1/14/2023

Abdurahman Gamal Ahmed

no campany

Clean code

**Ch7 Error Handling**

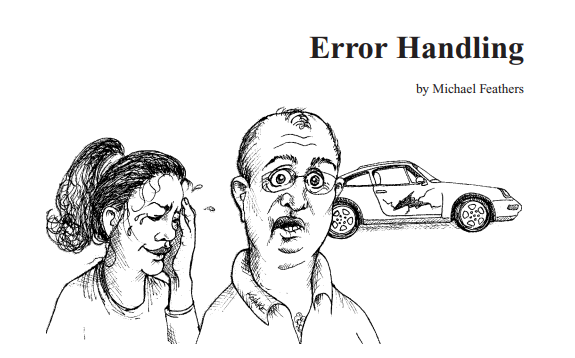


Table of Contents

[1-intro: 2](#_Toc124631613)

[2- Use Exceptions Rather Than Return Codes : 2](#_Toc124631614)

[2- Write Your Try-Catch-Finally Statement First: 3](#_Toc124631615)

[3- Use Unchecked Exceptions: 4](#_Toc124631616)

[4- Provide Context with Exceptions: 4](#_Toc124631617)

[5- Define Exception Classes in Terms of a Caller’s Needs: 4](#_Toc124631618)

[6- : 6](#_Toc124631619)

# 1-intro:

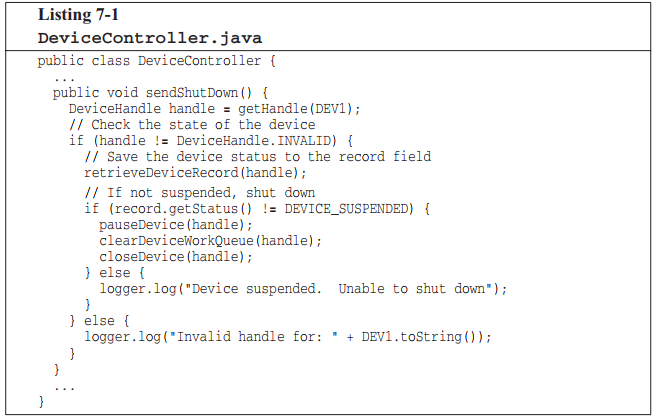
عم بوب بيبداء الشبتر وبيقلك يمكن حاجه غريبه ان يكون في شابتر كامل عن الError handling ف كتاب بتكلم عن الclean code .

ب الError handling هنا حاجه بتعملها مع الكود وممكن يكون ال input غلط والبرنامج يضرب ف دورنا كمبرمجين ان نصلح الدنيا ونتاكد ان الكود بيقوم بدوره تمم .

* ف الشابتر ده هنتكلم عن طرق واساليب نكتب بيها error handling بشكل clean وقوى .

# 2- Use Exceptions Rather Than Return Codes :

الاحسن نستخدام ال Exceptions بدال منرجع error code . زمان كان فيه لوغات كتير مفهاش Exceptions والطرق الى كانت مستخدامه عشان تهندل ال error كانت محدوده لما بتحظ error flag اوبترجع error code زى الاسكرين دى:

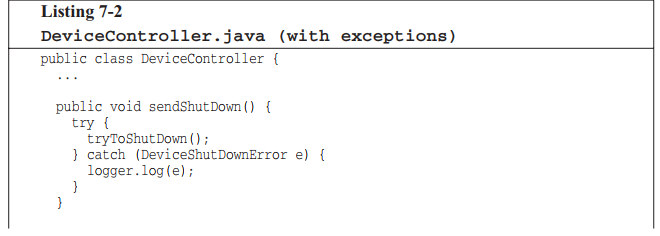


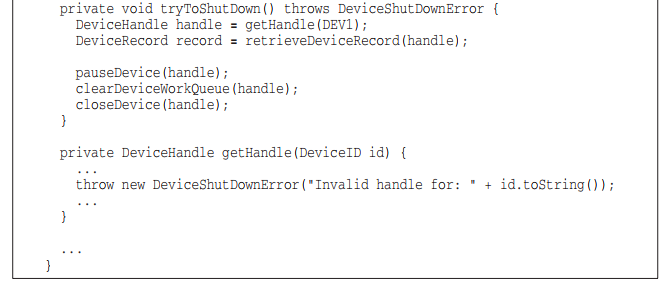
* الاسكرين دى بتوضح الطرق الى كانت بستخدامه زمان لعمل هاندل للerror : المشكله ف الطريقه دى انها بتلخبط الى هيستخدام الطرق دى .

والاحسن ان نعمل throw exception زى الصوره الى جايه :والطريقه الى جايه احسن عشان خلصتنا من مشكلتين ف الكود الى فات

1. ان فصلت ال logic بتاعنا عن ال error handling ف كدة نقدر نشوف كل واحد فيهم على حدا ونفهمه لواحده بطريقه منفصله.
2. انها خلت شكل الكود احسن ومنظم اكتر .
3. وطبعا الدوال بقت اصغر .

الجزء الى جاى انا الى بقوله مش عم بوب: هو مشرحش الاسكرين عشان كان شرحها قبل كده وكل الى فيها ان خلى ال body بتاع ال try و ال catch فانكشن بيعمها call .

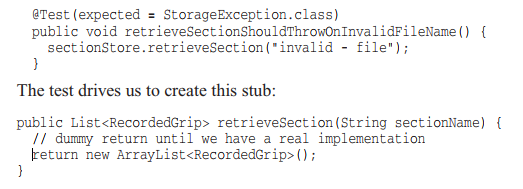




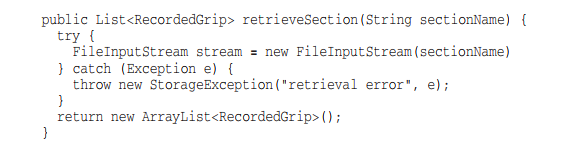
# 2- Write Your Try-Catch-Finally Statement First:

من افضل الحجات ف الexception ان ليه scope يعنى ال body بتاعه بيكون جوه {} . وال try بتكون لوحدها وال catch برضو بتكون لوحدها وال catch لازم تسيب البرنامج ف حاله شغاله وتمام مهما كان الى حصل ف ال try .

تعاله ناخد مثال : ف المثال ده الكود بيعمل access لfile وبيقره شويه serialized object والصورة دى كمان فيها unit test بتقول exception لو ال file مش موجود . طبعا الاسكرين دى مش كامله ف انا قلتلك عم بوب عاوز يعمل ايه.

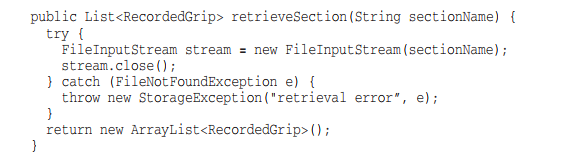


* تانى عشان تفهم الاسكرين الى فوق دى عم بوب بيقلك الى مفروض تعمله بس هى مش كامله هى مفروض بتقدر حاجه من file . طيب دلوقتى لو ال file مش موجود البرنامج هيضرب ويقف . ف عوزين عمل حل احين من كده ف هنستخدام ال exception .
* ده ال refactoring للكود الى فات : كل الحكايه ان ضفت كود ال try/catch وطبعا ال file Stream دة الى بيقره من file ف لو شم موجود هيرمى exception .



* نقدر نحسن الكود الى فات ونحدد نوع ال exception المحتمل الى ممكن يحصل . زى الاسكرين دى كل الى اتغير ان شلت

كلمه exception وخليتها ادق او محدده اكتر FileNotFoundException وده احسن



# 3- Use Unchecked Exceptions:

السكشن ده بيتكلم عن حاجه اسمها Unchecked Exceptions ودى موجوده ف الجافا بس اما الc++ و c#و c مفهاش

ال Exceptions Unchecked ف انا مش هكتب السكشن ده وهو مش مهم من وجه نظرى انا قريته ولقيته مش مهم .

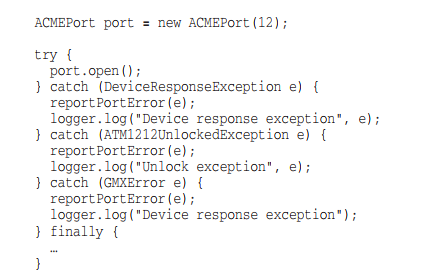
وكل الى فيه بيقلك متستخدامش ال Exceptions Unchecked .والاسباب الى مخليه المضوع مش كويس او مش مفيد

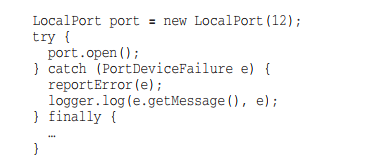
# 4- Provide Context with Exceptions:

كل exception بتعمله throw لازم يبقاله context يعنى سياق بيحدد مكان ومصدر ال error ف الجافا ممكن تشوف ال stack trace .بس برضو ال stack trace مش هيقدر يقلك غرض العمليه الى فشلت.ف الاحسن تعمل error massage ومعبره وبعتهم مع exception وحدد المشكله كانت ايه وايه نوعها . ولو بتعمل logging ابعت معلومات كافيه عشان تقدر تعمل logging ف الcatch .

# 5- Define Exception Classes in Terms of a Caller’s Needs:

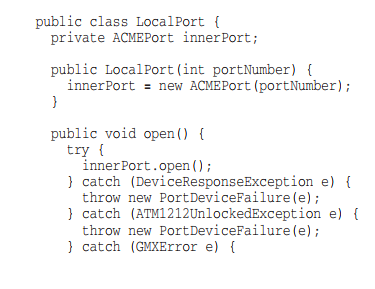
ف طرق كتير لتصنيف ال error ممكن نصنفهم بناء على المصدر يعنى هم من component x او من component y . او ممكن نصنفهم عن طريق نوعهم هلى هم network filer ولا device frailer . بس الاهم من كل ده ان لما نعمل exception class لازم نعرف ازاى نعملهم catch . لو مش فاهم تاله ناخد مثال .



* ف الاسكرين دى احنا بنغطى كل الexception الى ممكن تحصل بس المشكله ان فيه كود متكرر كتير زى
* reportPortError
* logger.log
* وموضع التكرار ده حاجه مش مفاجئه يعنى ف اغلب حلات ال exception handling الى بنعمله بيكون متشابه بغض النظر عن السبب الفعلى.
* ف عشان احنا عرفين ان الى بنعمله متسابه بغض النظر عن ال exception نقدر نبسط الكود خاالص عن طريق ان نعمل كلاس ونحط فيه الAcmePort الى ف الصورة ونادم داله ال open الى جوه كلاس AcmePortجوه الكلاس الجديد اللى هنعمله .
* وده شكل الجديد للكود الى فوق ف واضح اكيد هو ازاى مختصر ومنظم طيب السوال هو الكلاس الجديد اللى هنعمله ده شكله ازاى ؟

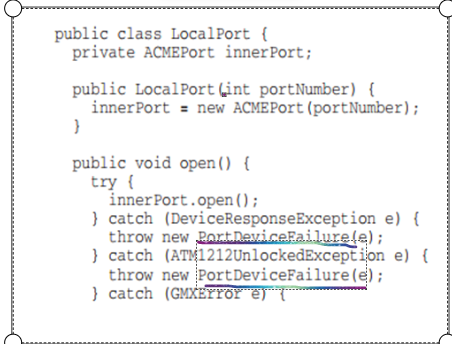
يعنى كلاس localPort جواه ايه ؟

* ده شكل الكلاس الجديد الى هنعمله : هيكون فيه object من ACMEPort . وكمان هيكون فيه داله اسمها open نفس اسم الداله اللى ف كلاس ACMEPort ف داله open الجديده هنادم داله open القديمه ونحطها ف try/catch .



طيب انا كده استفت ايه ؟

* اول حاجه كده عم بوب عمل كلاس اسمه reportDrviceFailure ده بيعمله throw مع كل ال error الى هتحصل . وممكن تعمل اكتر من كلاس وحسب ال exception هتعمل throw لكلاس معين .



* تانى حاجه كده انت خليت الlogic بتاع الكود بتاعك بعيد عن الجزء بتاع ال handling error والكود بقى اصغر واوضح .

# 6- Define the Normal Flow:

لو انت عملت زى ملقنا ف السكشن الى فاتت ف هتعرف تفصل كويس بين business logic وال error handling والكود هيكون احسن بس العمليه دى