https://github.com/Filipjacobson/POSSystemSem4
Choose between checked and unchecked exceptions.
• Use the correct abstraction level for exceptions.
Seams ok
• Name the exception after the error condition.
Seams a little strange that in Controller row 53, catch a OperationFailureException and throws a new OperationFailureException, shouldn't it throw another exception for better abstraction?

```
catch (OperationFailureException ofe)
{ throw new OperationFailureException(ofe.getMessage()); }
```

• Include information about the error condition.
Shows information about error for the user.
• Use functionality provided in java.lang.Exception
Seams ok.
• Write javadoc comments for all exceptions.
Seams to be ok.
• An object shall not change state if an exception is thrown.
Seams to be ok as fair as I can see.
• Notify users.
yes
• Notify developers.
Yes
• Write unit tests for the exception handling.
ItemRegistryTest, tests it.

Write unit tests for the exception handling.
Exists.

Could not find any pattern or anything about observer.

Discussion seams relevant and result are shown with pictures and explained decent.

https://github.com/Niclas130/IV1350-Objektorienterad-design/tree/master/Seminar%203

Choose between checked and unchecked exceptions.
• Use the correct abstraction level for exceptions.
Seams ok
• Name the exception after the error condition.
Seams ok.
• Include information about the error condition.
Seams ok.
• Use functionality provided in java.lang.Exception
Seams ok.
• Write javadoc comments for all exceptions.
Seams to be ok.
• An object shall not change state if an exception is thrown.
Seams to be ok as fair as I can see.
• Notify users.
yes
• Notify developers.
Yes
• Write unit tests for the exception handling.
ItemRegistryTest, tests it.

Do you agree with choices of observer and observed object?
Seams ok
• How is the reference to observer passed to observed object?
As an object

Does this have bad effects on coupling or cohesion?
Don't think so.

• What data is passed from observed object to observer? Does this brake encapsulation of the observed object?
All data about object. Not sure if it break encapsulation then?

Is the method and result explained in the report? Is there a discussion? Is the discussion relevant?
Result is clearly explained and the discussion seams to be relevant to the problem.

https://github.com/lunjoa/PointOfSaleAssignment/tree/master/src/main

• Choose between checked and unchecked exceptions.
Seems ok

• Use the correct abstraction level for exceptions.
Throws same exception trough multiple classes, which makes it low abstraction.

• Name the exception after the error condition.
Seams ok.
• Include information about the error condition.
A little vague.

• Use functionality provided in java.lang.Exception
Seams ok.

• Write javadoc comments for all exceptions.
Seams ok.

• An object shall not change state if an exception is thrown.
Seams ok

• Notify users.
Seams ok, console.

• Notify developers.
Seams ok, logging.

• Write unit tests for the exception handling.
Tests exceptions.

• Are pattern implementation(s) really correct implementations of the pattern(s).
Sometimes it seems like this at a _rst glance even though there are severe aws.

• Are design patterns explained and motivated?
Explained but not really motivated.

• Do you agree with choices of observer and observed object?
Seams ok

• How is the reference to observer passed to observed object? Does this have bad effects on coupling or cohesion?

• What data is passed from observed object to observer? Does this brake encapsulation of the observed object? Sale object don't think it breaks encapsulation.

• Is the method and result explained in the report? Is there a discussion? Is the discussion relevant?
Result is explained in detail with good pictures and words.
Discussion is relevant, but minimal.