

Seminarium 4

Objekt-Orienterad Design, IV1350

Daniel Westerlund

daweste@kth.se

2019-06-02

Innehållsförteckning

1	Introduktion	3
2	Metod	4
3	Resultat	4
4	Diskussion	6

1 Introduktion

Uppgiften är att uppdatera programmet ifrån seminarium 3 med exceptions för alternativ flödet 3-4a. Dvs när ingen produkt med en specificerad produktidentifierare är funnen ska det ske ett exception som meddelar användaren. Samt att simulera ett databas fel när man söker för en hårdkodad produktidentifierare. När view fångar ett exception ska det visas för användaren. Det skall även implementeras en felrapport när ett exception är fångat, antingen som skrivs till en extern fil eller visas i konsolen.

Jag har arbetat själv.

2 Metod

Jag valde att lägga mina exceptions nära kontrollern för enkelt kunna generera ett felmeddelande för användaren. Logghantering valdes att skötas på samma sätt som i kurslitteraturen. Kurslitteraturens metoder för att lösa enhetstestningen med exceptions användes också som grund.

3 Resultat

Klassen ViewDemo simulerar en enkel demonstration utav testet. I den klassen finns referenser till både ErrorMessageHandler och LogHandler som visat och loggar exceptions. Klassen ExternalInventory har ett enkelt, hårdkodad lager som får simulera data från en databas. Där finns en metod, getProductFromInventory(int eancode), returnerar ett Product-objekt om den hittar en sådan i lagret.

I Sale.addProduct(int eanCodeIn) försöker hämta ett Product-objekt med samma eankod som argumentet till objektet är. Om det ej går kommer den först fånga ett InvalidEanCodeException ifrån ExternalInventory.getProductFromInventory(int eanCode) och själv kasta ett CouldNotGetProductFromInventoryException, som i sin tur kommer att fångas av Controller. addProduct(int eanCode), som i sin tur kastar ett MethodFailedException till view-klassen där det loggas och ett felmeddelande genereras till användaren. Figur 1 visar en utskrift ifrån konsolen när programmet anropat klassen ViewDemo. Vid rödmarkering ser man ett felmeddelande som har sköts med hjälp av felhanteringsklasserna. Koden kan hittas på https://github.com/Mrw17/IV1350_Sem4.git.

Very Cool shop Coolstreet 123 12345 cooltown

Current inventory (product, quantity)
{cheese 20.0=15, beer 15.0=30, scratch 25.0=100, Cigarett 45.0=50}
Current cashregister balance: 1000.0

Cashier starts a new sale

Cashier scanning product(cheese)

Show total price 20.0

Cashier scanning product(cheese)

Show total price 40.0

Cashier scanning product(cheese)

Show total price 60.0

Cashier scanning product(cheese)

Show total price 80.0

Cashier scanning product(scratch)

Show total price 105.0

Cashier scanning product(beer)

Show total price 120.0

Cashier scanning product(beer)

Show total price 135.0

Cashier scanning product(cigarette)

Show total price 180.0

Cashier scanning product with eancode that doesnt exists in inventory

2 Jun 2019, ERROR: Product with eancode: 1005 could not be added

Show total price 180.0

No more products to scan

Show total price with discount: 145.0

Show total price with discount + VAT: 170.25

Customer approves

Cashier approves

Customer pays 100.0

You haven't payed enough, still missing: 70.25

Customer pays 100.0

You have payed enough, change is: 29.75

*****Recipe*****

Date: 2019-06-02 11:29:47

Very Cool shop Coolstreet 123 12345 cooltown

cheese 20.0 x 4

scratch 25.0 x 1

beer 15.0 x 2

Cigarett 45.0 x 1

Total price: 170.25

Total Discount 9.75

Total VAT: 25.25

Paid: 200.0

Change: -29.75

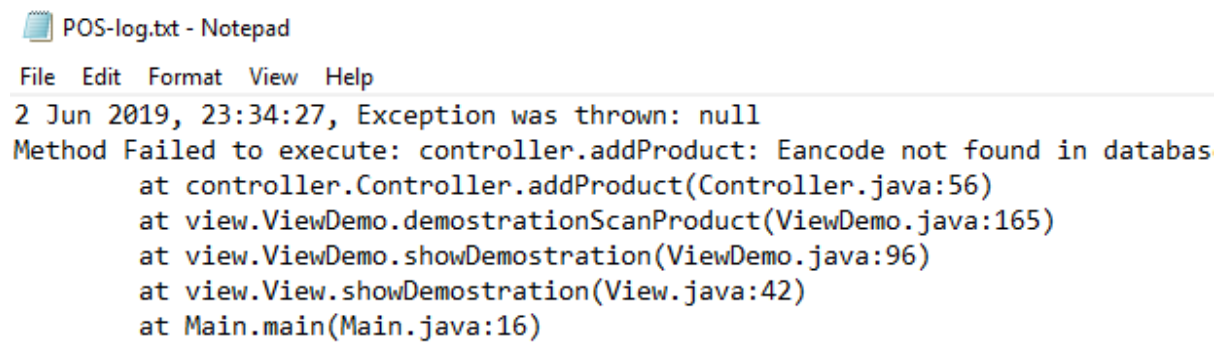
*****Recipe*****

Current inventory (product, quantity)

{cheese 20.0=11, beer 15.0=28, scratch 25.0=99, Cigarett 45.0=49}

Figur 1 Testkörning av programmet

Figur 2 visar det felet resulterade i för felmeddelande i loggen, som sparas skrivs till en .txt-fil.



```
File Edit Format View Help
2 Jun 2019, 23:34:27, Exception was thrown: null
Method Failed to execute: controller.addProduct: Eancode not found in databas
    at controller.Controller.addProduct(Controller.java:56)
    at view.ViewDemo.demonstrationScanProduct(ViewDemo.java:165)
    at view.ViewDemo.showDemonstration(ViewDemo.java:96)
    at view.View.showDemonstration(View.java:42)
    at Main.main(Main.java:16)
```

Figur 2 – Skärmdump av loggen efter testkörning av ViewDemo

4 Diskussion

Väldigt mycket kastande av exceptions fram och tillbaka som verkligen skulle underlättas om man hade byggt in det från början, framför allt när man inte har ett hundra procent koll på hur det fungerar så vart det mycket kompillerings fel överallt så får man lag till något exception.

Loggern hanteras genom `private void handleException(String uiMsg, Exception exc)` i view klassen. Där `uiMsg`, som är ett förenklat felmeddelande skickas till användaren och exception skickas till `LogHandler`, som skriver ut felmeddelande i en log fil med alla andra exceptions som uppstod under exekveringen. Även fasts detta program är relativt litet kan jag redan uppskatta att man till nästa projekt har en grundligt genomtänkt plan med framför allt en bra SSD, som jag tror förenklar och snabbar på kodandet, samt att man börjar att koda med exceptions direkt, tror jag även skulle göra det enklare med felhantering, även fast enhetstesterna var tidssparare tror jag det kan vara ännu mera utav en tidssparare.

Tyvärr fanns inte tiden till för att implementera Obersver-pattern för del 2, det hade varit mycket intressant.