9/1/2022

Abdurrahman Gamal Ahmed Gaber

no campany

Clean code

**Ch3 Functions**



Table of Contents

[1-Small : 3](#_Toc124441476)

[1.1-Blocks and Indenting : 4](#_Toc124441477)

[2-Do One Thing: 4](#_Toc124441478)

[2.1- Sections within Functions: 6](#_Toc124441479)

[3-One Level of Abstraction per Function : 6](#_Toc124441480)

[3.1- Reading Code from Top to Bottom: The Stepdown Rule: 7](#_Toc124441481)

[4- Sections within Functions: 9](#_Toc124441482)

[5- Use Descriptive Names: 10](#_Toc124441483)

[6- Function Arguments: 10](#_Toc124441484)

[6.1- Common Monadic Forms : 11](#_Toc124441485)

[6.2-Flag Arguments : 12](#_Toc124441486)

[6.3 - Dyadic Functions: 12](#_Toc124441487)

[6.4- Triads: 13](#_Toc124441488)

[6.5- Argument Objects: 14](#_Toc124441489)

[6.6- Argument Lists: 14](#_Toc124441490)

[6.7- Verbs and Keywords: 14](#_Toc124441491)

[7- Have No Side Effects : 14](#_Toc124441492)

[7.1- Output Arguments: 15](#_Toc124441493)

[8- Command Query Separation : 16](#_Toc124441494)

[9-Prefer Exceptions to Returning Error Codes: 17](#_Toc124441495)

[9.1- Extract Try/Catch Blocks : 17](#_Toc124441496)

[9.2- Error Handling Is One Thing : 19](#_Toc124441497)

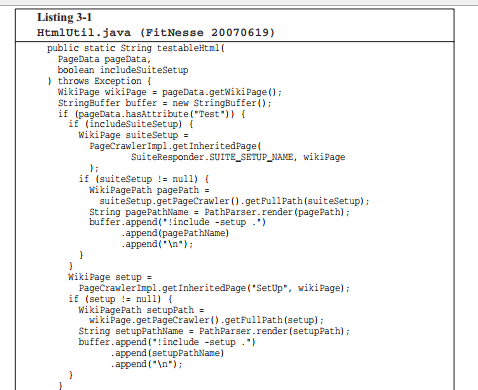
[9.3- The Error.java Dependency Magnet : 19](#_Toc124441498)

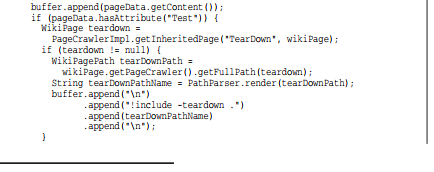
[10- Don’t Repeat Yourself : 19](#_Toc124441499)

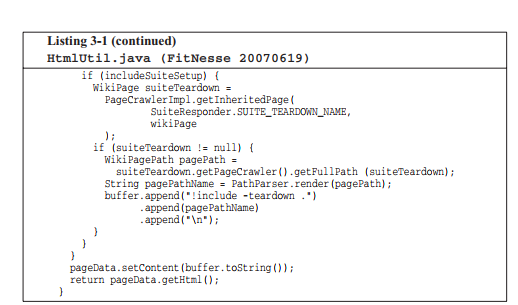
[11- Structured Programming : 20](#_Toc124441500)

الشبتر دة كان مليان معلومات الصراحه . وهو بيتكلم عن ال function وال function دى مهمه جداا عشان بتعتبر من بدهيات التنظيم لاى برنامج

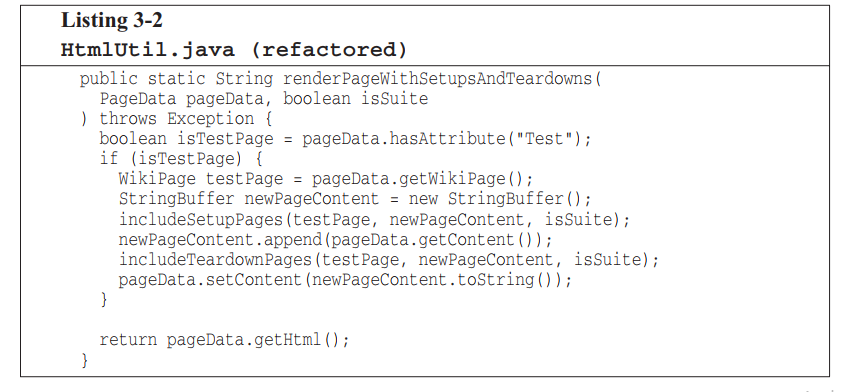
عمنا بوب من البدايه كدة عشان يفهمنا المشكله عطانى اتنين فانكشن عشان نقارن بينهم :دى اول فانكشن







الاسكرين دى باقى الفانكشن الى فوق مش واحده تان

وطبعا باين هى كبيرة قد ايه وبيتحداك انك تفهمها ف تلت دقايق . وحول الفانكسن دى كلها لواحدة تانى كل الى فيها 9 سطور .ودة الشكل للفانكشن بعد التعديل والتحسين

* طبعا بيان الفرق الشاسع بين الاتنين الاوله اصلا مش هحاول افهمها اما التانيه دى لو عرفت السياق الى اتكتبت فيه هفهما بسهوله عادى :والفانكشن الاوله الى مكتوب عليها 3-1 دىكانت معقده بسبب ان فيه level of abstraction ومعنى abstraction هنا ف الشبتر مستوى التعقيد او مستوى التفاصيل الى ف الكود وهنتكلم تانى عن الموضع دة باتفصيل .
* طيب ايه الى خلى الفانكسن التانيه اسهل ف القرايه والفهم ؟

# 1-Small :

القاعدة الاوله عشان اخلى ال function اسهل ف الفهم : انك تخليها small

القاعه التانيه انك تخليها : Small اكتر كمان

وبيقلك ان مفيش قاعدة معينه ممكن تثبتلك ان كل ماكانت ال function اصغر كل مكان احسن . بس على مدار 40 كتب فيهم function كبيرة ممكن توصل ل 3000 سطر ومتوسطه حوال 300 سطر وصغيرة ف حدود 30,20 سطر . والى عمنا بوب اكتسبه من كل التجارب دى ان ال function لازم لازم تبقا صغيرة قدر الامكان .

و فى ال 80s المبرمجين التفقه ان الحجم المناسب لل function هو بحجم الشاشه يعنى اقدر اشوف كل ال function من غير معمل scroll .

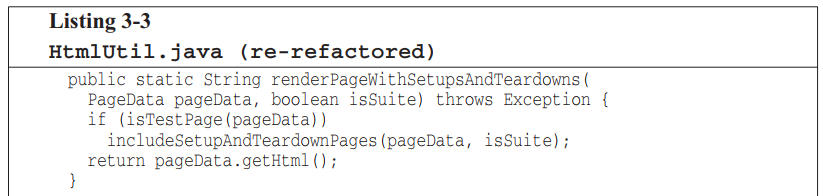
بس الكلام دة طبعا زمان يعنى كان حجم الشاشه ف الوقت دة يشيل 24 سطر اما دلوقتى حجم الشاشه ممكن تشيل 100 سطر ف القاعه بتاعه ال 80s مش صح خالص نطبقها دلوقتى .

وعمنا بوب بيحكلنا ان لما كان ف زياره لصاحبه ف شاف عنده برنامج لما يحرك الموس بينزل نجوم ف البرنامج دة كل ال function الى فيه كان حجمها بين سطرين او تلاته باكتير اوى وكل function لتنقلك لل function الى بعدها .

طيب طول ال function مفروض يبقا قد ايه؟ بيقلك شايف ال function رقم 2-3 ال function بتاعتك لازم تبقا اصغر منها ودة شكل ال function الى عمنا بوب عاوزك توصله.

* وملحوظه 2-3 دة رقم موجود على الاسكرين الى سبتهالك من شويه .
* وكمان الاسكرين الى جايه هى طلعت لما عملنا refactoring لتانى اسكرين الى رقمها 3-2 والاسكرين الى رقمها 3-2 دى ناتجه لما عملنا refactoring الاسكرين رقم 1-3.

ف بختصار الاسكرين رقم 3-3 دى طلعت لما عملنا refactoring للاسكرين رقم 1-3. وطبعا باين الفرق الشاسع ف الحجم .



ممكن تكون بتسئل نفسك هو وصل للشكل دة ازاى ف هو موضحش وساب الاسكرين كدة وبعد شويه هيقلك عمل كدة ازاى كنوع من انواع التشويث يعنى .

## 1.1-Blocks and Indenting :

معنى ال Blocks الى هنا الى هو body بتاع ال if, loop ,switch كدة يعنى .

ف عمنا بوب بيقلك ال Blocks مفروض يبقا سطر واحد بس ومش بس كدة دة السطر دة مفروض يبقا function وفايده ان ال body او ال Blocks يبقا function ان كدة الاسم هيبقا معبر عن الغرض وبالتالى هتفهم الكود بسهوله .

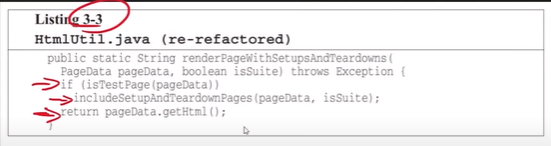
ومفروض تكون فهمت ان مينفعش يكون حجم ال function كبير لدرجه ان يكون فيه nested structures .

# 2-Do One Thing:

بيقلك ان ال function رقم 3-1 بتعمل حجات كتير خالص ودة من اسباب انها معقده وكبيرة .

وال function رقم 3-3 والى كانت صغيرة خالص وواضحه دى كانت بتعمل حاجة واحدة بس والى هى اسم الfunction .

**ف ال function مفروض تعمل حاجة واحدة وتكون بتعملها صح وتعملها هى بس .** ومشكله النصيحه دى انك تعرف ايه الحاجة الواحدة الى الfunction مفروض تعملها .

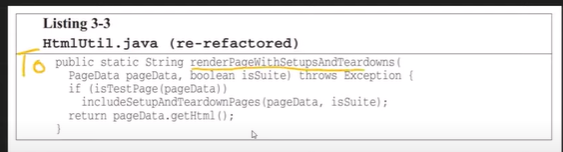
 طيب تفتكر ان الfunction رقم 3-3 بتعمل حاجة واحدة بس؟

لو جينا شفنا ال function هنلاقيها بتحدد

* هل ال pageData هى testPage ولا لا
* لو هى test هتعمل include
* وبعد كدة بتعمل render لصفحه ال html .

طيب كدة افهم انها بتعمل حاجة واحدة ولا لا ؟ الfunction دى بتعمل التلت حجات الى جواها وهم من نفس level ال abstraction يعنى اقدر اوصف ال function دى على انها paragraph . واول ال paragraph يبداء ب to يعنى هضيف to قبل اسم ال function واكمل شرح ليها لو لقيت ان عادى الموضع ماشى بتسلسل ف كدة معناه ان ال function دى بتعمل حاجة واحده .

* تعاله نوصف ال function رقم 3-3



س

**To** renderPageWithSetupsAndTeardowns we check if the pageData is test page and if so we includeSetupAndTeardownPages and in all case we render datapase .

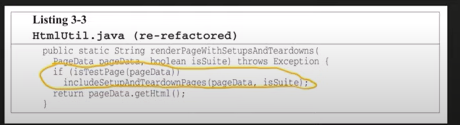
ف انا وصفت الfunction كلها على اكنها one paragraph . ف كدة ال function دى فيها level واحد من ال abstraction وكده هى بتعمل حاجة واحدة بس . والموضع مهم ومن اكتر لمواضيع الى عجبتنى .

**وفى العموما السبب الى بيخلينا نتكتب function اصلا هو اننا بنحلل او بنصغر concept كبير لمجموعه خطوات ف level الabstraction الى بعده او لمجموعه خطوات اقل .**

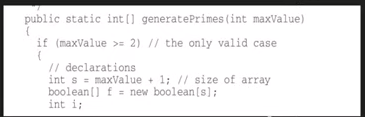
وجه قلك ان صورة 3-1 دى كبيرة عشان فيها اكتر من level of abstraction وصورة 3-2 دى برضو فيها اكتر من level of abstraction والدليل ان قدرنا نعملها refactoring ونطلع منها ب الصورة رقم 3-3.

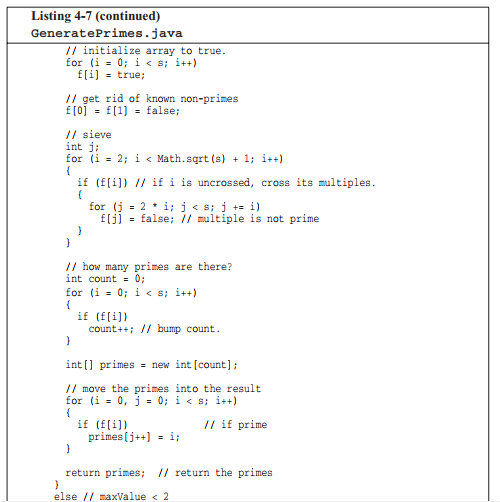
ف عمنا بوب حط قاعدة تانى عشان تعرف ال function دى بتعمل حاجة واحدة وله لا: **قلك لو قدرت تجزء ال function او تصغرها من غير ميبقا الى انت بتعمله اعادة تسميه بس يبقا هى مش بتعمل حاجة واحدة .**

* دى ال function رقم 3-3 واحنا طبقنا عليها قاعدى to وكمان لو حابين نطبق عليها اخر قاعدة هنلاقى منقدرش نصغرها عن كدة ولو قلتلى ممكن احط ال if ف function لوحده هقلك الى بتعمله ملهوش لازم ولا هيفرق اى حاجة ف ال readability .



## 2.1- Sections within Functions:

لو ال function مقسمه لاكتر من section ف كدة ال function دى مش بتعمل حاجة واحده . الاسكرين دى كلها function واحدة والسكشن هنا قصده بيه كلمه declaration و initialize array to true الخ ف هو بيقلك الfunction دى مفروض يحصلها refactoring وتبقا بتعمل حاجة واحدة بس .

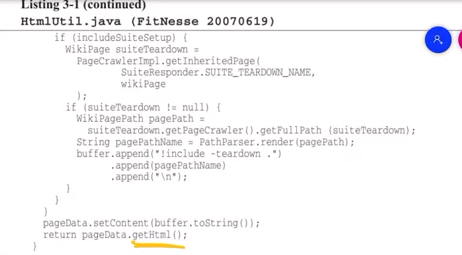


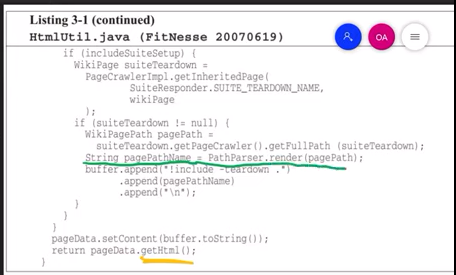
# 3-One Level of Abstraction per Function :

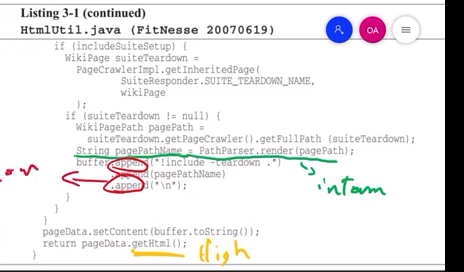
السكشن ده بيتكلم عن ان لاوزم ال function يكون فيها level واحد من abstraction .

وبيوضحلنا كام مثال على موضع ال abstraction من الصورة رقم 3-1 اول صورة عندنا .

* اول حاجة بيقلنا عليها هى ان getHtml دة يعتبر high level of abstraction .



* والجزء دة الى معلم بالخضر دة يعتبر intermediate level of abstraction .
* والجزء الى بالاحمر دة low level of abstraction .

وطبعا ال low level دة واضح ان فيه تفاصيل كتيره.

ف كل ما level ال abstraction قل كل ما التفصيل

بانت.

ف عمنا بوب بيقلك لو لقيت حاجة زى كدة ف دة عك

ومتعملهوش .

## 3.1- Reading Code from Top to Bottom: The Stepdown Rule:

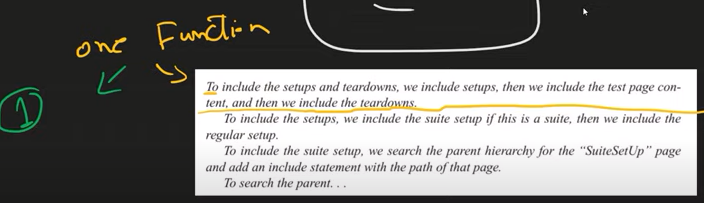
ف السكشن دة عم بوب بيقلنا احنا عوزين الكود يبقا كانه روايه يتقراء من فوق لتحت يعنى اكنك بتقره جرنان او روايه .

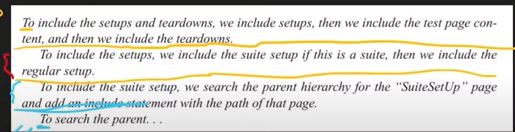
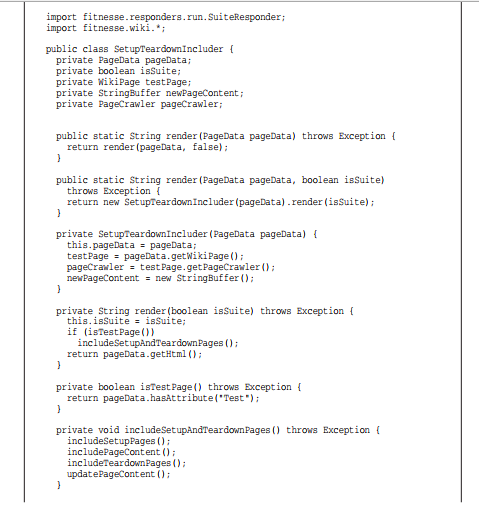
وكل function تكون ال function الى بعدها اقل ف level abstraction . يعنى كل مانزل لتحت ف الكود انقص level abstraction بمقدار واحد بس.

ف الطريقه دى اسمها Stepdown Rule : وبختصار احنا عوزين نحول الكود لparagraphs بدايه كل paragraph كلمه

to وشرحت موضع ال to من شويه .

* وعم بوب بيديك مثال على تحويل الكود لparagraphs وبيقلك ان الكلام الى فوق الخط الاصفر دة بيعبر عن one function وبيكون جوها level واحد من ال abstraction ومينفعش يكون ف اكتر من level .

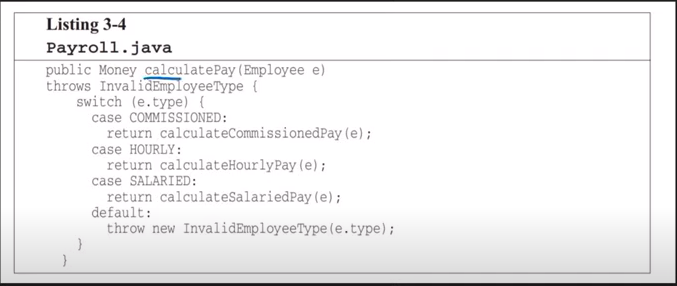


* ف كدة كل function هنمثلها ب paragraph واوله كلمه to ف الاسكرين دى فيها4 paragraph يعنى بتوصف 4 function
* واكيد اموضوع انك تعمل function جواه level واحد بس من ال abstraction دة صعب بس انك تحاول تطبقه هيفيدك . وهو مفتحاح انك تخلى ال function صغيرة وبتعمل حاجة واحدة بس.
* ودة كلاس تطبيق على القاعده . والموضع واضح حتى بالنسبالى انا وانا مش عارف logic وره الكلاس ده .بس انت كل معينك تقع على function تحس انك عارف او فاهم الfunction وحتى مش متعقد منها .
* ف اكيد الوضوع واضح او ازاى كويس وبالاخص لو طبقت قاعه ال to .

# 4- Sections within Functions:

من الصعب قوى انك تعمل switch صغيرة . وهنا لو switch فيها two case بس ف يعتبر كبيرة . وطبعا مينفعش اعمل switch فيها one case عشان ال switch اصلا معموله عشان يبقا فيها اكتر من case .

* ف بكل بساطه نقدر نستخدام ال polymorphism عشان نعمل ال switch يعنى بدال معمل switch هنعمل polymorphism .
* ودة مثال على ازاى استخدام ال polymorphism بدال ال switch وكنت شارح نفس الموضع ف ال design pattern :



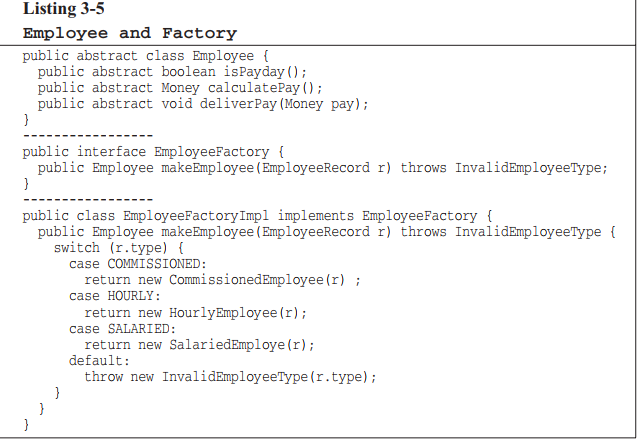
اول حاجة تعاله نشوف مشاكل ال function الى معظمها جاى عشان فيه switch :

1. اول حاجة هى كبيرة
2. لما نضيف نوع موظف جديد ف مفروض ال function دى هفتحها واعدل فيها ف بالتالى هتكبر اكتر .
3. والجزء دة كلامى انا عبد الرحمن : ال function دى بتنتهك مبداء ال principle single responsibility وشرحت الموضع دة ف جزء ال design pattern .
4. وكمان ال function دى بتنتهك مبداء ال open/closed principle عشان لو حصل اى تغير ف الbussines logic هفتح ال function دى واعدل فهيا . ودة كلام عم بوب
5. واسواء عيب ممكن يكون موجود ان يكون ف اكتر من function بنفس الشكل ف العيب دة هو اسواء من كل الى فاته .

ف ممكن يكون فيه function تانيه ف ال system دة اسمها isPay(employee e) و display(employee e) ف بيقلك ال structure ده ضار جداااااا جدااااااااااااااااااااا "لو مش وصلك ليه هو ضار ف شوف شرح DP الى انا عامله"

طيب الحل ايه :

* القاعده الى هنمشى عليها هى لو ال switch هتظهر مرة واحدة بس فى الsystem ف ممكن نتقبلها عادى غير كدة لا .
* وعم بوب بيقول ان هو نفسه سعات مش بيطبقع قاعدة الswitch يعنى كل حاله وليها ظروفها .
* والاسكرين دى بتوضح الحل بس عم بوب مشرحش الحل بس هو استخدام abstract factory وده DP انا شارحه وشارح نفس المثال تقريبا ف ارجعله احسن ف تلخيص ال DP .



# 5- Use Descriptive Names:

دايما لازم تدى لاختيار الاسماء حقه يعنى متستعجلش وانت بتختار الاسماء ف دة مش تضيع وقت .

ودايما افتكر مبداء واحد اسمه word :" انت بتعرف ان الكود بتاعك clean لما كل حجة بتعملها بتطلع زى ما انت عايز ف دة موشر انك بتكتب ب clean code ف نص طريقه عشان تحقق مبداء word انك تعمل function صغيرة واسمها كويس وبتعمل حاجة واحدة

وكل ماكانت ال function صغيرة كل مكان سهل اختيار اسم معبر ليها ومتخفش من اختيار اسم طويل معبر للfunction والاسم الطويل المعبر احسن من الاسم القصير الى مش معبر او اسم طويل معبر احسن من comment طويل معبر واستخدام ال name convention وقصدى ب ال name convention حاجة زى ال camel case "

* نرجع لعم بوب بيقلك متخفش تاخد واقت ف اختيار الاسم وممكن تجرب اكتر من اسم وتجرب تقره الكود بلاسم لحاد متوصل لاحسن اسم معبر للfunction
* وطبيعى وانت بتبحث عن اسم معبر دى ممكن يادى الى اعاده هيكله الكود .
* خلى اسماءك ثابته يعنى استخدام نفس المصطلحات او نفس ال pattern . يعنى خلى ليك style ف ال naming .

# 6- Function Arguments:

احسن عدد Arguments الى هتخدها function هو صفر يعنى تكون ال مبتخداش Arguments .

* تانى افضل حاجة هى انك تخلى ال function تاخد one Arguments
* تالت افضل حاجة هى انك تخلى ال function تاخد two Arguments
* ومفروض تتجنب قدر المستطاع ان يكون ف function بتاخد Arguments 3
* اكتر من 3 Arguments مفروض يبقا عندك مبررات قويه وبرضو مينفعش تعمل كدة يعنى من الاخر مينفعش تخلى ال function تاخد اكتر من 3 Arguments

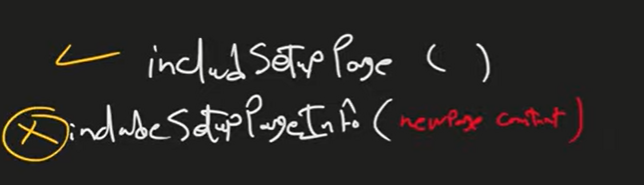
ودة عشان ال Arguments بتاخد معايه جهد زهنى عشان تفهمها .

ولو اخدنا مثال زى : لو خلينا مثلا ال function رقم 2-3 تاخد argument اضافى زى ال buffer ف كدة هيخلى الموضع اصعب للفهم يعنى وانت بتقره الfunction هتكون عاوز تعرف هى بتعمل ايه وكمان عاوز تعرف ايه لزمه الbuffer ف دة جهد اضافى يجب .



ودة مثال تانى : طبعا الاسهل ف الفهم الfunc الاوله ودة بسبب انها مش بتاخد argument

تانى func دى بتاخد argument من abstraction level مختلف عن اسم الfunction وهيجبرك تعرف تفاصيل اكتر مش وقتها دلوقتى .



ومن اهم الاسباب الى بتخلينا نبعد عن ال argument ان ال argument بتخلى موضوع ال testing صعب .يعنى عشان تعمل مجموعه من ال test case تتاكد بيها ان الfunction شغاله مظبوط المضوع هيكون صعب وكبير .

ف لو مفيش اى argument موضع الtesting هيكون سهل خاالص.

اما لو فيه argument واحد موضع الtesting هيكون مش صعب برضو .

اما لو فيه اتنين argument ف موضوع الtesting هيكون صعب شويه .

اكتر من اتنين argument الموضع هيبقا صعب فعلا .

وموضع ال output argument اصعب ف الفهم من ال input argument . يعنى احنا متعودين على فكرة ان الحاجة الى هتدخل ال function هتكون عن طريق ال argument والحاجة الى هتطلع من الfunc هتكون عن طريق ال return value ف دة المتعارف متعودناش ان ف حاجة تخرج من ال function عن طريق argument . عشان كدة ال output argument بتكون حاجة غريبه او مش متعودين عليها .

ف افضل حاجة ان يكون ف argument واحد بعد ان ميكونش فيه اى argument .

## 6.1- Common Monadic Forms :

Monadic يعنى اوحادى.

ف السكشن دة بتكلم عن الحلات المشهوه الى بيكون فيها ال function بتاخد one argument .ف حالتين عشان نبعت one argument للfunction .

1. انك تكون بتسئل سوال عن ال argument :

* ودة مثال boolean fileExists(“MyFile”); وهنا كنا بنسئل عن file هو موجود ولا لا ؟

1. انك تكون عاوز تحول type معين ل type تانى :

* وده مثال InputStream fileOpen(“MyFile”) وهنا عاوزين نحول ال string مثلا InputStream. ولاحظ هنا خلينا الحاجة الى هترجع من func هتكون return value .

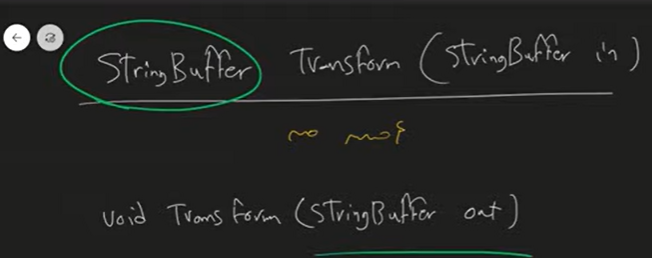
1. ف حاله تالته عشان اخلى ال function تكون بتاخد argument one وهى مش مشهورة بس موجوده الحاله دى هى انك هتعمل *event وفى الحاله دى بيكون ف* argument  *input ومفيش output argument . وف الحاله دى اخنا بنستخدام ال input argument عشان نعمل تغير لstate معنيه ف ال system .*

* *ودة مثال* void passwordAttemptFailedNtimes(int attempts) وعمك بوب بيقلك خلى بالك وانت بتتعامل مع النوع دى لازم تكتب اسم معبر وواضح ان ده event .

1. ولازم تتجنب ال function الى مش بتتبع ال format او ال pattern الى فات :

دة مثال مفروض تتجنب ان هو يحصل : includeSetupPageInto(StringBuffer pageText) وهنا هو استخدام ال input argument على انه كمان output . ودة غلط ان يحصل اى function بتعمل transform من type ل type تانى استخدام ال return value .

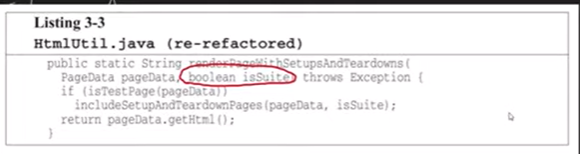
* ف بختصار الشكل الشكل الاول احسن من الشكل التانى ف الاسكرين الى جايه عشان بستخدام ال return value .



## 6.2-Flag Arguments :

حاجة مش حلوه تماما انك تبعت ل function متغير boolean عشان دة بيخلها معقدة وبكل وضوح كدة ال func دى هتعمل اكتر من حاجة واحنا قلنا ان من الافضل نخلى ال func تعمل حاجة واحدة بس .

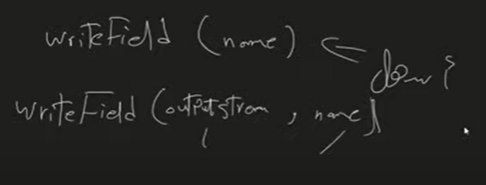
* ف انت لما تبعت ل function متغير bool اكيد كدة هتعمل حاجة معينه ف حاله ال false وحجاجه تانى ف حاله ال true .
* وعم بوب بيقلك ان هو بعت bool flag للfunction رقم 3-3 الى كان عملها عصب عنه عشان الfunction رقم 2-3 الى كان بيعملها refactoring عشان يطلعfunction رقم 3-3 كانت بتاخد boolean



* **والحل لما نلاقى function هتاخد flag خلها 2 function مش واحدة . اول func تكون بتعمل حاله ال false وتانى function تكون بتعمل حاله ال true .**

## 6.3 - Dyadic Functions:

Dyadic الكلمه دى معنها ثونائيه . وبيقلك ان ال function الى بتاخد 2 argument صعب شويه ف الفهم اصعب من الى بتاخد argument one .

يعنى لوعندى اتنين function زى الى ف الاسكرين هتلاقى الاوله اسهل ف الفهم . وانت ممكن تتجاه اول argument ق تانى func عشان تفهم ال function بتعمل ايه ودة حاجة مش كويسه لان غالبا ال bug بتكون موجود ف الجزء الى بتتجاهله.

واكيد سعات بنكون محتاجين نعمل اتنين argument ف موضع ان عمل function بتاخد 2 argument مش دايما بيكون حاجة غلط.

فى حلات زى مثلا ان اعرف point(x,y) ف منطقى ف الحاله دى ان يكون فيه 2 parameter . للfunction او ال class .

واكيد هتكون مندهش لو شفت الشكل دة point(0) مش منطقى ان يبقا فيه نقطه ليها احداثى واحد بس . وف الحاله دى بنقول

على two argument انهم *are ordered components of a single value عكس writeFiled(outputStream,name ) دى مفيش اى صله او ترابط بين ال argument بتوعها ومفيش* natural ordering يخلنى مطلغبطش واكتب الاول مكان التانى وانا بعمل call .

وحتى الدوال الواضحه الى بتاخد 2 argument زى assertEquals(expected, actual) الى بنستخدمها ف ال testing ممكن تسبب مشكله كام مرة وانت بتعمل call عكست ال argument . ف دة مثال تانى لعدم وجود ordering naturalبين ال argument .

* ف ال function الى بتاخد 2 argument مش وحشه بس لازم تعرف ان ليهم cost زى انك تتلغبط ف ترتبهم .

ولازم تستغل اى فرص ممكن تحول فيها من argument 2 الى one argument .

* وعم بوب بيدينا حلول ممكن نحول بيها ال function الى اسمها) *writeFiled(outputStream,name*

1. *ممكن اخلى writeFiled دى تكون داله جوة ال outputStream class بحيث اقدر اعمل الشكل دة* outputStream.  
   writeField(name)
2. *تانى حل ان اخلى ال outputStream دى member field جوة الكلاس الى فيه داله writeFiled*
3. تالت احل ان اعمل كلاس جديد اسمه FieldWriter يكون بياخد outputStream ف ال contractor بتاعه وجواه write method.

## 6.4- Triads:

السكشن دة هيتكلم عن شويه ملحوظات على الfunction الى بتاخد 3 argument . ف بسم الله كدة لازم تفكر كتير قبل متعمل function بتاخد 3 argument . وحتى لو لحظت عم بوب كان بيشد ف شعره من ال function الى بتاخد 2 argument مبالك بقا بالى بتاخد مبالك بقا بالى بتاخد argument 3

* ف حالتنا هنا هتلاقى المشاكل بتاعه ا ل ordering اكتر بكتير من السكشن الى فات وكمان مشاكل ال ignoring.

## 6.5- Argument Objects:

لو ف function محتاجه اكتر من اتنين او تلاته argument ف ممكن تحط بعض او كل ال argument دى فى كلاس مع بعض .

وعلى سبيل المثال لو عندى ال function دى Circle makeCircle(double x, double y, double radius);

ممكن نخليها بالشطل ده Circle makeCircle(Point center, double radius);

* ف موضوع ان اقلل عدد ال argument دى حاجة حلوه واحنا كل الى عملناه ف المثال ان خلينى x,y يكونوا ف كلاس مع بعض وده عشان الاتنين مرتبطين ببعض .

## 6.6- Argument Lists:

انا مفهمتهوش الصراحه ف هكتبه بعد مخلص الشبتر ان شاء الله

## 6.7- Verbs and Keywords:

اختيار اسم ال الfunction ممكن ياخد وقت طويل عشان تخليها معبرة عن وظيفه function و ترتيب ال argument بس الموضع يستحق .

ف حاله ال function الى بتاخد one argument لازم تخلى اسم ال function وال argument متناسقين ومعبيرن . واحنا عرفين ان اسم ال function بيكون verb او بيعبر عن action معين , والargument بيكون noun ف عم بوب بيقلك اختار verb/non معبرين .

* تعاله ناخد مثال لو هنعمل مثلا function كدة write(name) دى كويسه وكل حاجة وهنا ال name هو الحاجة الى هتتكتب . بس لو خلينها كدة writeField(name) دى معبرة اكتر عشان بتقلنا الاسم الى هيتكتب دى field . ف المثال ده كان بيوضح الوظيفه بتاعه الfunction
* احنا ممكن كمان نخلى اسم الfunction بيعبر عن ال ordering بتاع ال argument زى مثلا فى الداله بتاعه ال testing نخلى اسمها كدة assertExpectedEqualsActual(expected, actual)

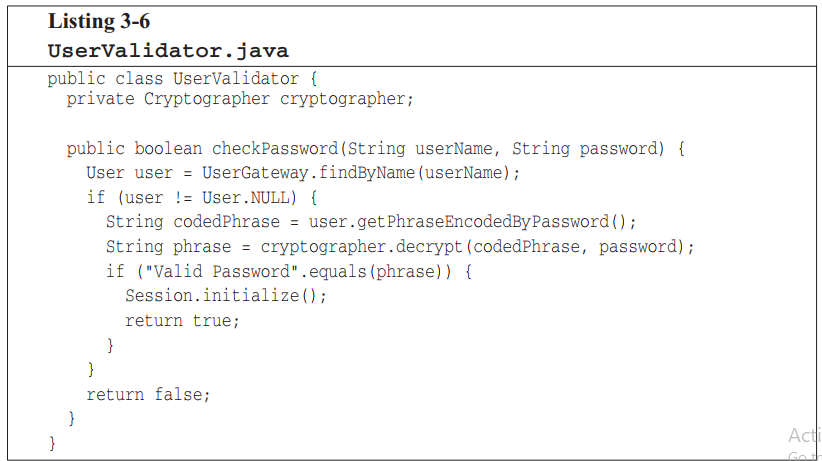
# 7- Have No Side Effects :

Side Effects :عم بوب بيقلك معنها lies اكاذيب يعنى ال function بتاعتك تكون بتوعدنا او بتظهر من خلال اسمها انها بعتمل حاجة واحدة بس ومن جوها بتون بتعمل حاجة تانى اضافيه مش متوقعه والحاجة زى زى انها تغير ف field ف الكلاس او تغير ف parameter هى بتاخدة كدة يعنى .

* ف كل الحلات الى قلتها دى غالبنا بتؤدى لمشاكل كبيرة وغالبا بتؤدى لحاجة

اسمها temporal couplings and order dependencies.

* Temporal couplings المصطلح ده معنها الاقتران الذمنى يعنى الكود بيبقا معتمد على الوقت .

تعاله ناخد مثال على Side Effects : الfunc دى بتعمل match a userName to a password بس ليها Side Effects ؟!

* الفانكشن دى من اسمها check pasw وهتلاقى جوها بتعمل Session.initialize(); يعنى اسمها مش بيدى اى اشاره ان بيغير ف ال Session الى موجودة **. وخطورة الموضع** ده ان ممكن تلاقى واحد بيعمل validate للpasw ومن غير مياخد باله كدة ال session الى موجودة هتكون اتمسحت عشان حصلها initialize .
* ف ال side effects : ده بيسبب ال temporal coupling يعنى ال function هنديها ف وقت معين بس او بمعنى تانى هقدر انادم ال function لما يكون امن ان اعمل initialize لل session .يعنى مش هسبب اى مشكله لو حصل initialize ودة طبعا مش متحاح ف كل الاوقات .
* لو انت مضتر ان يكون فيه او يحصل Temporal couplings ف خلى اسم ال function يكون واضح فيه ال Temporal couplings يعنى هنخلى اسم ال function الى فاتت كدة checkPasswordAndInitializeSession ودة طبعا بينتهك اول مبداء احنا قلناه ان ال function لازم تعمل حاجة واحدة بس دة غصب عننا والاحسن ننتهك اول مبداء على ان يبقا فيه Temporal couplings مخفى.

## 7.1- Output Arguments:

احنا متعودين نخلى ال argument انه يكون input form function . ولو انت كنت مبرمج ليك اكتر من سنه اكيد انت

خليتargument input يكون output ف نفس الوقت. يعنى مثلا عملت كدة

public void appendFooter(StringBuffer report)

appendFooter(s);

* اى حد هيقرة ال function دى بعدك هيفهم ايه هل انت بضيف حاجة عل ال s ولا بضيف s على حاجة
* وهل ال s دى input ولا ف نفس الوقت output

ف المضوع هياخد منك وقت لو كنت بتخلى ال input ف بعض الاخيان output . ف بختصار تجنب انك تخل ال input argument يكون output كمان . واى حاجة هتخليك تروح تشوف ال declaration بتاع ال function دى يعتبر مجهود عليك ومفروض تتجنبه .

* ولو كان لازم تخلى func تغير ف ال state بتاعه argument خليها فى ال state بتاع ال object الى هى فيه : كدى يعنى

report.appendFooter(); ف كدة ال object بتاعك هو report ولما عملت call لل appendFooter غيرت ف state موجودة جوة ال object .

# 8- Command Query Separation :

Command دى معناها امر يعنى مثلا هعملset لقيمه متغير هحزف حاجة يعنى بعمل action من الاخر .

Query: دى معناها تساؤل يعنى بسئل مثلا عن موجود attribute معين ولا لا يعنى مجرد سوال اجابته ب true او false .

* ف احنا اتفقنا ان ال function هتعمل حاجة واحدة بس لما هتجاوب على سوال او هتعمل حاجة معنيه .

ف تعاله تاخد مثال عشان نفهم السكشن دة :

Public boolean set(String attribute, String value);

الفنكشن دى تحط قيمه ف attribute ميعن لو موجود وبترجع true ولو ال attribute مش موجود هترجع false .

المشكله بقا لو عملنا حطينا ال call بتاع ال function ف condition :

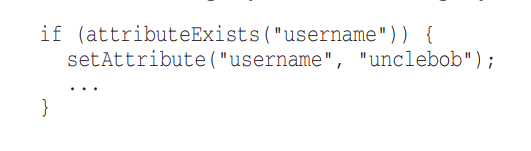
if(set("username", "unclebob"))

لو انت reader للstatement دى فاتت دى هتفهم ايه ؟

* هل هو بيسئل اذا كان ال username هو unclebob ؟
* هل هو بيسئل اذا username بقى unclebob ؟
* يعنى مش واضح انا بسئل عن ال username ؟ ولا يعمل action على ال username زى انى اغير قيمته ؟
* يعنى الى عمل الكود كان غرضه من set انها تعمل action بس لما حطينا ال set جوة if مبقيش الغرض بتاع ال function واضح للى هيقرها والغرض لاى حد هيشوف ال function انها بتشوف قيمه username مش بتغير فيها .

**ف الحل**: ممكن نغير اسم ال function ونخليها setAndCheckIfExists بس الحل دة هتحسه محلش الموضع ف حاله ال if condition ولسه الدنيا ملعبكه .

* **الحل الاصح الى مفروض نعمله ان نفصل ال quarry عن ال command :**
* يعنى ف الاول نعمل ال query الى هتشوف هو ال attribute موجود ولا لا ودى هتكون condition ال if .
* وبعد كدة نعمل ال command الى هيغير القيمه بتاعه ال attribute ودة هيكون ال body بتاع ال if .



* ف كدة بختصار فصلنا ال query او التساؤل عن ال command او عمله ال set .ودة عنوان السكشن .

# 9-Prefer Exceptions to Returning Error Codes:

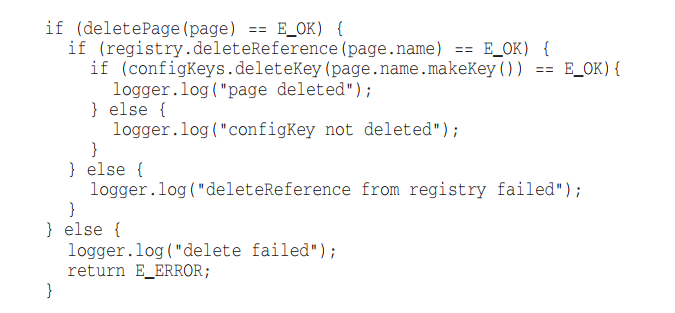
الافضل انك تستخدام ال Exceptions على ان ترجع Error Codes وتعاله نشوف ازاى :

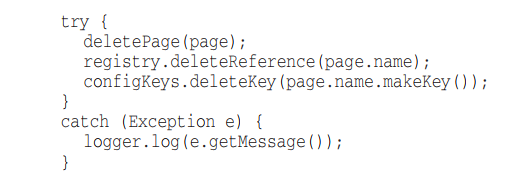
* انك ترجع Error Codes من function كدة انت بتنتهك مبداء ال Command Query Separation تعاله نشوف ازاى :

if (deletePage(page) == E\_OK)

دة مفهوش المشكله بتاعه ال set الى شفنها من شويه يعنى مفهوش مشكله verb/objective بس فيه مشكله انك هتعمل nested structure . ولما ترجع error code انت كدة بتتسبب او بتخلى ال caller يتعمال مع ال error حالا .

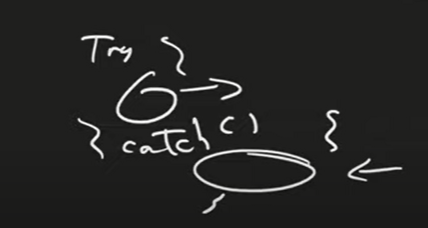
ودة شكل ال nested structure : وطبعا الشكل دة غلس خاالص ومتعب ف الفهم

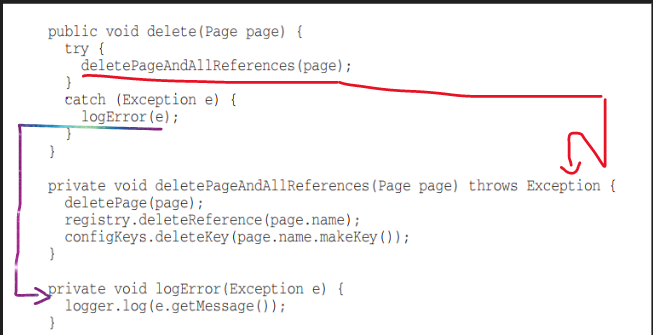
* وعشان هو بيرجع error code ف هو عامل else لكل if

اما بقا لو انت كنت استخدامت exception : ف انت مش بس لغيت ال nested structure انت كمان عملت فصل لerror code عن ال happy path ودة هيكون الشكل بتاع الكود .

## 9.1- Extract Try/Catch Blocks :

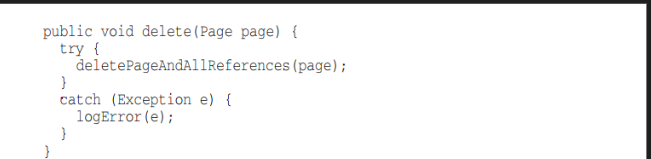
طبعا عم بوب مش هيسيب ال try/catch statement :

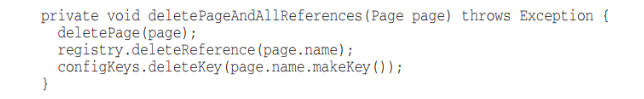
try/catch دى ف حد ذاتها وحشه وبطلغبط الكود ف الافضل انى اخلى ال body بتاع ال try هو function وال body بتاع ال catch كمان function بالشكل دة .

ف الfunction بتاعه ال delete الشكل النهاشى ليها هيكون كده : وكل الى عمله ان حولت body بتاع الtry هو وبتاع ال catch ل function .

الفايدة من الاسكرين الى فاتت :

1. ان خليت الfunction دى تكون مسئوله عن ال error processing وطبعا كدة هتكون سهله ف الفهم خااااالص وكمان اقدر اتجاهل الfunction دى لو هى مش هتمنى وانا عاوز خطوات الحذف صفحه بس .



1. ان كمان خليت مراحل حذف الصفحه لوحدها وكدة اقدر اتجاهل عمليه ال error log لو هى مش هتهمنى .
2. ودى عمليه ال error log

## 9.2- Error Handling Is One Thing :

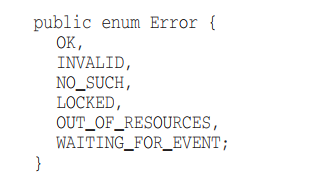
ال function مفروض انها بتعمل حاجة واحدة وال Error Handling دة المفورض انه حاجة واحدة . ف لو عندى function بتعمل handling error ف المفروض متعملش حاجة تانى .معنى كدة ان لو فtry/catch جوة function ف مفروض ال try دى تكون اول حاجة ف ال function وال catch تكون اخر جاجة ف ال function .

## 9.3- The Error.java Dependency Magnet :

Magnet : magnet دة معنا معناطيس .

Dependency : ف حالتنا هنا file معتمد على file تانى .

Error.java : دة file

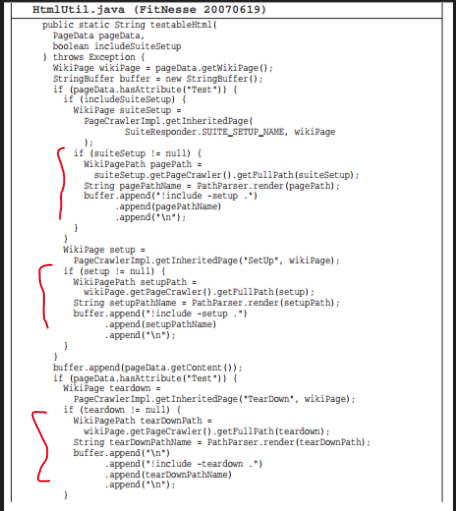
لو انت كنت بتستخدام ال error code او بترجع error code ف دة معناه انك عامل enum او كلاس معرف فيه ال error .

ف عم بوب بيقلك ان ال كلاس الى زى كدة دى بتكون *dependency magnet يعنى كلاس تانى لازم تعملها import وتستخدمها .*

*ولما ال* Error enum يتغير كل ال class الى بتسخدامه هتحتاج تعمل recompiled , redeployed ف عشان كدة الdeveloper مش هيضيفه اى error code على ال enum وهسيستخدامه اى error موجود وخلاص ودة طبعا عك يصحبى .

* ف بدال كل الدوشه دى استخدام ال exception بدال ال error code . وكدة كدة كل ال new exceptions هتكون مشتقه من ال exceptions ف عشان كدة مش هتحتاج تعلم recompilation or redeployment
* ف استخدام ال exceptions يحب عشان كمان ال enum الى هيكون فيه error دة بيتنهك مبداء ال open/closed principle .

# 10- Don’t Repeat Yourself :

عم بوب بيقلك ان صورة 3-1 دى كبيرة خااالص عشان فيها كمان algorithm بيتكرر حوالى 4 مرات

ف التكرارات دى ممكن مكنتش اخت بالك منها لو مقلتلكش عليها .

والتكرار هو اساس كل الشرور ف ال sw وف حجات كتير وجدت عشان تمنع

وجود تكرار ف ال code زى ال

oop , Structured programming, Aspect Oriented Programming,

Component Oriented Programming,

# 11- Structured Programming:

عم بوب بيقول ان فيه بعض المبرمجين بيتبعه قواعد عم Dijkstra والى بتقول : ان اى function او اى block جوة function لازم يبقا فيه

One entry و one exit يعنى كل function مفروض يبقا فيها return وحدة بس ويبقاش فيه اى break او continue او goto ومعنى goto دى ان يبقا فيه function call تخلينى اروح لمكان تانى .

* لو انت بتخلى الfunction بتاعتك صغيرة قاعدة عم Dijkstra مش هتبقا مواثرة معاك بل فى بعض الاحيان ممكن تخلى ال function اوضح .

# 12- How Do You Write Functions Like This?

عم بوب بيقلك ان كتابه ال sw زى كتابه اى حاجة ف الدنيا زى انك بتكتب مقاله :

* اول حاجة بتحط افكارك
* بعد كدة بتبداء تصيغ الافكار دى ف الاول النسخه الاوله اكيد هتكون معقده او مش مرتبه
* وعم بوب بيقلك هو لما بيكتب function بتكون فالاول معقدة وملغبطه ف الاول
* وبعد كدة بيفضل يظبط الكلام الى كتبه وفى الاخر بيكون عندة ال function بتتبع كل القواعد الى حطها .
* وبيقلك هو مش متخيل ان ف حد ممكن يتكت الكود من الاول صح ومتبع القواعد ف عادى عك وبعد كدة صلح الدنيا.

ومتفهمنيش غلط صلح الدنيا قبل متسلم المشورع .