12/31/2022

Abdel-Rahman Gamal Ahmed

no campany

Design pattern

Structure Design pattern



Table of Contents

[1-Topic: 2](#_Toc123488911)

[2-structure DP: 2](#_Toc123488912)

[3-Adapter DP : 2](#_Toc123488913)

[3.1-problem : 2](#_Toc123488914)

[3.3- structure of adapter: 4](#_Toc123488915)

[3.4-appilicability : 4](#_Toc123488916)

[3.5- cons and pros : 5](#_Toc123488917)

[4- Bridge: 5](#_Toc123488918)

[4.1-problem : 6](#_Toc123488919)

[4.2-applicaplty : 7](#_Toc123488920)

[4.3-cons and pons : 8](#_Toc123488921)

[5-important notes 8](#_Toc123488922)

[6-composite : 9](#_Toc123488923)

[6.1-problem : 9](#_Toc123488924)

[7.2-calculte total : 10](#_Toc123488925)

[7.2.1-why common interface : 10](#_Toc123488926)

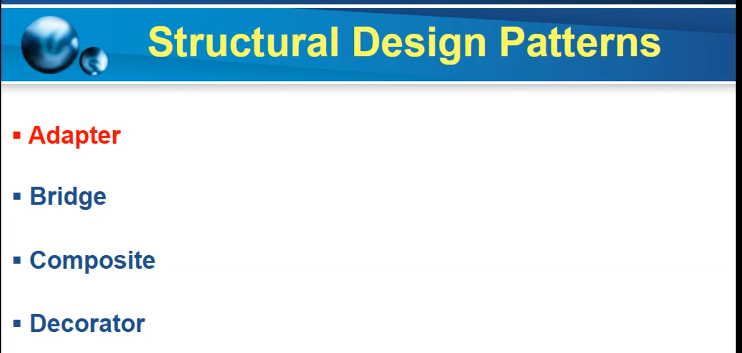
[7.3-structure: 12](#_Toc123488927)

[7.4- when can use this pattern: 12](#_Toc123488928)

[7.5-applicapilty : 12](#_Toc123488929)

[7.6-cons and pros : 12](#_Toc123488930)

# 1-Topic:



# 2-structure DP:

دى زى مقلنا قبل كدة ان بنستخدام ال structure DP عشان نحل المشاكل الى هتقبلنا واحنا بنعمل design للstructure بتاع المشروع بتاعنا . وان ازاى نقدر نجمع الكلاس مع بعض. وايه علاقه الكلاس ببعض ف large structure .

# 3-Adapter DP :

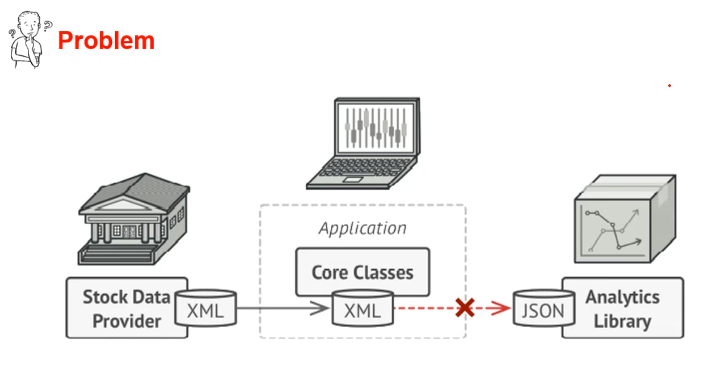


ال Adapter دى هو باضبط زى المشترك الى بيحول الوصله الثلاثبه بتاعه الاب لوصله ثنائيه .

ف كدة وظيفه ال Adapter ان بيخلى اتنين كلاس مش عارفين بتواصله مع بعض يتواصله مع بعض عادى . واحنا قلنا ان 2 كلاس مش عارفين يتكلمه مع بعض عشان كل واحد فيهم بيتكلم لغه غير التانى .

* المترجم الى وافق بين اتنين بيترجملهم هو عبارة عن Adapter . ف نفس الفكرة ف عندنا ان فيه 2 كلاس مش عارفين يتكلمه مع بعض يعنى مش بيتكلمه لغه واحدة ف عشان كدة هنحطلهم Adapter يقوم بدور المترجم بين الاتنين كلاس .

## 3.1-problem :

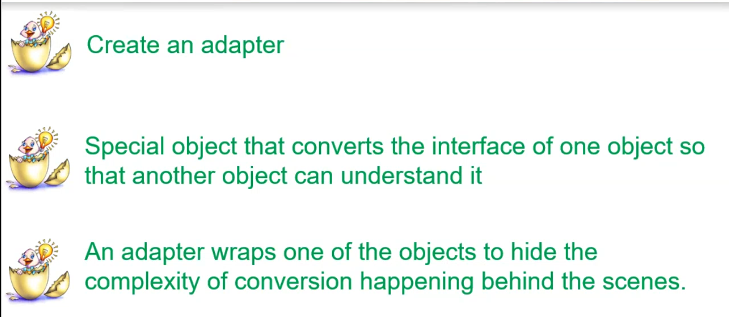
انا عامل برنامج عشان يقره اخبار البرصه ف عندنا data provider بنجيب منه الداتا . ف data provider دة بيوفر للapp بتاعى الداتا على شكل xml طيب انا عاوز استخدام ف ال app الى عامله library بتقره وتكتب Jason ف كدة ال app بتاعى وال library مشهيعرفه بتواصله مع بعض .دى المشكله الى عوزين نحلها .

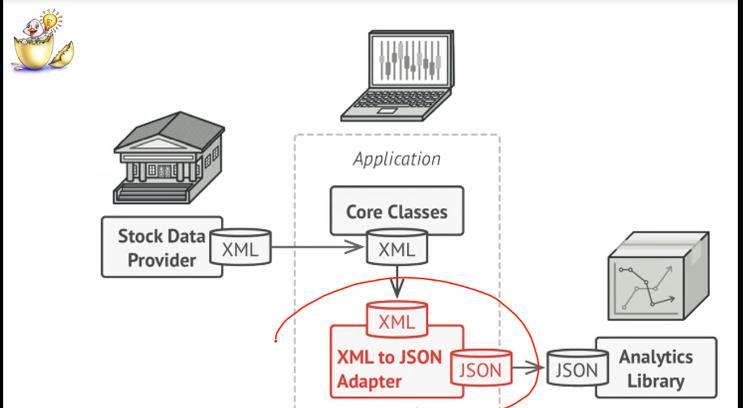
ف الحل :

-1ان نوفر adapter ف النص.

2- وهنستخدام ال adapter عشان يحول من نوع معين لنوع تانى وف حالتنا هنا عشان يحول من xml لJason والعكس .

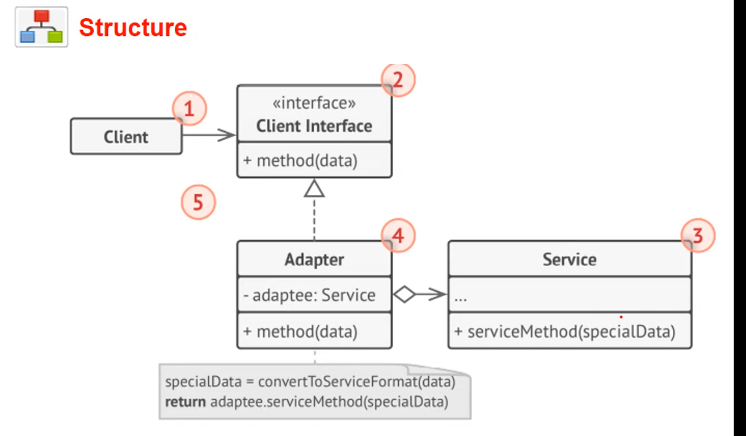
3- ف كدة نقدر نقول على ال adapter انه بيعمل wrap يعنى بيخبى واحد من الاتنين object عن التانى يعنى من الاخرالapp بتاعى مش هيتعامل مع ال library بتاعه ال Jason بشكل مباشر هو هبتعامل مع ال adapter .وال adapter هو الى هيتعمل مع ال library .



ودة شكل التصميم لما نركب الadaptor . كل الحكايه ضفنا ال adapter الى بياخد input على شكل xml وبيطلع output عل شكل Jason .

## 3.3- structure of adapter:

* هنا ال client هو المكان الى هاخد منه الداتا .
* وال services دى الحاجة الى عاوز اكلمها يعنى ال library الى كانت ف المثال الى فات .
* ف كل الى عملناه عملنا interface وعملنا concrete class يعمل implement لل interface دة.
* الinterface دة زى مهو واضح ف داله واحد .
* ال concrete class دة فيه object from services وجواه كمان method ورثها من ال interface بتاخد data من نواع الى بيوفر data provider وتحول للنوع الداتا الى عوزها ال services .
* جوة الداله الى ف concrete class انا بحول الداتا للنوع الى بيقبله ال services . وزى مقلنا انا معايه object من الservices ف ال concrete class ف اقدر انادم اى داله من بتوع ال services وابعتلها الformat بعد ماحولته . ودة الى ظاهر ف الاسكرين.
* ف خلاصه الى حصل ان عملت layer جيدة بينى وبين الservices وخليها تحول لنوع الداتا الى بتخدها الservices وكمان خليت جوها object من الservices عشان اقدر انادم اى داله وابعتلها الداتا بعد محولتها للformat المطلوب .

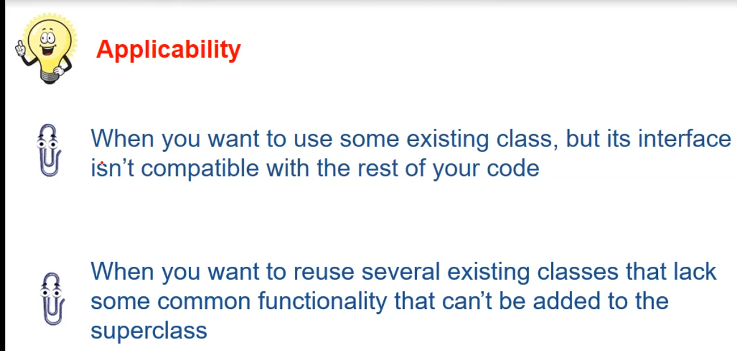


* احنا على فكرة استخدمنا ال adaptor class دة ف الprograming network ومثال على الى استخدمنها لو مخنتنيش الذاكرة ال netwokstream ,file .

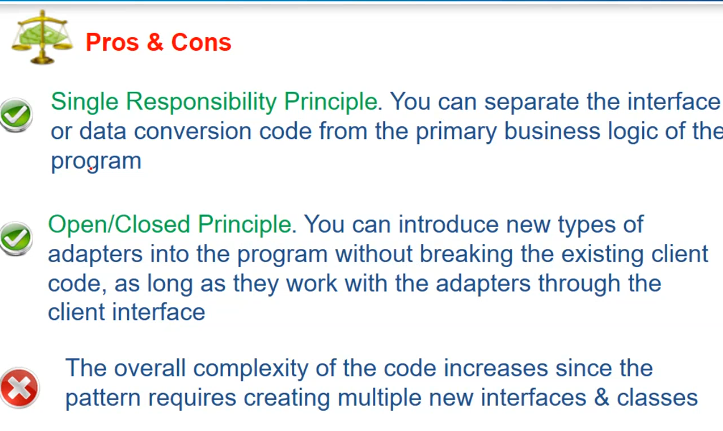
## 3.4-appilicability :

امتى هستخدام ال adapter DP :

1. لما يكون معنا اى اتنين "class ,library " وعوزين يتكلمه مع بعض وهم مبيتكلموش نفس اللغه .
2. وبالشكل دة نقدر نستخدام library قديمه ف حجات جديدة بنعملها مع انهم من تكنولوجى مختلفه او جين من data format مختلف ودة كله بسبب ان هيكون ف layer ف النص هى الى هتقوم بعمليه التحويل .

* والاسكرين دى نفس الخطوات الى شرحته .

## 3.5- cons and pros :

دى المزايه والعيوب .

# 4- Bridge:

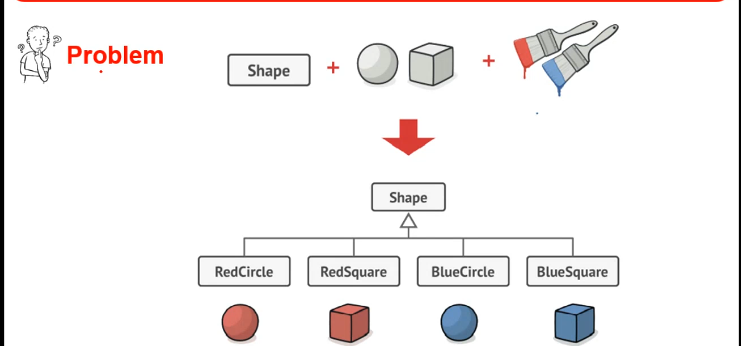


من اسمه كدة هو كبرى مبين two class . طيب كدة ايه الفرق بين ال adapter وال bridge ؟ الفرق ان

* الadapter : هو واقف بين اتنين ميعرفوش بعض وواحد منهم عاوز يتواصل مع التانى بس اللغه بتاعه كل واحد فيهم مختلفه عن التانى ف عشان كده احتجنا الadaptor .
* ال bridge هو كبرى بين اتنين كلاس بس الميزه هنا ان ال two class كانوا ف الاصل كلاس واحد كبير لاكن بعد شويه الكلاس الكبيرة دة فصلته لاتنين وخدنا مثال على كدة "ان ممكن الاقى مجموعه من ال member واخدين جمب لوحدهم عن الكلاس وبعملهم معامله مختلفه عن الكلاس ودة معناه ان الناس دى مكنها اصلا مش ف الكلاس ومفروض اخليهم ف كلاس لوحدهم وده معنها برضو ان ف علاقه بين ال2 class الى ظهره ف العلاقه دى اسمها bridge . وتقريبا الشرح دة كان ف 2part ف ال intro slide"
* ف ال bridge هو حلقه وصل بين اتنين كلاس هم ف الاصل كانوا كلاس واحد وانت قررت تفصلهم عن بعض عشان يبقا ف coherence .

## 4.1-problem :

* عندى كلاس اسمهshape ف اكتشفت ان عاوز امثل شويه اشكال مختلفه بناء على ال شكل الهندسى geometry. يعنى هعمل كلاس اسمه circle ,squire وخليهم يورثه من shape .
* ف بعد شويه حبيت اقسم ال shape بناء على الcolor .ف مثلا اللون الاحمر بيمثل الاعداء والblue shape بيمثل الاصدقاء دة يعنى لو بعمل تصميم لعبه .لاحظ ان بقول عاوز اقسم ال shape مش الcircle مثلا . يعنى كده كل geometry هيكون فى منها عدد نسخ بعدد الالوان . لو عندى دايرة هيكون ف نسختين منها نسخه بالون الاحمر ونسخه بالون الازرق دة لو عندى لونين بس .



* ممكن تقلى دى نفس المشكله بتاع ال abstract factory ومصنع الاساس الى خدناهم قبل كدة . هقلك لاعشان هنا ال 2 diminution ملهمش علاقه ببعض .الاللوان ملهوش علاقه ب ال geometry . اما امبارح كان ف product وكل product ليه انواع مختلف عن التانى بس ف الاخر مرتبطين ب family واحدة .art, modern ,Victorian دول عيه واحده ليهم علاقه ببعض يعنى .
* لو مشينا ف طريق ان كل شويه ازود diminution تانى يعنى ممكن اعمل تقسيم تانى على حسب المساحه "مساحه صغيرة ومساحه كبيرة ". ف كدة التصميم هيكبر بشكل مش منطقى ف مفروض نشوف حل تانى.

**ف الخلاصه ان المشكله بسبب**: انك عاوز تزود diminution جديد ملهوش علاقه بالموجود .

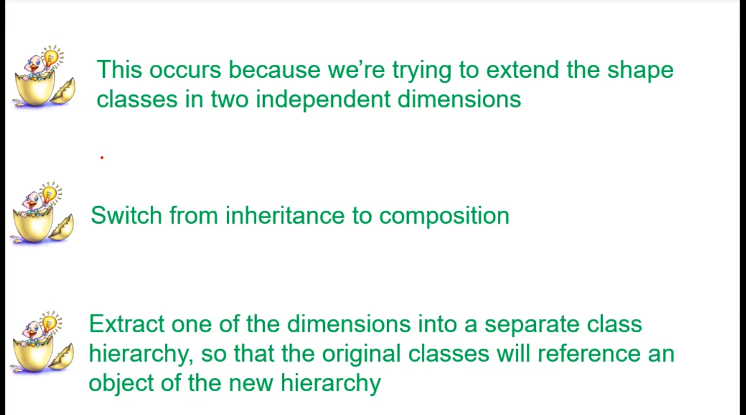
**طيب الحل** ؟هنتحول من inheritance ل composition .

هتشوف عندك كامdiminution هتلاقى اتنين ف هتحول واحد فيهم من inheritance الى composition .

1. يعنى لما اخلى فيه interface اسمه shape واخلى الى يورث منه هم كلاس ,squire circle . واخلى جوة ال shape يكون فيه object من color
2. الحل التانى ان اعكس المضوع يعنى اخلى فيه interface اسمه shape واخلى الى يورث من هم كلاس blue ,red ويكون ف object من geometry مثلا جوة ال shape .

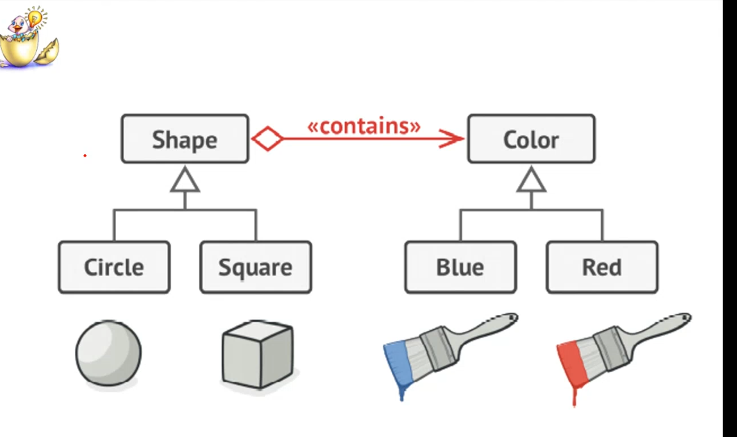
طيب انهى diminution فيهم الى اخليه inheritance ؟ الاقرب ف ال business يعنى اشوف ال logic بتاع المشروع مين فيهم . اهم ف بنختار الحاجة الاهم ونخليها inheritance

دة نفس الكلام الى شرحه لحل المشكله والمشكله نفسها .



ف دة شكل التصميم بعد محليت المشكله .

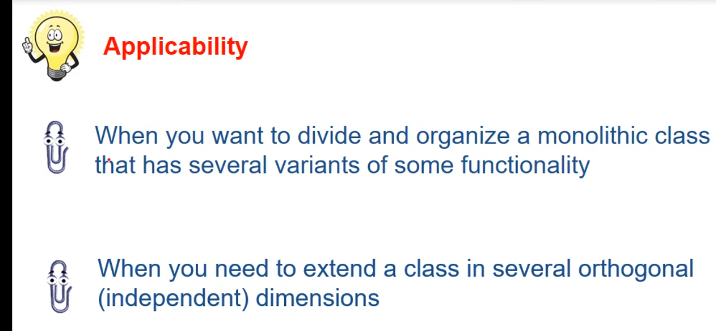
* خليت shape هو base class وبيورث منه square, circle .
* وعملت النحيه التانيه color ووارث منه red, blue
* وكمان خليت جوة ال shape يكون فيه object من الcolor . **ف ال bridge ف حالتنا هنا ان خليت ان خليت ف object من ال color او للدقه reference من color جوة ال shape .**



## 4.2-applicaplty :

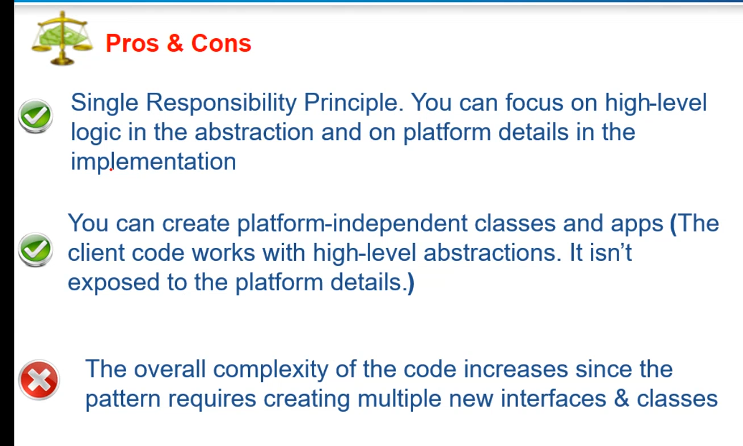
امتى هستخدام ال bridge :

1. لما يكون عندى كلاس كبير ومفيش coherence بين member بتاعته ف هقسم الكلاس دة لكلاسين واعمل بينهم bridge .
2. او لو عندك business فيه اكتر من diminution بس ال diminution دى مختلفين او independent .



ودى نفس الاسباب الى شرحتها .

## 4.3-cons and pons :

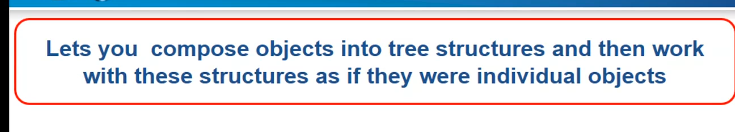
1. Single responsibility : لو اخت بالك ان انا عملت hierarch بتاع ال color لواحدة وال hierarch بتاع ال shape لوحدة ف من هنا طبقنا ال hierarch .
2. عشان ال diminution كل واحد مش معتمد على التانى ف كدة اقدر ابنى مجموعه من ال controller من غير معتمد على ال platform الى هشتغل عليها وقصدى ب ال platform هنا os . بس مكدة هحتاج bridge يربط بين ال controller الى بعملها وال platform . وقلنا ان معنى ال bridge ان هحط object من كلاس معين ف class تانى.

# 5-important notes

دايما هتلاقى ان معظم المشاكل الى هتقبلها عشان:

* عاوز تزود diminution جديد ف هتلاقى غصب عنك ال hierarch.بيزيد بشكل مش منطقى .
* او عاوز تتعامل مع typeانت مش عارف نوعها .
* والكلام دة ينطبق على ال creation pattern وكمان ال structure pattern .

# 6-composite :

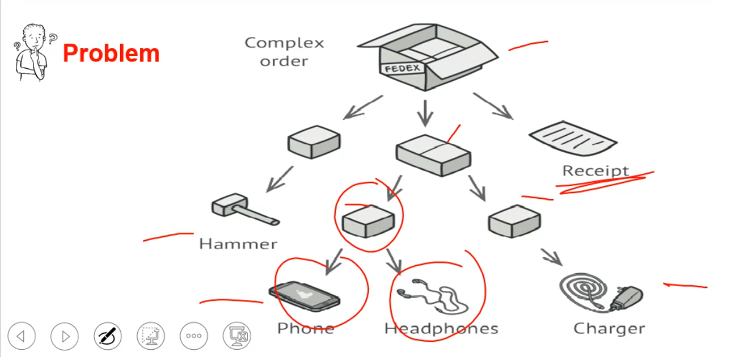


ال composite DP بيخلينى اقدر اعملof object tree او بمعنى تانى . يعنى هبنى structure زى ال tree

والميزه هنا ان :

1. هقدر امسك اى جزء من الtree واتعامل معاه اكنه object واحد. يعنى كل object ف ال tree هيحتوى على مجموعه من ال child object . ف الميزة ان اقدر امسك اىobject بالchild بتوعه واتعامل معاه على انه object واحد .
2. وكمان ال object الى هتكون موجودة ف ال tree ممكن تكون جايه من كلاس مختلفه بس بتعامل معاهم كلهم اكنهم جاين من كلاس واحد .

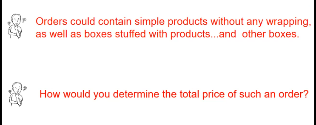
## 6.1-problem :



افرض ان بعمل implement ل سوق . ف اى حد يقدر يعمل order وال order الواحد ممكن يبقا فيه مجموعه من الproducts وكمان ممكن يبقا فيه packages و ال packages دى مكونه من مجموعه من ال الproducts .

وكمان ممكن يبقا معا packages جواها مجموعه من ال packages .

* ف احنا هوزين نحسب ال total بتاع ال order ده . ف محتاج الف واحسب سعر كل الproduct الموجودة . المشكله انا الorder مكون من product وكمان package مش بس products . ف ال composite هيسعدنا ف الموضع دة .



والاسكرين دة نفس الشرح بتاع ال order

## 7.2-calculte total :

عشان احسب السعر بتاع order محتاج :

1. اعمل interface مشترك يندرج تحته ال product وال package ودة عشان اقدر اتعامل مع ال product و packageعلى اننهم حاجة واحدة .

* لو بتسئل نفسك ليه هنعمل common interface تعاله اقلك :

### 7.2.1-why common interface :

احنا قلنا قبل كدة هيكون عندنا package او للتسهيل خلي اسمها box وكمان عندنا product . طيب انا عاوز اشوف الاتنين دول اكنهم واحد بالضبط ف الحل ان اعمل interface . وتعاله نشوف لو معنديش انترفيس الوضع هيكون ازاى .

* هيكون عندنا كلاس product و كمان كلاس box .
* ف احنا عوزين نخلى اى box يحتوى على box وكمان product .
* ف كدة كلاس order يحتوى على box وكمان product
* ف دة هيكون شكل التصميم بتاعى: واضح ان هو معقد وكدة مش بعامل ال product و box على انهم حاحة واحده ف محققتش الى عاوزة .
* ملحوظه بس المكان الى فاضى ف الاسكرين مفروض فيه product .



* طيب تعاله نشوف لما نضيف interface الدنيا هتبقا اوضح ازاى :

1. كل ال هعمله هعمل interface اسمهitem وجواه داله اسمها execute مثلا ودى كل همها انها تحسب الtotal .
2. هعمل 2 class اسمهم product ,box وخليهم يعمله implement للinterface ف كدة هيكون فيهم داله ال execute .
3. هضيف بس list of items للbox يعنى ال box كدة هيكون فيه array من الinterface الى اسمه item .
4. نيجى بقا لداله ال execute دى هتكون ف ال box وكمان ف ال product

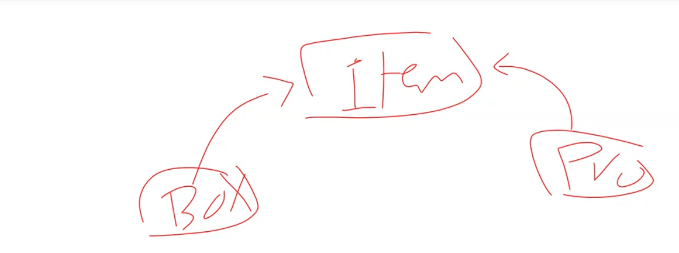
-ف ال product كل الى هتعمله هتضرب عدد ال product الى اشترتهم ف سعر ال product وترجعلك القيمه .

- اما ف ال box هتلف على ال array of item ولو كان ال object الى جوة ال array دى من نوع product

هترجعلك السعر \*العدد الى اشتريته . اما لو ال object هو box يعنى انا كدة كنت بلف على array جوة box

لقيت object form box ف ال array ف كدة الى هيحصل هنادم دالهexecute الى ف كلاس box

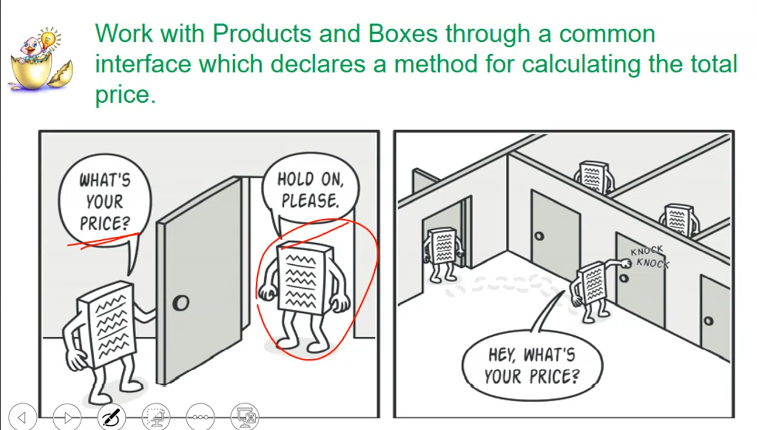
الى كانت بتلف على array of object . اكنى شغال recursion .



* ف الخلاصه الدالهexecute الى جوة product هترجع السعر مضروب ف العدد .
* اما الدالهexecute الى جوة box هترجع هتلف على كل array of item الى جوة ال box ولو ال item كان product هتنادم داله ال execute الى ف product ولو ال item كان box هتنادم دال execute الى ف box .

والاسكرين الى جايه دى بتوضحلك : ان لو الitem كان product كدة الى هيرجعلك القيمه على طول

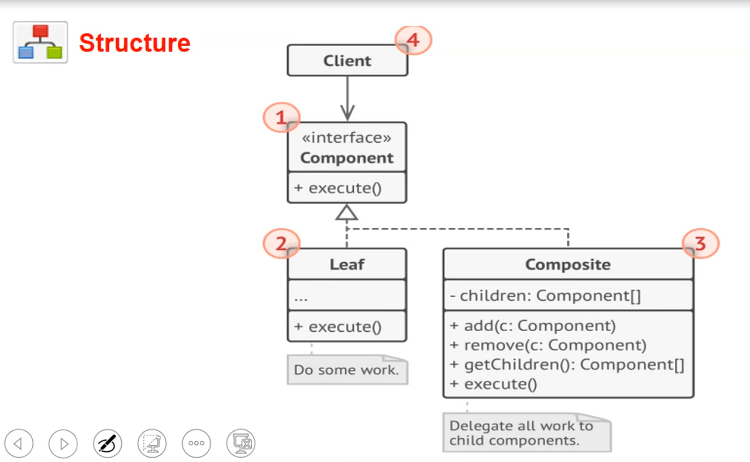
اما لو كان box ف هيروح يلف على الitem الى جواه وهكذا .



## 7.3-structure:

ف الاسكرين دى نفس الشرك الى فوق بس الاختلاف ف الاسماء .

* ال interface الى اسمه component دة هو الitem الى شرحته من شويه وفيه داله ال execute .
* الleaf هو ال product .
* وال composite دة هو ال box . وهتلاقى كام داله ف الكلاس دة عشان يعرف يضبف عناصر ويحزف وكدة .وطبعا هيكون فيه array of item زى مشرحنا قبل كدة .



## 7.4- when can use this pattern:

* الامثله كتير خاااااااالص ودة مثال سريع .
* ف المحاسبه ف حاجة اسمها شجرة الحسبات ودى بتتكون من مجموعه من الحسبات لحاد متوصل لمجموعه من ال leave .

## 7.5-applicapilty :

* زى مشفنا قبل كدة وقلت اكتر من مره .

1. لما اكون عاوز اعمل structure شبه ال tree .
2. لما اكون عاوزالclient يتعامل مع complex type و simple type على انهم حاجة واحدة .

## 7.6-cons and pros :

المميزات والعيوب :

1. بيخلينا نعمل tree like structure
2. Open/close principle مطبق.
3. العيب الوحيد ان ممكن ملاقيش common interface . ف المثال الى فات كان الاتنين جاين من نفس ال domain عشان كدة عرفت اوفر interface يربط بينهم . ف احيان كتير او اغلب الوقت الcommon interface هيكون interface مش

Abstract class عشان اوقات كتير بيكون الاتنين الى هيدخله ف علاقه tree like structure جبين من domain مختلف ف مش هيون ف بينهم علاقه is a . ف عشان كدة ف معظم الاحيان هنعمل ال common interface الى هو component على هيئه interface مش abstract class عشان الconcrete class هتبقا جايه من domain مختلفه .

* ودايما افتكر ان معنى interface مميكن يكون abstract او interface