed, the flow of events is still ***implementation-independent***
  - This document will describe in detail what the user of the system will do and what the system itself will do

لازم لما نعمل ال use case نكتب معاها document اسمه الflow of event وهو شرح بسيط لرسمة ال use case عشان اعرف الرسمه ماشيه ازاى وتتابع الاحداث فيها مين بيحصل الاول مين بينادى مين وهكذا ....

- Notice ***the pattern*** in the flow of events:
  - The user does something, then
  - The system does something in response,
  - Then the user does something, then the system responds, and so on

لازم وانا بكتب الflow of events اكون مختصره اوى فى كلامى يعنى اكتب الuser عمل كذا فال system رد بكذا يعنى ماكتبش تفاصيل كتير

Flow of Events Types:

- There are three types of flows:

فى انواع لل flow of events وهى :

- ***Primary flow*** is the "happy day" scenario, or the most frequently used path through the use case

وده لما يكون ال system شغال تمام ومفيش اى مشاكل حصلت

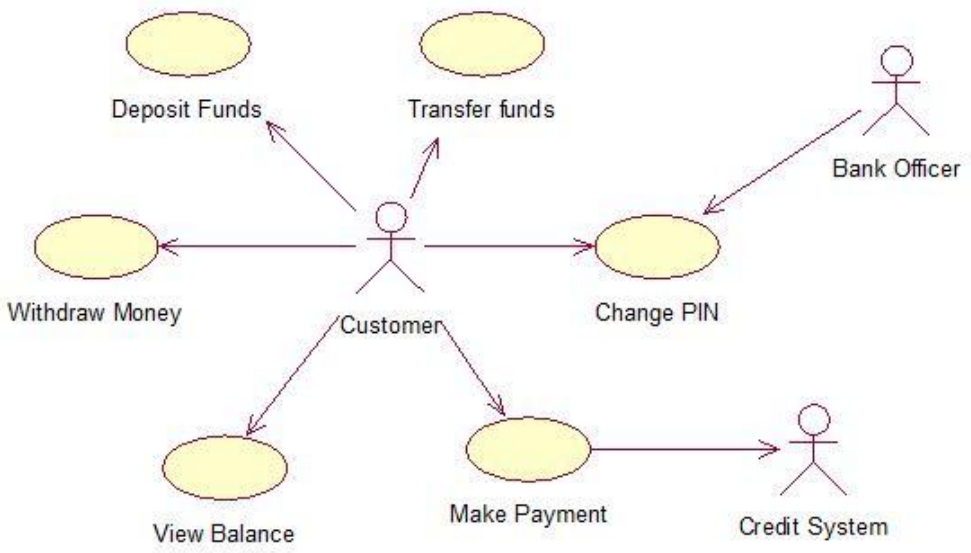
- ***Alternate flows*** are deviations from the primary flow that do not suggest an error condition

وده لما يحصل حاجه فى نص فلازم يكون فى بديل يعنى مثلا الuser كان ال credit بتاعه مفيش فيه فلوس او اى سبب تانى يسبب فشل لعمليه معينه فى system

- ***Error flows*** are deviations from the primary or alternate flows that suggest some sort of error condition. Error flows suggest that there is a problem with the system itself

وده لما يحصل خطأ فى ال system يعنى النور يقطع او الكومبيوتر يهنج او اى مشكله تحصل لل system نفسه

وده مثال لرسمة ال use case لل ATM machine



2- Sequence Diagrams:

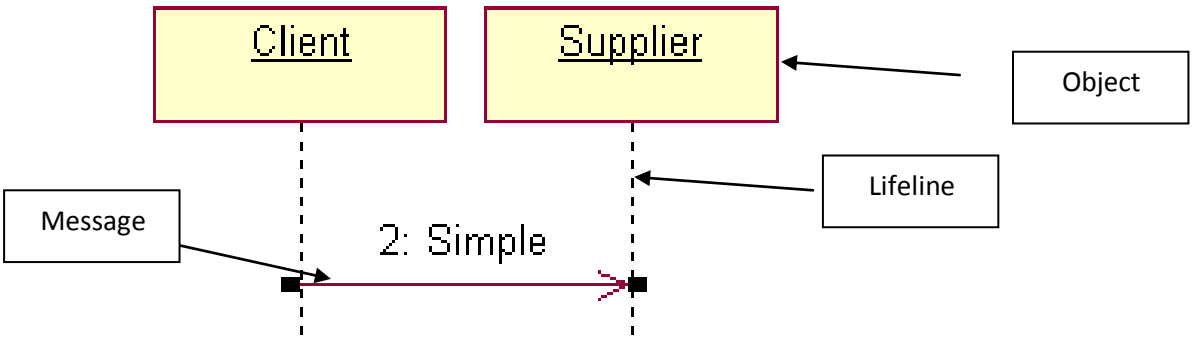
ودى تانى رسمه فى ال design وهى ان امسك كل flow of event واعمله رسمة sequence واوضحه step by step وال sequence diagram بيتقسم لأربع حاجات الobjects ودى الحاجات اللى هعمل معاها interaction وmessages ودى اللى هيبعتها كل object للتانى وال actor وهو اللى بيعمل initiate لل flow واخر حاجه ال order وده مهم جدا ان يكون فى ترتيب للاحداث

- Shows, ***step-by-step***, flows through a use case:
  - What ***objects*** are needed for the flow
  - What ***messages*** the objects send to each other
  - What ***actor initiates*** the flow
  - What ***order*** the messages are sent
- Sequence diagrams are interaction diagrams ***ordered by time***
  Each diagram represent one of the flows through a use case

- ***Actors*** and ***Objects*** shown at the ***top*** of the diagram
    - Each object has a lifeline, drawn as a vertical dashed line below the object

      - ***The lifeline begins when the object is instantiated and ends when the object is destroyed***
- A ***Message*** is drawn between the lifelines of two objects to show that the objects ***communicate (each message will become an operation)***. Messages can also be ***reflexive***, showing that an object is calling one of its own operations

ال actors والـ objects بترسمو فوق وكلهم ليهم lifeline بترسم vertical وبيبقى dashed
والـ message بتعبر عن ال communication بينهم وبتترسم بين الـlifelines بتاعة ال objects وممكن يبقى فيه reflexive messages وبتبقى معناه ان الـobject بيعمل calling لـfunctions جواه



وده مثال لل flow of events بتاع الـATM machine

Example: ATM Withdrawal

1. **Customer** *inserting his card* into **the card reader**
2. Then, the card reader *reads the card number*,
3. *opens* **Joe's account** object,
4. *And initializes* ***the ATM screen****.*
5. The screen *prompts* Joe for his PIN. He enters 1234.
6. The screen *verifies* the PIN with the account object.
7. The screen *presents* Joe with his options, and he *chooses* withdraw.
8. The screen then *prompts* Joe for the amount to withdraw.
9. He *chooses* $20.
10. Then, the screen *withdraws* the funds from the account.
11. The account object, *verifies* that the account contains at least $20.
12. Then, it *deducts* the funds from the account.
13. Next, it *instructs* **the cash dispenser** to provide $20 in cash.
14. Joe's account also *instructs* the dispenser to *provide* a receipt.
15. Lastly, it *instructs* the card reader to *eject* the card

لو جينا من هنا نطلع ايه هى الـobjects والـmessages والـactor :

Objects: cash reader, ATM screen, user`s account, cash dispenser
Actor: user (joe)

Messages:

Accept (enter) cash card, read card number(reflexive), initialize screen, open account, prompt PIN, accept (enter) PIN, verify PIN, prompt transaction type, select withdraw, prompt amount , accept (enter) amount , withdraw funds, verify funds (reflexive), *deducts* the funds (reflexive), provide cash, provide receipt, eject card

بس بعد ما طلعنا كل حاجه نبتدى نرسم