Seminar 2 Assessment

2024-04-18

Ayda Ghalkhanbaz (aydag@kth.se)
Emma Lindblom (ekali@kth.se)
David Wrang (sdwrang@kth.se)

Alice Ljungström

- **Is it easy to understand the design?**
  connections(arrows) in the class diagram are a bit unclear, specifically, between main and integration layer. (it is however clear in the startup communication diagram).

- **Are the MVC and Layer patterns used correctly? Are there enough layers? Enough packages? Enough classes? Does the controller handle any logic (it should not)?**
  MVC and the layer patterns are followed correctly and the controller doesn't handle any logic.
  Model could have more classes for example for handling payment

- **Is there view-related code in the model (there should not be any)?**
  No view-related code found in the model!

- **Is there low coupling, high cohesion and good encapsulation? Is the use of low coupling, high cohesion and encapsulation explained in the report?**
  Sale could be divided into more classes such as payment, to achieve higher cohesion. We believe there is low coupling, however there aren't too many classes so that should be kept in mind if adding more. In the communication diagrams there are almost spider in the web tendencies, one solution could be to divide them one step more. Good encapsulation with only having package private where classes dont interact outside of package.

- **Are parameters, return values and types correctly specified? Is it clear from where values come?**
  In the startup communication diagram, a stereotype (create) is missing when main creates an instance of classes controller, view, etc.
  In the payment communication diagram, there is a return type receiptAndChange which is unclear what this type of object does or by which class it is constructed. (a comment/note could also be enough)

- **Are objects used instead of primitive data as much as possible?**
  There are logical numbers of objects along with primitive data types.
- **Are there static methods or attributes? If so, is that a mistake?**
  no
- **Objects shall be kept intact as much as possible. Is that the case, or do objects hand out their attributes?**
  Seems good

- **Are Java naming conventions followed?**
  Name conventions seem good

Samuel Brodin

- **Is it easy to understand the design?**

  Easy to follow the design with good namings.
- **Are the MVC and Layer patterns used correctly? Are there enough layers? Enough packages? Enough classes? Does the controller handle any logic (it should not)?**
  MVC and layer patterns are used mostly correctly in all diagrams. Why does view connect with a class in model? No layer is missing. Good amount of classes. Controller doesn't handle any logic.

- **Is there view-related code in the model (there should not be any)?**
  No

- **Is there low coupling, high cohesion and good encapsulation? Is the use of low coupling, high cohesion and encapsulation explained in the report?**
  Encapsulation seems good. Coupling seems low. Relatively high cohesion, perhaps sale could be divided into more classes for even higher cohesion.

- **Are parameters, return values and types correctly specified? Is it clear from where values come?**
  No constructors in main, other than that, it seems fine.

- **Are objects used instead of primitive data as much as possible?**
  Good use of DTOs and other relevant objects instead of too much primitive data. Could perhaps use even more objects. A lot of methods in the communication diagrams return a lot of primitive data.

- **Are there static methods or attributes? If so, is that a mistake?**
  No

- **Objects shall be kept intact as much as possible. Is that the case, or do objects hand out their attributes?**
  Why is there a receipt attribute and a receipt class. Does the objects stay intact with this? Other than that objects seem to be kept intact

- **Are Java naming conventions followed?**
  Naming conventions are followed.

Ammar Alonzo

- **Is it easy to understand the design?**
  Arrows are missing between item and PoS, and item and DTO.

- **Are the MVC and Layer patterns used correctly? Are there enough layers? Enough packages? Enough classes? Does the controller handle any logic (it should not)?**
  Should DTO really be its own package/layer, or could it be a part of model or integration? Other than that MCV and layer patterns seems followed correctly. No layer is missing. Good amount of classes. Perhaps calculate change could be considered controller handling logic.

- **Is there view-related code in the model (there should not be any)?**
  no

- **Is there low coupling, high cohesion and good encapsulation? Is the use of low coupling, high cohesion and encapsulation explained in the report?**
  Coupling seems low if you leave out the data layer, as was allowed in the task. High cohesion seems good. Encapsulation seems good for this diagram, no package private or private methods would fit as of now.

- **Are parameters, return values and types correctly specified? Is it clear from where values come?**
  Is the best choice to update a saleDTO or would externalInventorySystem be a better choice? Attributes in a DTO cannot be changed after creating them.

- **Are objects used instead of primitive data as much as possible?**
  Seems like a good amount of objects instead of primitive data. Is it meant to be as many void functions as there currently are?

- **Are there static methods or attributes? If so, is that a mistake?**
  No

- **Objects shall be kept intact as much as possible. Is that the case, or do objects hand out their attributes?**
  Objects seems to have been kept intact.

- **Are Java naming conventions followed?**
  Naming conventions are followed