Coding task

Chosen challenge: Problem statement 2: Business Rules Engine.

Goals:
- OOPS, cleans coding SOLID and best practices
- implementation using TDD
- GIThub as repository

This task seems to be oriented towards a developer within this development stage only.
As I am applying for Lead developer with some architecture, I would like to demonstrate an approach which is covering not only the development effort, but also the organizational and planning stage

Step 1. - The plan
To plan a project following is need as minimum:

a. agreed requirements with the stakeholders at least in some stage, which can change during an iterative process. The challenge provides this information.

b. The team needs coding standards, repository, development process.
Coding standards – I will focus on using explanatory names, using design patterns, using best practices. Pair coding, code review, …, ensuring the coding standards and providing overall code quality cannot be done in this project.
Repository is given by the project
Development process. So the only required process here is the TDD assuming an iterative development. I will also add an architecture of the solution prior any implementation to identify the interfaces, data flow, …
A professional system will typically require a build machine, so typical local developer machine issues are avoided and the full build is documented and can be replicated in another environment. This is the one producing the actual output.
Note about TDD: The first step in TDD approach is to demonstrate a failure by not implementing the code. In a professional environment this will be less desireable, as breaking the build will not demonstrate the fulfilment as the test steps cannot run either. I would use another approach, which is supported by many compilers today and that is to throw a “not implemented” exception to fail the test step and not the build.

c. The plan for how to integrate into systems, which this system is going to integrate to, both as data input and data output, shall be prepared. It is assuming connection to external systems.

d. an infrastructure this solution is going to run on. TBD.

e. the data safety and product security. Here we assume that everything is safe and secure.

f. maintenance of such solution. TBD

Step 2 - The analysis/decisions (normally part of an Architecture

Data input:
Orders assuming to reside in an external system (probably a DB table). Each order has minimum one OrderItem associated with payment for an item part, probably several (rows).

Missing information: I would expect an order have also a orderer associated for verifying a providing an information for the membership, packing slips, etc. especially if the task is to add new parts.
Decision: as the time frame given for solution is limited to 1.5 hours, the system will not pass "orderer" information to the other systems.

The description also refers the owner of a membership. I guess that we cannot just assume that the owner is the same as the "orderer" for B2B.
Decision: Any owners must be managed by the membership system. The payment must contain a unique identification of the membership transaction.

Described (input) **Order Items types**:
Physical product (PP), book (book can also be assumed as PP), membership (not a PP), upgrade to an existing membership (not a PP, but requires a link to existing membership), video (PP),

The assumption is that it is possible to "serialize" the OrderItems to internal OO representation. This is not part of the project.

Unclear definitions:
Physical product is any physical product including book and video (I assume that we are not talking about streaming). So the last sentence is unclear:
If the payment is for a physical product or a book, generate a commission payment to the agent.

Decision: The commission will be generated also for the video, as this is not clearly defined.

Optional access, as we are checking specific products, we should have access to a product database. Eg. if "Learning to Ski" is replaced by "Learning to ski 2", so we don't need to reprogram the whole solution.
Decision: Not implemented

The question here is whether the system should also verify the validity, or we can assume that the systems before has validated the data input. E.g. is upgrade to a membership valid at all – did the membership exists before.
Decision: as the time frame given for solution is limited to 1.5 hours, assume that the system should not validate any input! The validation of an action will be the responsibility of the output system, failing the processing of the order.

Required actions (output):
packing slip, packing slip targeting royalty dep., update membership system, e-mail system, agent system with commission.

Output systems:
Shipping department, royalty department, membership system, agent system.

Step 3 - The Architecture

The architecture is affected by the decision made above and the time frame. Please note that all the external system are all placed to the right. The data input order system was also placed there to fit better to A4 format.

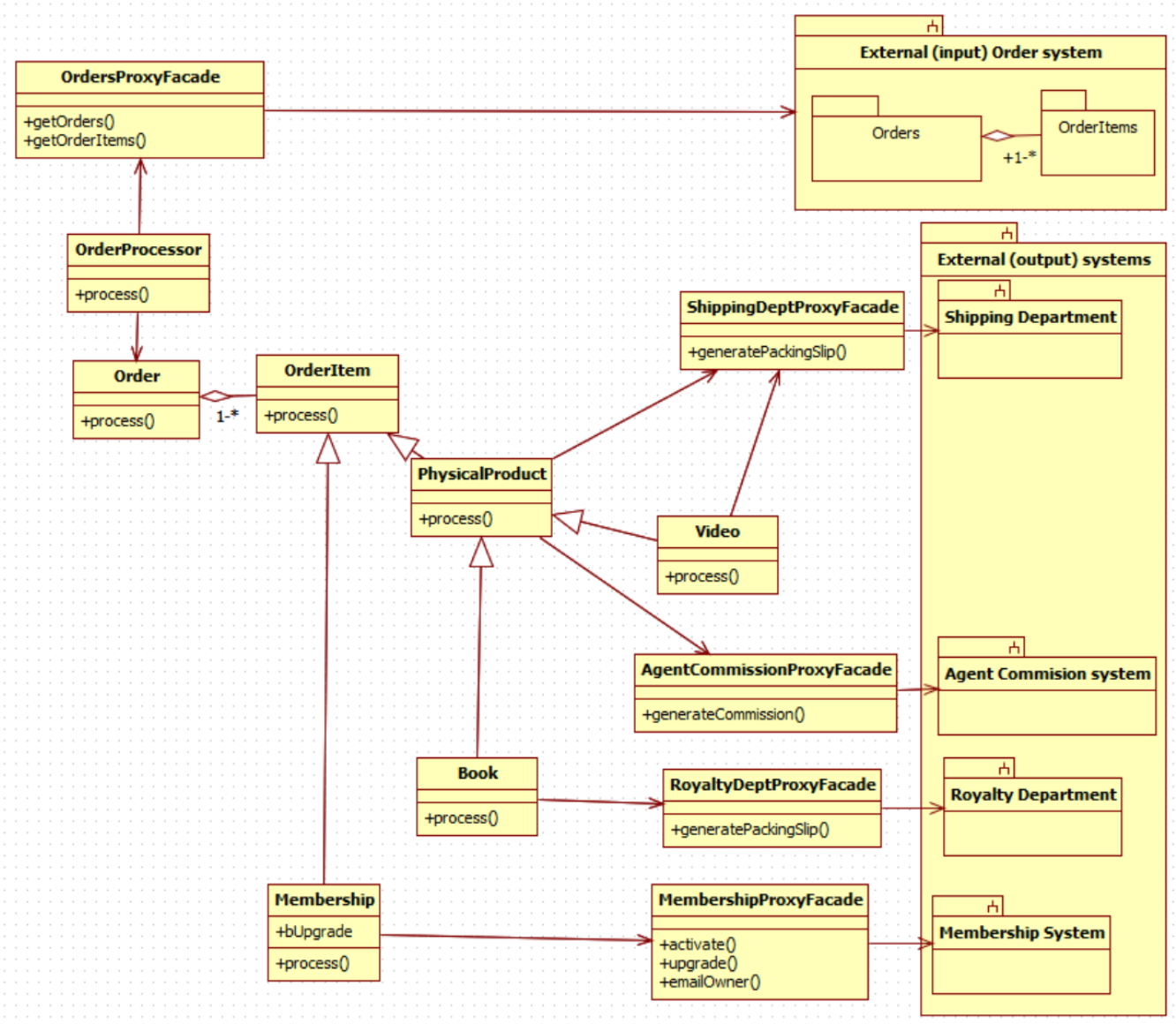

Figure 1: The solution architecture

The solution is using Facade to implement connection to external systems. It will be a singleton design pattern for simplicity. The arrows show dependencies (and not the data flow as it could be expected).

The OrdersProxyFacade is not implemented as OO, because the assumption is that this type of information usually originates from a database system modeling the association with master/detail table.

Counting for testing
For testing purpose the different proxy facades inherit from one common ProxyFacade super class.

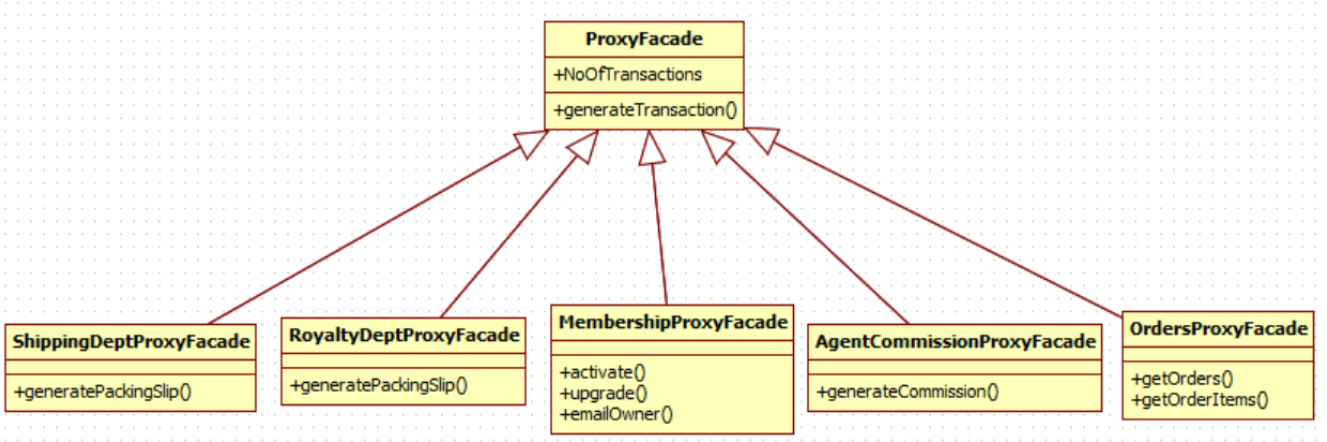

Figure 2: Proxy facade class diagram

Discussion
The system could also be modeled as rule based design, where the rules could be parameters to the OrderItems to create an universal business rule engine. However the time frame puts a limit for designing solutions of this magnitude. It's a decision about "time to marked", maybe not the best, but realistic solution. This design, thus limited, still fulfills the requirement to the design being open to extensions.

Step 4 – Implementation of the solution
Demonstrated in the Github.