12/14/2022

Abdo Gamal Ahmed

no campany

DP

Part3



Table of Contents

[1 –solid principle 2](#_Toc122389367)

[2 –single responsibility principle : 2](#_Toc122389368)

[2.1 – loose coupling vs. Tight coupling: 3](#_Toc122389369)

[2.2 –separation of concerns : 3](#_Toc122389370)

[2.3 – cohesion : 3](#_Toc122389371)

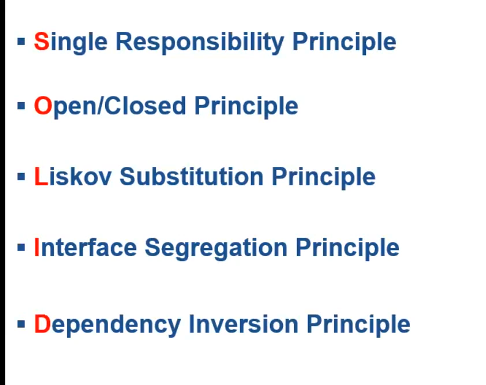
[2.4 – example on single responsibility: 4](#_Toc122389372)

[3- open/closed principle : 5](#_Toc122389373)

[3.1- example1: 6](#_Toc122389374)

[3.1 –how can we predict future change whithout over engineer: 8](#_Toc122389375)

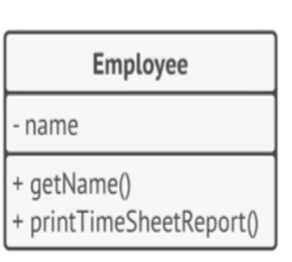
# 1 –solid principle

النهاردة هنتكلم عن كل واحد بالتفصيل من دول :

# 2 –single responsibility principle :

من اسمه كدة معناه كل واحد يخليه ف ورقته , كل واحد يعمل حاجة واحدة بس . ودة على مستوى ال class و ال function .

وعمنا بوب بيقول : "a class should have just one reason to change " يعنى الclass الواحد لازم يكون عندة سبب واحد بس عشان يتغير السبب دة يبقا مرتبط ب ال business logic بتاع ال class . بمعنى لو عندى class employee وبعد شويه قلولك عوزين نزود property اسمها addressمكنتش موجوة جوة ال class قبل كدة . ف دى حاجة تخلينا نفتح الكلاس ونغير فيه .

* ف لو انت خليت الclass من الاول responsibility لمجموعه حاجة متخصهوش اصلا ف كدة انت معرض انك تفتح الكلاس اكتر من مرة عشان تعدل فيه لو فيه حاجة جديدة اضافت .
* ****ف كل مكان الكلاس مسئول عن حجات اقل مجرد الحجات الى تخصه ف كدة نسبه التعديل فيه هتقل .

**الهدف من ال principle دة هو reduce complexity**

**تعاله ناخد مثال :**

ف الclass دة هل مطبق مبداء ال single responsibility ؟ لا

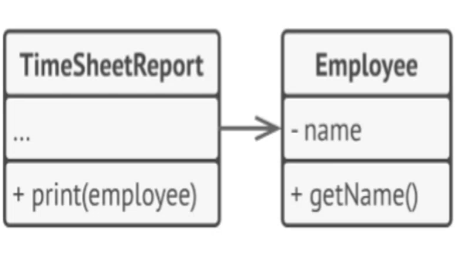
والسبب لو مشيت على كل الحجات الى جوة هتلاقى

* ال name ده هتلاقيه دة من الحجات الاساسيه بتاعه ال emp
* ال get name هتلاقيه دة من الحجات الاساسيه بتاعه ال emp
* اما ال print دى ملهاش علاقه ب ال . emp ممكن تقلى انك بتطبع الداتا بتاعه ال employee

هقلك تمم بس ف فرق بين انك تطبع الداتا بتاعه ال emp وان ال action يكون

بيعمله ال emp يعنى احنا هنا بنتكلم هل الطباعه دى من action الى بيعمها ال emp class ؟

اكيد لا

* ف لو ال print دى موجوة جوة ال employee كل متغير حاجة ف نوع الطباعه هتفتح ال emp وتغير فيه .وحتى لازم الجمله الى قلتها "**كل مغير حاجة ف الطباعه “** ف التغير ملهوش علاقه ب ال emp .
* ف الخلاصه ان ال emp مش مسئول عن ال action ده ولازم الاaction يطلع برة ال employee .

نرجع ناكد على شويه مفاهيم من خلال ال single responsibility بقدر احققهم وهم :

* loose coupling vs. Tight coupling
* separation of concerns
* cohesion

## 2.1 – loose coupling vs. Tight coupling:

احنا هدفنا دايما نعمل تصميم فيه loose coupling يعنى ارطبات الdata او ال class مش مرتبطه ببعض او مش معتمدة على بعض .ودة حققناه لما عملنا interface layer . وزى مقلنا كل مكانت الclass معتمدة على بعض بشكل كبير دة هيوصلنا لصعوبه التغير لو حبيت اغير عشان لما اغير ف class معين هغير برضو ف باقى الclass الى معتمدة عليه .

## 2.2 –separation of concerns :

ودة معناه كل واحد يبص ف وراقته وكل واحد يعمل حاجته .يعنى ف المثال الى فات هفصل ال responsibility بتاعه ال emp عن ال responsibility بتاع الطباعه .

فدايما هنعمل separation of concerns عشان كل كلاس يكون مسئول عن الحاجة بتاعته بس .

## 2.3 – cohesion :

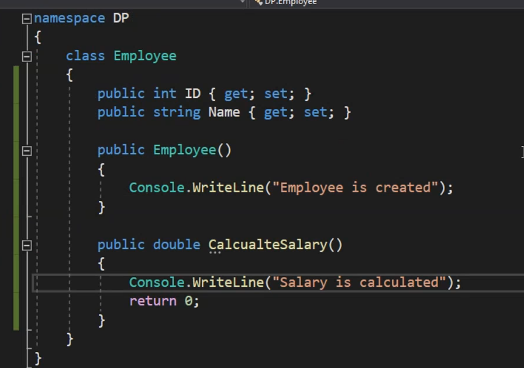
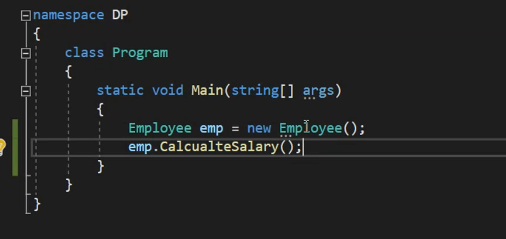
يعنى كميه الترابط بين ال member الى موجوة جوه الكلاس. بمعنى لازم ال member الى جوة الكلاس يكون فيها بينهم علاقه او صله بشكل او باخر .يعنى هتلاقى ال function الى جوة الكلاس بتشتغل على ال member variable الى جوة ال class .

لو لقيت مثلا عندى كام function بيشتغله على 2 او 3 من ال attribute الى عندى وسيبين الباقى .و2 او 3 من ال function التانين بيشتغله على 2 او 3 من ال attribute الى عندى وسيبين الباقى . والكلام دة بيتكرر معاك ف كل ال function الموجودة جوة ال class . ف هنا لازم ال function دول يتفصله ف كلاس لوحهم .

***تعاله ناخد مثال ف ال vs يحب .***

## 2.4 – example on single responsibility:

- هنفتح.net 5 console app.

دة الكلاس الى عملته و ال main :

السوال بقا : هل الشكل الى عملته دة مطبق ال single responsibility ؟

* ممكن تقلى لا عشان داله ال salary دى مفروض تكون برة الemp عشان مش من ال business logic بتاعه ؟ هقلك لا غلط طلام ال salary مش بيتاثر بحاجة خارجيه ممكن تخليك تغير فيه كل ما الموثر دة يتغير وطلام ال salary بيعتمد على شويه attribute جوة ال emp يبقا خلاص من اختصاص ال emp .
* ولاحظ هنا ان دة مجرد مثال عشان كدة الكلاس مش كامل ف متقليش ان الfunction مش بتتشتغل على ال attribute .
* وهزيك من الشعر بيت لو عندى firtName attribute وكمان lastName attribute وعاوز اجيب ال fullname يبقا برضو هعمل ال function بتاعه ال concatenate جوة class emp مع ان ال attribute بتاع ال fullName مش موجود جوة الكلاس . **ف القاعدة طلام الحاجة الى بعمل process ليها مرتبطه ب ال business logic بتاع employee ف خلاص يبقا تفضل فيه فل .**

طيب هتقلى يبقا الشكل الى فات بيطبق ال single responsibility ؟ هقلك برضو لا مش بيطبق المبداء. لو فاكر ف كورس فات قلتلك ان الكلاس مش مفروض يكون جواه اى input او output طيب ليه ؟ عشان بعد شويه لو حبيت تستخدام ال class emp ف app تانى هل هينفع ؟ اكيد لا عشان الapp التانى ممكن يكون web app وال app الى فيه ال emp دة مثلا console app ؟ ف مش هينفع موضع الطباعه الى انت عامله ف الكود فوق دة . ف طريقه ال logging الى انت بتعملها مش هتكون مناسبه لكل الapp .

* **وكمان مش منطقى تعمل** logging **جوة ال class emp هو مش مسئول عن ال** logging **اصلا .**

**ف من الاخر عشان تعرف القصه دى اسئل نفسك سوال :** لو عمليته الطباعه دى اتغيرت هل هتحتاج تفتح كلاس emp وتغير فيه يعنى مثلا لو قلتلك احنا مش عوزين نعلم logging على ال console app وقلتلك احنا هنعمل logging جوه json file هو ف databse سعتها هتفتح كلاس emp وتغير فيه ولا لا ؟اكيد ايوة هتغير فيه

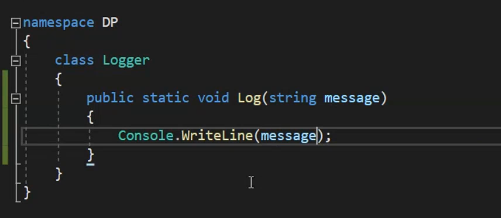
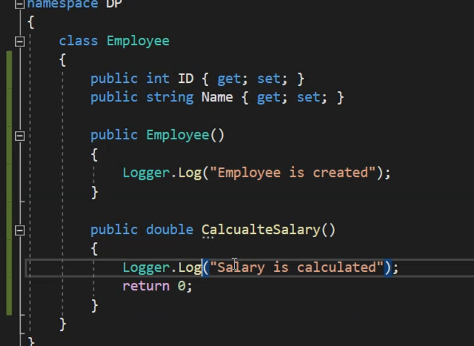
* طيب لو عندى كلاس emp وكمان dept وعاوز تغير غير طريقه ال log ف هتغير ف كام حته لو فرضنا ان بعمل log برضو ف ال dept ؟ اكيد هحتاج اغير ف كل الكلاس الى عامل فيها logging طيب ليه عشان مكتبتش الحاجة ف مكانها المظبوط .

**طيب الحل :** separation of concerns . يعنى مفروض نعمل class هو المسئول عن عمليه ال login .

* ولو انت مش عارف معنى logging او ايه لزمته . فمعنى logging ان اسجل وقت لكل الaction الى user هيعملها . يعنى مثلا اسجل ان ال user عمل registration ف الوقت كذا . ودة على كرة مطبق ف كل الapp الى ف ارض الواقع مش الكحيانه الى انت بتعملها. وفايه ال logging ان لو حصلت مشكله مثلا اقدر اعرف تسلسل الاحداث حصل ازاى لحاد ما المشكله حصلت .

**دة شكل التصميم الجديد يخوياا :**

* عملت class اسمه logger يكون lمسئول عن ال logging وكمان عدلت ف class emp وبد ال ما بطبع نديت داله ال logging الى ف كلاسlogger

****

* **طيب السوال امهم هنا** لو بعد شويه طريقه ال logging اتغيرت هل هحتاج اغير حاجة ف ال emp ؟لا
* كل التغير هيحصل ف ال logger ودة منطقى عشان اهو الى مسئول عن التغير .ف بدالم مثلا مكنت بطبع على الشاشه هروح احفظ ف database ومن غير مغير ف اى كلاس بيعمل logging .

ف خلى كل كلاس مسئول عن حاجته بس وملهوش دعوة بحاجة غيرة .انا مجرد هستغين بحاجة الناس التانيه مش هعملهم شغلهم . يعنىطبق دايما مبداء ال single responsibility عشان هو مهم واديك شوفت الفايدة .

# 3- open/closed principle :

دة تانى بداء واهم مبداء هنقابله .

**Class should be open for extension but closed for modification**

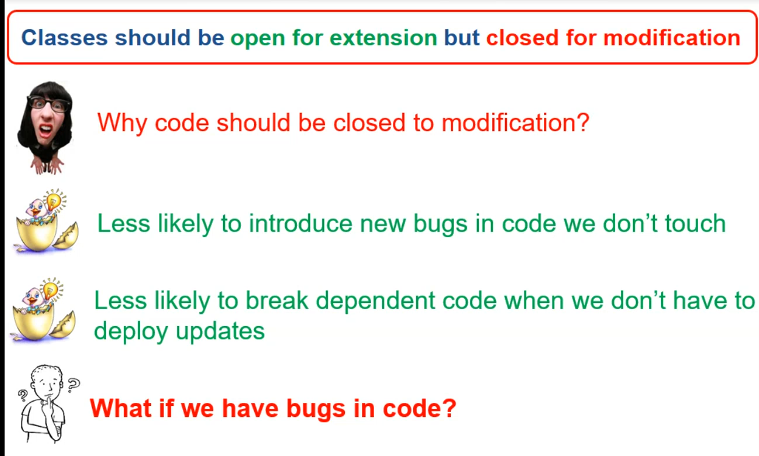
ودة بيقلك ازاى اعمل design لل app بتاعى بحيث : اقدر اضيف future جديدة ف ال system من غير ما افتح الكود وغير فيه .

**طيب هو ليه ف مشكله لما افتح الكود واعدل فيه ؟**

1. عشان ف مبداء بيقلك طول م الحاجة شغاله متعدلش فيها **. عشان لما تفتح الكود بتاعك وتعدل فيه مليون ف الميه هيطلع bug جديد** عشان وانت بتصمم لكود لو قديم مش هتكون فاكر ال business وكمان لو انت مش عامل clean code مش هتفهم الكود . عشان كدة بنقول متفتحش كود قديم وتعدل فيه .
2. ان ف ناس معتمدى على الحجات الى كنت كتبها زمان . بمعنى ان انت دلوقتى كتبت كلاس وحتيطه ف library وبداء ناس تانى ف التيم تستخدامه ف ازاى تيجى وتغير فيه ! ف ناس معتمده على ال function الى انت كنت عملها .

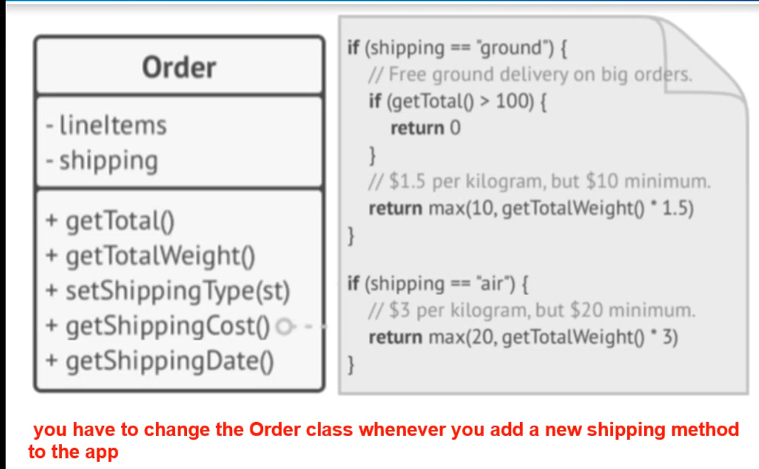
وعشان كدة الشركات المحترمه زى MS بتعمل حاجة ايمها countable mode يعنى ازاى اطلع update جدية وخلى ال app القديم بتاعك لسه شغال .تخيل مثلا كل مشركه تطلع os جديد تلغى القديم وتقلك والله لازم تستخدام الجديد .ف مش هتلاقى حد يشترى الشركه دى .

1. بس دة مش معناه ان لو لقيت bug انك تسيبه ومتصلحوش . ف الحاله دى عادى افتح الكود وصلح ال bug

ودة الشرح الى فات بس بالانجليزى :

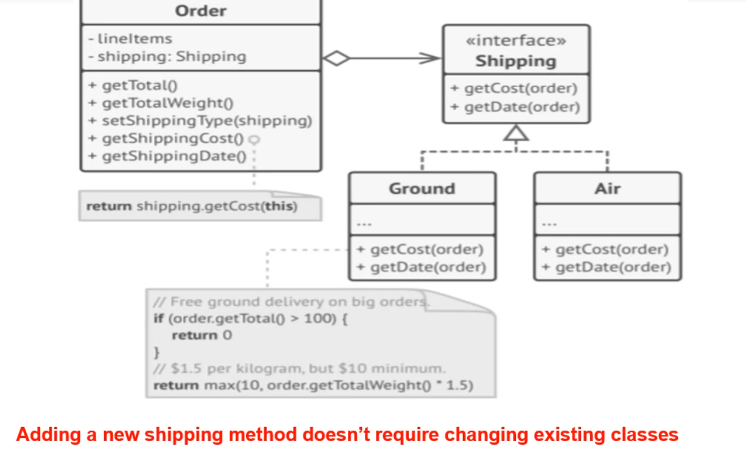
**وتعاله ناخد امثله .**

## 3.1- example1:

**** دة مثال شبه الorder tax الى شفناه قبل كدة . افرض انك ف موقع سوق مثلا . ومعايه class order . وهنهتم بداله الى بتحسب التلفه النقل shippingCost .

* دة نفس الحوار بتاع مبداء ال encapsulate what vars : كل متتغير طريقه shipping للاسف طريقه الحساب هتتغير . ف الافضل ف الحاله دى ان نعتمد علىabstraction .

1. ف هنعمل interface layer نحط فيها ال function بتاعه ال calculation بتاعه ال shippingCost واسمهshipping
2. جوة كلاس order هيكون معانا object من الshipping او reference for class implement shipping
3. وهنعمل 2 class يورثه من ال shipping وليه 2 class عشان لحاد دلوقتى عندى حالتين بس ف الكود القيم لطرق النقل
4. وجوة ال order هخلى داله getShipingCost تنادم داله ال getcost الى جوة ال shipping .
5. وطبعا احنا هنبعت ال objet للorder class حسب م احنا نحب لو طريقه النقل دى عن طريق البر ف هنبعت object من ground وهكذا
6. ولو ف طريقه جديدة هتضاف للشحن بسيطه كدة هنعمل class ونخليه يورث او يعمل implement من ال shipping .ونبعت object منه لل order عادى خاالص من غير منفتح ونعدل ف اى كود قديم .

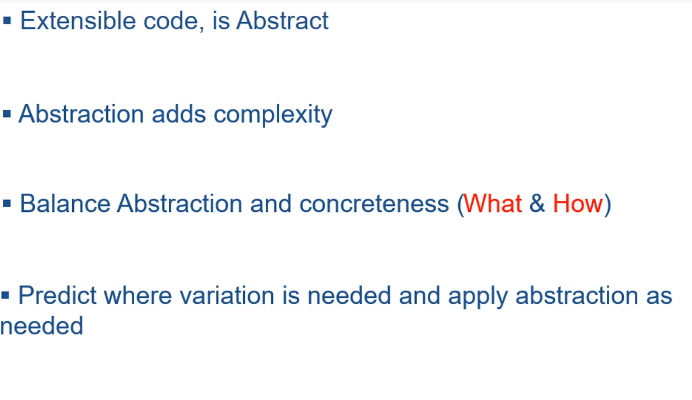


* **ف كدة انا كصابح كلاس order هعمل كلاس order مرة واحدة بس وهقول لاى حد عاوز يشتغل معايه ويضيف طريقه للنقل تانى يروح يعمل implement او inherit لل shipping ويبعت object للorder .** والطريقه دى لو تفتكر زى زمان لما اتكلمنا على ال function sort وقلنا ان function sort ريحت دماغها عشان مش كل ميجلها data type مختلف تقول انا هقارن م بينها ازاى . راحت حاطه abstraction بينها وبين الclient وقالت لو عاوز تشتغل معايه روح implement ال interface الى اسمه Icomparable .

"وقصدى ب ال client الى هو احنا ال developer الى هنشتغل معاه ".

**كدة خلصنا المثال تعاله نقول كام ملحوظه على الماشى كدة .**

1. لو عاوز تعمل كود extendable يبقا هتتجه نحيه ال abstraction .
2. خلى بالك لما تتجه نحيه ال abstraction الكود بتاعك هيكون معقد اكتر .
3. ف بعض الاحيان الكتير موضع انك تتجه ناحيه ال abstraction دة بيكون مقبلول عشان بيساعدنى ف حل مشكله كبيرة عندى مقابل شويه complexity . بس لازم برضو توازن بين ال abstraction وال concreteness ف انت بتفضل ماشى فموضع ال abstraction لبعض الوقت عشان توصل ال what بس بعد شويه محتاج توصف ال how تعمل implement بقا . ف انت ف مرحله معينه هتعتمد على concrete class مش abstraction . ف التوازن بين الاتنين هو الحل . ان يكون معاك تصميم fixable وف نفس الوقت ميسببلك over engineer .
4. طيب بالنسبه لل open closed principle اعمل ايه فيه : **هنشوف التغير ممكن يحلص فين وبداء طبق ال abstraction** . طيب هو انا من الاول اعرف ازاى ان ف تغيرات هتحصل وعشان منروحش ف موضع ال engineer over ونقد نفرض افترضات عشان لو فضلت ف الاول افضل حجات مش هخلص والتصميم هيكون كبير .ف الحل ؟ تابع معايه خطوات تطبيق ال open/closed principle .

دى صورة للاربع خطوات الى فاته بالانجليزى :

واخر حاجة كنت بتكلم فيها هو ازاى اعرف ان هيكون ف تغير

عشان اعرف اطبق ال open closed principle من

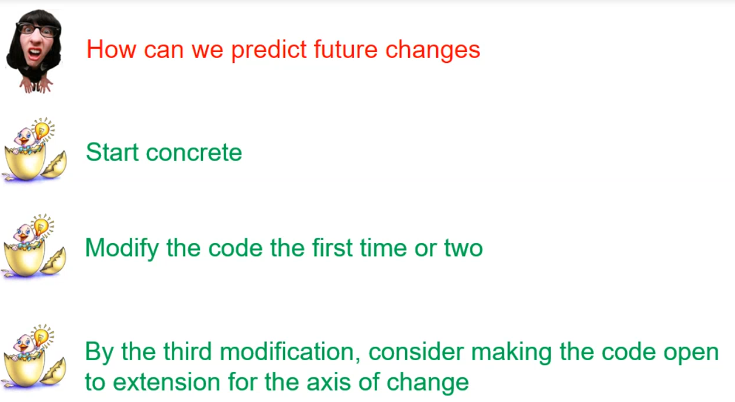
غير ميحصل معايه engineer over .

**ف الحل هتمشى ورى الخطوات الى جايه عشان تعرف فيه**

**تغير ولا لا :**

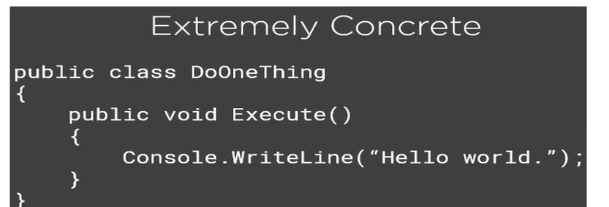
## 3.2 –how can we predict future change without over engineer:

1. ابداء ف الاول متعملش اى نوع من انواع ال abstraction . يعنى ابداء concrete يعنى لو على مستوى ال shipping اعمل code بif conduction عادى . دة ف الاول بس عشان متعملش over engineer
2. ابداء تابع ال changes الى بتحصل . لو لقيت انك غيرت ف الموضع مرة او اتيتن بس كمل concrete ليه عشان متزودش ال complexity متزدش على الفاضى . عشان ممكن ف الاخر تلاقيك عامل طيارة عشان تروح بيها الجامعه ملهاش لزمه .
3. لما تلاقى التغير بداء يزيد شويه . يبقا لا لازم تبداء تفكر ازاى تعدل الكلام دة وتتجه لل abstraction .

وده نفس التلت خطوات الى فوق بس بالانجليزى .

## 3.3 – typical approaches to OCP :

اواى بقا نطبق ال principle بتاع ال open closed ده او ايه ال approaches الى ممكن امشى فيها عشان اطبق ال OCP .

1. Parameters :