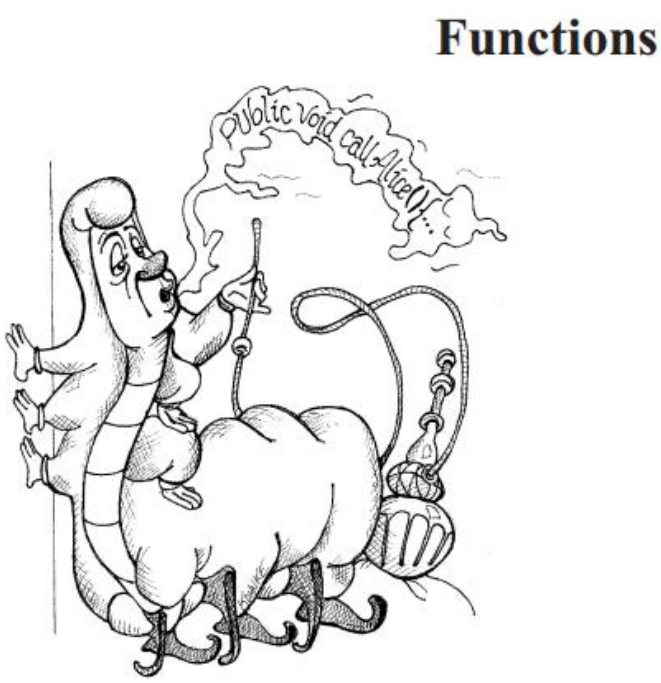
9/1/2022

Clean code

Ch3 Functions



Abdurrahman Gamal Ahmed Gaber
NO CAMPANY

Table of Contents

الشبتر دة كان مليان معلومات الصراحه . وهو بيتكلم عن ال function وال function دى مهمه جدااا عشان بتعتبر من بدهيات التنظيم لاى برنامج عمنا بوب من البدايه كدة عشان يفهمنا المشكله عطانى اتنين فانكشن عشان نقارن بينهم :دى اول فانكشن

Listing 3-1
HtmlUtil.java (FitNesse 20070619)

```
public static String testableHtml(
  PageData pageData,
  boolean includeSuiteSetup
) throws Exception {
  WikiPage wikiPage = pageData.getWikiPage();
  StringBuffer buffer = new StringBuffer();
  if (pageData.hasAttribute("Test")) {
    if (includeSuiteSetup) {
      WikiPage suiteSetup =
        PageCrawlerImpl.getInheritedPage(
                SuiteResponder.SUITE_SETUP_NAME, wikiPage
        );
      if (suiteSetup != null) {
        WikiPagePath pagePath =
          suiteSetup.getPageCrawler().getFullPath(suiteSetup);
        String pagePathName = PathParser.render(pagePath);
        buffer.append("!include -setup .")
              .append(pagePathName)
              .append("\n");
      }
    }
    WikiPage setup =
      PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
    if (setup != null) {
      WikiPagePath setupPath =
        wikiPage.getPageCrawler().getFullPath(setup);
      String setupPathName = PathParser.render(setupPath);
      buffer.append("!include -setup .")
            .append(setupPathName)
            .append("\n");
    }
  }
  buffer.append(pageData.getContent());
  if (pageData.hasAttribute("Test")) {
    WikiPage teardown =
      PageCrawlerImpl.getInheritedPage("TearDown", wikiPage);
    if (teardown != null) {
      WikiPagePath tearDownPath =
        wikiPage.getPageCrawler().getFullPath(teardown);
      String tearDownPathName = PathParser.render(tearDownPath);
      buffer.append("\n")
            .append("!include -teardown .")
            .append(tearDownPathName)
            .append("\n");
    }
```

Listing 3-1 (continued)
HtmlUtil.java (FitNesse 20070619)

```
    if (includeSuiteSetup) {
      WikiPage suiteTeardown =
        PageCrawlerImpl.getInheritedPage(
                SuiteResponder.SUITE_TEARDOWN_NAME,
                wikiPage
        );
      if (suiteTeardown != null) {
        WikiPagePath pagePath =
          suiteTeardown.getPageCrawler().getFullPath (suiteTeardown);
        String pagePathName = PathParser.render(pagePath);
        buffer.append("!include -teardown .")
              .append(pagePathName)
              .append("\n");
      }
    }
  }
  pageData.setContent(buffer.toString());
  return pageData.getHtml();
}
```

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

Listing 3-2
HtmlUtil.java (refactored)

```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite
) throws Exception {
  boolean isTestPage = pageData.hasAttribute("Test");
  if (isTestPage) {
    WikiPage testPage = pageData.getWikiPage();
    StringBuffer newPageContent = new StringBuffer();
    includeSetupPages(testPage, newPageContent, isSuite);
    newPageContent.append(pageData.getContent());
    includeTeardownPages(testPage, newPageContent, isSuite);
    pageData.setContent(newPageContent.toString());
  }

  return pageData.getHtml();
}
```

- 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 1-3 دكانت 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 functionاصغر 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 الfunction 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 الfunction مفروض يبقا قد ايه؟ بيقلك شايف ال function رقم 3-2 ال function بتاعتك لازم تبقا اصغر منها ودة شكل ال function الى عمنا بوب عاوزك توصله.

- وملحوظه 3-2 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 3-1.

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

Listing 3-3
HtmlUtil.java (re-refactored)

```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite) throws Exception {
  if (isTestPage(pageData))
    includeSetupAndTeardownPages(pageData, isSuite);
  return pageData.getHtml();
}
```

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 .

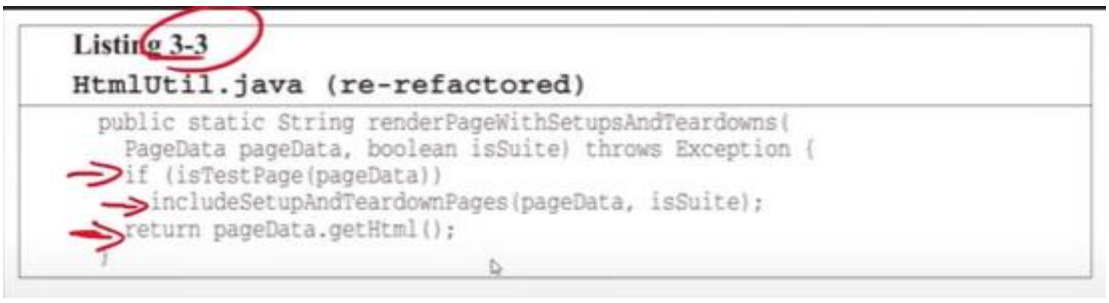
2-Do One Thing:

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

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

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

طيب تفتكر ان ال function 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 ؟ ال function 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


Listing 3-3
HtmlUtil.java (re-refactored)

```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite) throws Exception {
  if (isTestPage(pageData))
    includeSetupAndTeardownPages(pageData, isSuite);
  return pageData.getHtml();
}
```

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

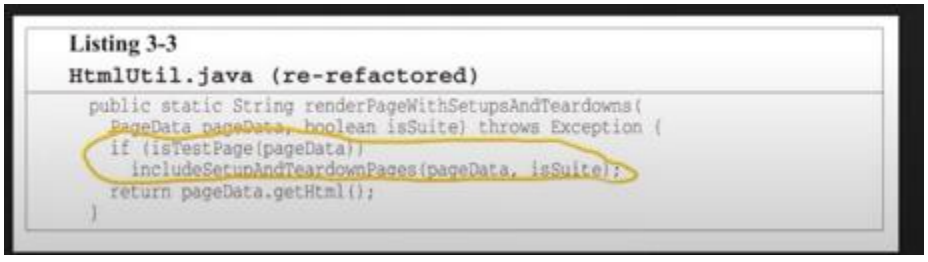
ف انا وصفت ال function كلها على اكنها one paragraph . ف كدة ال function دى فيها level واحد من ال abstraction وكده هى بتعمل حاجة واحدة بس . والموضع مهم ومن اكتر لمواضيع الى عجبتنى .

وفى العموما السبب الى بيخلينا نتكتب function اصلا هو اننا بنحلل او بنصغر concept كبير لمجموعه خطوات ف level ال abstraction 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 functionلوحده هقلك الى بتعمله ملهوش لازم ولا هيفرق اى حاجة ف ال readability .


Listing 3-3
HtmlUtil.java (re-refactored)

```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite) throws Exception {
  if (isTestPage(pageData))
    includeSetupAndTeardownPages(pageData, isSuite);
  return pageData.getHtml();
}
```

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 ال functionدى مفروض يحصلها refactoring وتبقا بتعمل حاجة واحدة بس .

```
public static int[] generatePrimes(int maxValue)
{
  if (maxValue >= 2) // the only valid case
  {
    // declarations
    int s = maxValue + 1; // size of array
    boolean[] f = new boolean[s];
    int i;
```

Listing 4-7 (continued)
GeneratePrimes.java

```
    // initialize array to true.
    for (i = 0; i < s; i++)
      f[i] = true;

    // get rid of known non-primes
    f[0] = f[1] = false;

    // sieve
    int j;
    for (i = 2; i < Math.sqrt(s) + 1; i++)
    {
      if (f[i]) // if i is uncrossed, cross its multiples.
      {
        for (j = 2 * i; j < s; j += i)
          f[j] = false; // multiple is not prime
      }
    }

    // how many primes are there?
    int count = 0;
    for (i = 0; i < s; i++)
    {
      if (f[i])
        count++; // bump count.
    }

    int[] primes = new int[count];

    // move the primes into the result
    for (i = 0, j = 0; i < s; i++)
    {
      if (f[i])            // if prime
        primes[j++] = i;
    }

    return primes;  // return the primes
  }
  else // maxValue < 2
```

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 .

Listing 3-1 (continued)
HtmlUtil.java (FitNesse 20070619)

```
    if (includeSuiteSetup) {
      WikiPage suiteTeardown =
        PageCrawlerImpl.getInheritedPage(
                SuiteResponder.SUITE_TEARDOWN_NAME,
                wikiPage
        );
      if (suiteTeardown != null) {
        WikiPagePath pagePath =
          suiteTeardown.getPageCrawler().getFullPath (suiteTeardown);
        String pagePathName = PathParser.render(pagePath);
        buffer.append("!include -teardown .")
              .append(pagePathName)
              .append("\n");
      }
    }
  }
  pageData.setContent(buffer.toString());
  return pageData.getHtml();
}
```

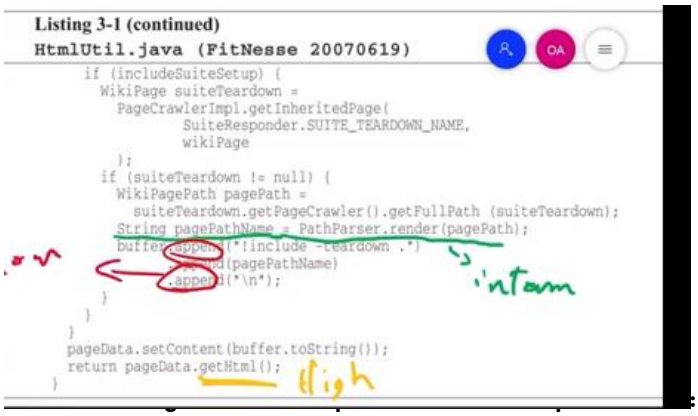
- والجزء دة الى معلم بالخضر دة يعتبر intermediate level of abstraction .

Listing 3-1 (continued)
HtmlUtil.java (FitNesse 20070619)

```
if (includeSuiteSetup) {
  WikiPage suiteTeardown =
    PageCrawlerImpl.getInheritedPage(
            SuiteResponder.SUITE_TEARDOWN_NAME,
            wikiPage
    );
  if (suiteTeardown != null) {
    WikiPagePath pagePath =
      suiteTeardown.getPageCrawler().getFullPath (suiteTeardown);
    String pagePathName = PathParser.render(pagePath);
    buffer.append("!include -teardown .")
          .append(pagePathName)
          .append("\n");
  }
}
pageData.setContent(buffer.toString());
return pageData.getHtml();
```

- 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.

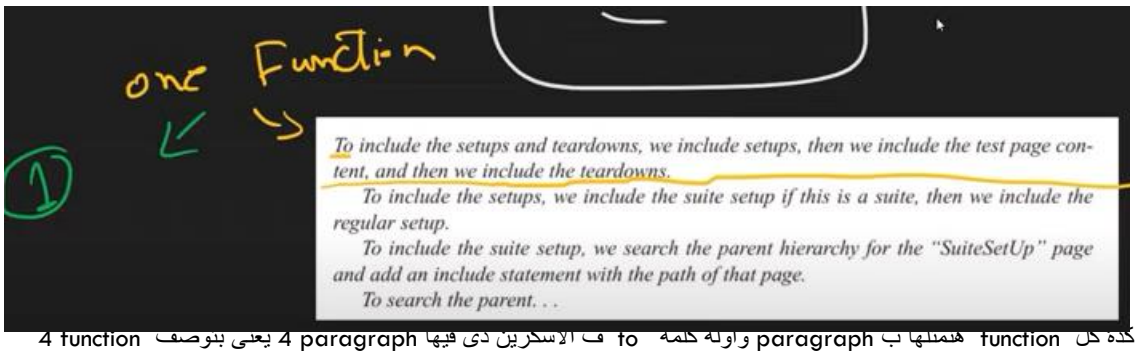
ف عمنا بوب بيقلك لو لقيت حاجة زى كدة ف دة عك ومتعملهوش .



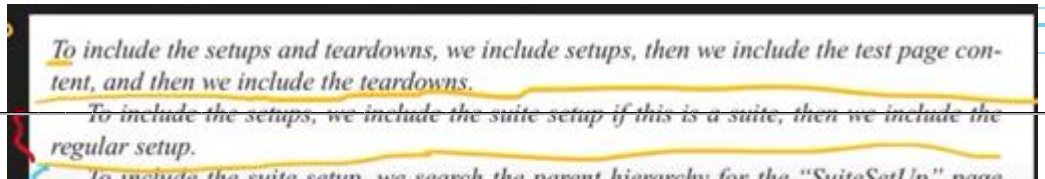
ف السكشن دة عم بوب بيقلنا احنا عوزين الكود يبقا كانه روايه يتقراء من فوق لتحت يعنى اكنك بتقره جرنان او روايه .
وكل function تكون ال function الى بعدها اقل ف level abstraction . يعنى كل مانزل لتحت ف الكود انقص level abstraction بمقدار واحد بس.

ف الطريقه دى اسمها Stepdown Rule : وبختصار احنا عوزين نحول الكود ل paragraphs بدايه كل paragraph كلمه to وشرحت موضع ال to من شويه .

- وعم بوب بيديك مثال على تحويل الكود ل paragraphs 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 واولة كلمة to ف الاسكرين دى فيها 4 paragraph يعنى بنوصف 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 ال function وحتى مش متعقد منها .
- ف اكيد الوضوع واضح او ازاى كويس وبالاخص لو طبقت قاعه ال to .

```
import fitnesse.responders.run.SuiteResponder;
import fitnesse.wiki.*;

public class SetupTeardownIncluder {
  private PageData pageData;
  private boolean isSuite;
  private WikiPage testPage;
  private StringBuffer newPageContent;
  private PageCrawler pageCrawler;


  public static String render(PageData pageData) throws Exception {
    return render(pageData, false);
  }

  public static String render(PageData pageData, boolean isSuite)
    throws Exception {
    return new SetupTeardownIncluder(pageData).render(isSuite);
  }

  private SetupTeardownIncluder(PageData pageData) {
    this.pageData = pageData;
    testPage = pageData.getWikiPage();
    pageCrawler = testPage.getPageCrawler();
    newPageContent = new StringBuffer();
  }

  private String render(boolean isSuite) throws Exception {
    this.isSuite = isSuite;
    if (isTestPage())
      includeSetupAndTeardownPages();
    return pageData.getHtml();
  }

  private boolean isTestPage() throws Exception {
    return pageData.hasAttribute("Test");
  }

  private void includeSetupAndTeardownPages() throws Exception {
    includeSetupPages();
    includePageContent();
    includeTeardownPages();
    updatePageContent();
  }
```

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 :

Listing 3-4
Payroll.java

```
public Money calculatePay(Employee e)
throws InvalidEmployeeType {
    switch (e.type) {
      case COMMISSIONED:
        return calculateCommissionedPay(e);
      case HOURLY:
        return calculateHourlyPay(e);
      case SALARIED:
        return calculateSalariedPay(e);
      default:
        throw new InvalidEmployeeType(e.type);
    }
  }
```

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 ال bussines 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 جداااا جداااااااااااااااااا "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 ال system 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 .

Listing 3-5
Employee and Factory

```
public abstract class Employee {
  public abstract boolean isPayday();
  public abstract Money calculatePay();
  public abstract void deliverPay(Money pay);
}
-----------------
public interface EmployeeFactory {
  public Employee makeEmployee(EmployeeRecord r) throws InvalidEmployeeType;
}
-----------------
public class EmployeeFactoryImpl implements EmployeeFactory {
  public Employee makeEmployee(EmployeeRecord r) throws InvalidEmployeeType {
    switch (r.type) {
      case COMMISSIONED:
        return new CommissionedEmployee(r) ;
      case HOURLY:
        return new HourlyEmployee(r);
      case SALARIED:
        return new SalariedEmploye(r);
      default:
        throw new InvalidEmployeeType(r.type);
    }
  }
}
```

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 لل function 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 3 Arguments
- 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 ال function هتكون عاوز تعرف هى بتعمل ايه وكمان عاوز تعرف ايه لزمه الbuffer ف دة جهد اضافى يجب .

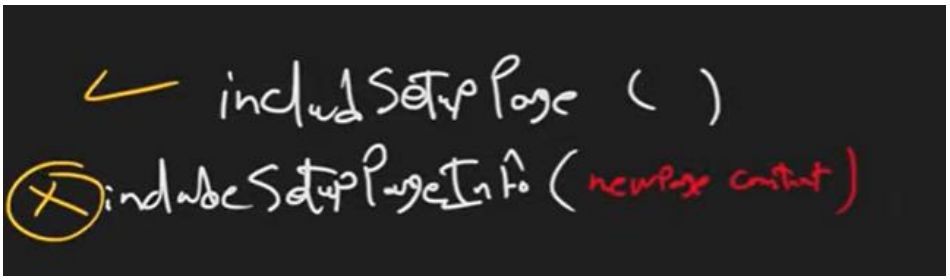
Listing 3-2
HtmlUtil.java (refactored)

```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite
) throws Exception {
  boolean isTestPage = pageData.hasAttribute("Test");
  if (isTestPage) {
    WikiPage testPage = pageData.getWikiPage();
    StringBuffer newPageContent = new StringBuffer();
    includeSetupPages(testPage, newPageContent, isSuite);
    newPageContent.append(pageData.getContent());
    includeTeardownPages(testPage, newPageContent, isSuite);
    pageData.setContent(newPageContent.toString());
  }

  return pageData.getHtml();
}
```

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 ال function 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 ال function شغاله مظبوط المضوع هيكون صعب وكبير .

ف لو مفيش اى argument موضع ال testing هيكون سهل خالص.

اما لو فيه argument واحد موضع ال testing هيكون مش صعب برضو .

اما لو فيه اتنين argument ف موضوع ال testing 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 :

Monadicيعنى اوحادى.

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

1- انك تكون بتسئل سوال عن ال argument :
   - ودة مثال `boolean fileExists("MyFile");` وهنا كنا بنسئل عن `file` هو موجود ولا لا ؟

2- 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` .

3- ف حاله تالته عشان اخلى ال function تكون بتاخد one argument وهى مش مشهورة بس موجوده الحاله دى هى انك هتعمل *event وفى الحاله دى بيكون ف* argument *input ومفيش output argument . وف الحاله دى احنا بنستخدام ال input argument عشان نعمل تغير ل state 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` .

4- 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 لل function رقم 3-3 الى كان عملها عصب عنه عشان ال function رقم 3-2 الى كان بيعملها refactoring عشان يطلع function رقم 3-3 كانت بتاخد boolean

Listing 3-3
HtmlUtil.java (re-refactored)

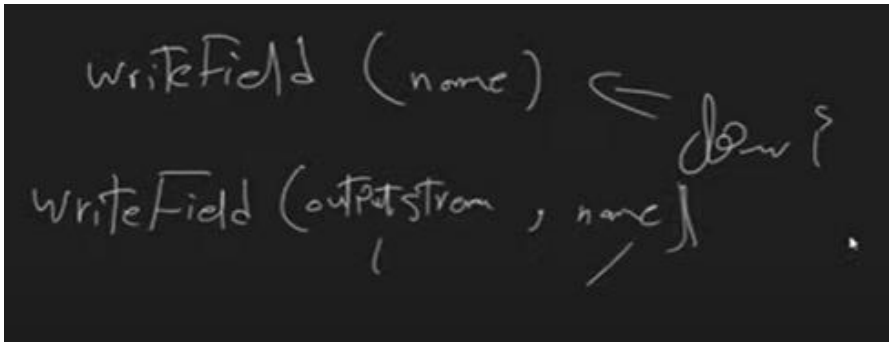
```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite) throws Exception {
  if (isTestPage(pageData))
    includeSetupAndTeardownPages(pageData, isSuite);
  return pageData.getHtml();
}
```

- **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 one argument .

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. لل function 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 argumentانهم *writeFiled(outputStream,name) عكس are ordered components of a single value 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 natural orderingبين ال `argument` .

- ف ال `function` الى بتاخد `argument` 2 مش وحشه بس لازم تعرف ان ليهم `cost` زى انك تتلغبط ف ترتيبهم . ولازم تستغل اى فرص ممكن تحول فيها من 2 `argument` 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 ال function 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 مبالك بقا بالى بتاخد 3 argument بالى بتاخد

- 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 ال function 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 , وال argument 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 ال `function`

- احنا ممكن كمان نخلى اسم ال function 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 : ال func دى بتعمل match a `userName` to a `password` بس ليها Side Effects ؟!

Listing 3-6
UserValidator.java

```
public class UserValidator {
  private Cryptographer cryptographer;

  public boolean checkPassword(String userName, String password) {
    User user = UserGateway.findByName(userName);
    if (user != User.NULL) {
      String codedPhrase = user.getPhraseEncodedByPassword();
      String phrase = cryptographer.decrypt(codedPhrase, password);
      if ("Valid Password".equals(phrase)) {
        Session.initialize();
        return true;
      }
    }
    return false;
  }
}
```

- 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 متاح 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 خليت argument 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 هعمل set 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 لل statement 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 .

```
if (attributeExists("username")) {
  setAttribute("username", "unclebob");
  ...
}
```

- 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 الExceptions 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

```
if (deletePage(page) == E_OK) {
  if (registry.deleteReference(page.name) == E_OK) {
    if (configKeys.deleteKey(page.name.makeKey()) == E_OK){
      logger.log("page deleted");
    } else {
      logger.log("configKey not deleted");
    }
  } else {
    logger.log("deleteReference from registry failed");
  }
} else {
  logger.log("delete failed");
  return E_ERROR;
}
```

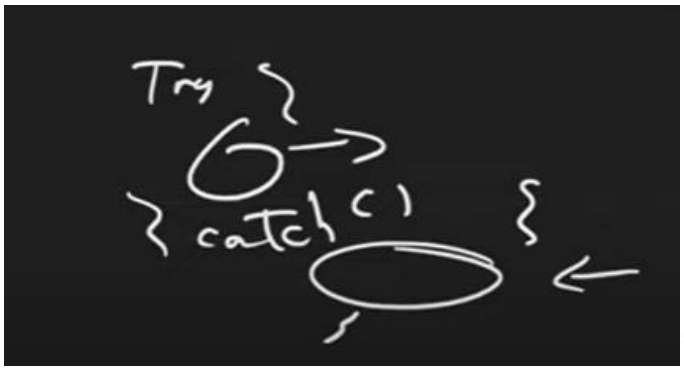
اما بقا لو انت كنت استخدامت exception : ف انت مش بس لغيت ال nested structure انت كمان عملت فصل ل error code عن ال happy path ودة هيكون الشكل بتاع الكود .

```
try {
   deletePage(page);
   registry.deleteReference(page.name);
   configKeys.deleteKey(page.name.makeKey());
}
catch (Exception e) {
   logger.log(e.getMessage());
}
```

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 ال function بتاعه ال delete الشكل النهاشى ليها هيكون كده : وكل .

```
public void delete(Page page) {
  try {
    deletePageAndAllReferences(page);
  }
  catch (Exception e) {
    logError(e);
  }
}

private void deletePageAndAllReferences(Page page) throws Exception {
  deletePage(page);
  registry.deleteReference(page.name);
```

الفايدة من الاسكرين الى فاتت :

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

```
public void delete(Page page) {
  try {
    deletePageAndAllReferences(page);
  }
  catch (Exception e) {
    logError(e);
  }
}
```

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

```
private void deletePageAndAllReferences(Page page) throws Exception {
  deletePage(page);
  registry.deleteReference(page.name);
  configKeys.deleteKey(page.name.makeKey());
}
```

3- ودى عمليه ال error log

```
private void logError(Exception e) {
  logger.log(e.getMessage());
}
```

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 ف try/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 .

```
public enum Error {
   OK,
   INVALID,
   NO_SUCH,
   LOCKED,
   OUT_OF_RESOURCES,
   WAITING_FOR_EVENT;
}
```

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 بتستخدامه هتحتاج تعمل recompiled , redeployed ف عشان كدة ال developer 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 1-3 دى كبيرة خاالص عشان فيها كمان algorithm بيتكرر حوالى 4 مرات

```
HtmlUtil.java (FitNesse 20070619)
  public static String testableHtml(
    PageData pageData,
    boolean includeSuiteSetup
  ) throws Exception {
    WikiPage wikiPage = pageData.getWikiPage();
    StringBuffer buffer = new StringBuffer();
    if (pageData.hasAttribute("Test")) {
      if (includeSuiteSetup) {
        WikiPage suiteSetup =
          PageCrawlerImpl.getInheritedPage(
                SuiteResponder.SUITE_SETUP_NAME, wikiPage
          );
        if (suiteSetup != null) {
          WikiPagePath pagePath =
            suiteSetup.getPageCrawler().getFullPath(suiteSetup);
          String pagePathName = PathParser.render(pagePath);
          buffer.append("!include -setup .")
                .append(pagePathName)
                .append("\n");
        }
      }
      WikiPage setup =
        PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
      if (setup != null) {
        WikiPagePath setupPath =
          wikiPage.getPageCrawler().getFullPath(setup);
        String setupPathName = PathParser.render(setupPath);
        buffer.append("!include -setup .")
              .append(setupPathName)
              .append("\n");
      }
    }
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test")) {
      WikiPage teardown =
        PageCrawlerImpl.getInheritedPage("TearDown", wikiPage);
      if (teardown != null) {
        WikiPagePath tearDownPath =
          wikiPage.getPageCrawler().getFullPath(teardown);
        String tearDownPathName = PathParser.render(tearDownPath);
        buffer.append("\n")
              .append("!include -teardown .")
              .append(tearDownPathName)
              .append("\n");
      }
```

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 وبيقاش فيه اى break او continue او goto ومعنى goto دى ان يبقا فيه function call تخليني 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 .

13- Conclusion:

اكتر سكشن بحبه الى فيه الذتونه :

كل system بيتم بناءه ب domain-specific language .ف ال function الى فى ال system بتكون هى الافعال الى ف اللغه دى والclass بتكون اسماء

- فن البرمجه كان وماذل هو فن ال design for language .

عم بوب بيقلك ان المبرمجين التقال بيفكرة ف ال system اكنه قصه هتتحكى مش برامج هتتكتب وبيستخدامه لغه البرمجه المناسبه ليهم عشان يبنه لغه معبرة اكتر ف وصف القصه بتاعتهم .

- الشبتر دة كان عن كيفيه كتابه ال function ف لو اتبعت القواعد الموجودة ال function بتاعتك هتكون صغيرة وليها اسم معبر وبتعمل حاجة واحد بس .
- وعم بوب بيقلك اوعى تنسه ان هدفك الاساسى انك تحكى القصه بتاعه ال system وان ال function الى بتكتلها محتاحه تبقا متلائمه مع بعض بشكل كويس ف لغه واضحه وبيسطه عشان تساعدك على بناء القصه او ال system .

لو كان الحاجة دى مفيدة ليك لو ممكن تقره الفاتحه لستى الله يرحمها وجميع موته المسلمين .