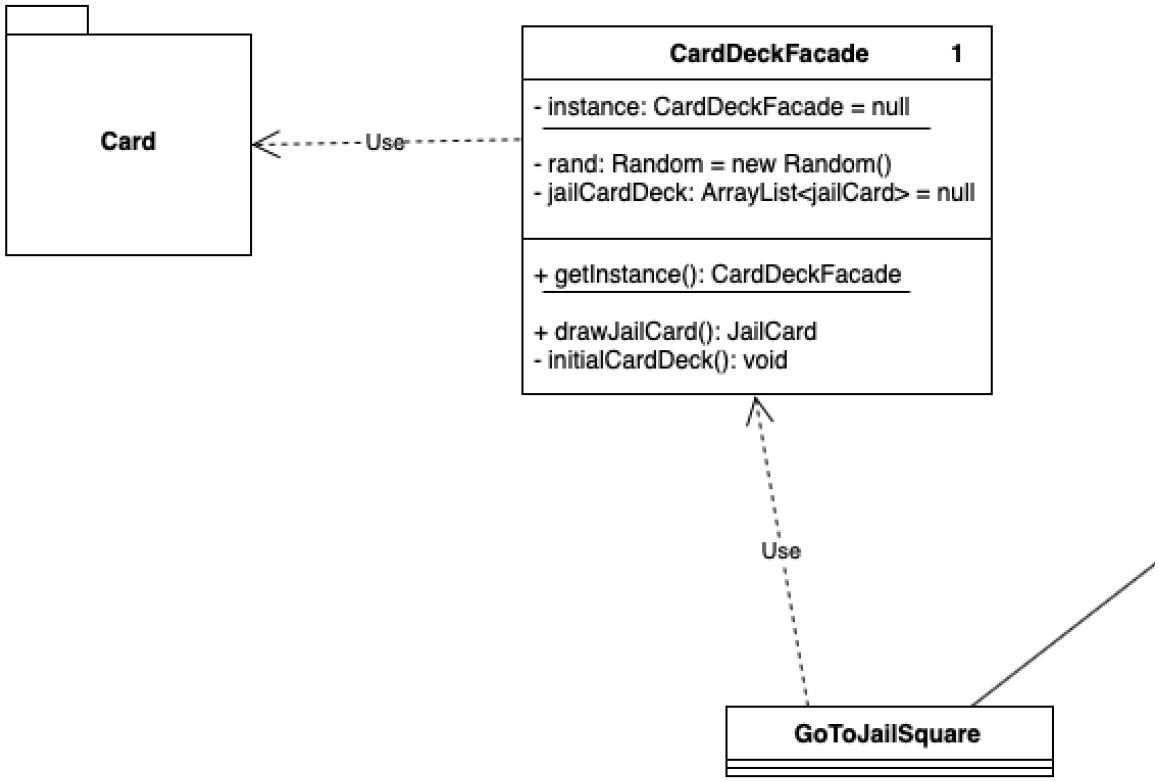
SWEN30006 Software Modelling and Design
Workshop 9: GoF Patterns (Part 3)
Facade, Observer and Decorator

School of Computing and Information Systems
University of Melbourne
Semester 2, 2020

The complete solution diagram will be attached at the end of the solution.

Part 1 Adding LuckCards

Please find the partial diagram for part 1 attached below.

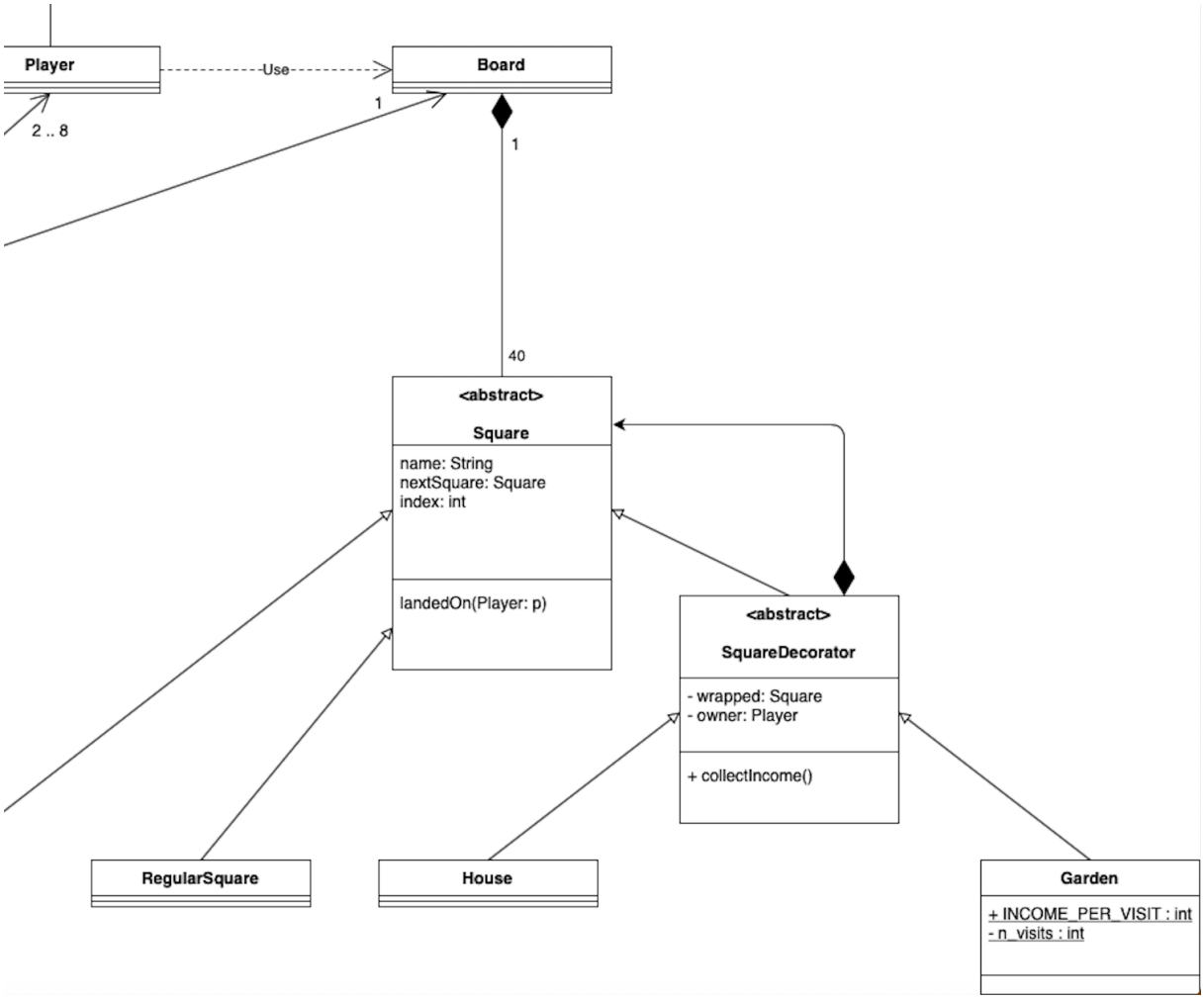


The Facade pattern is appropriate for the implementation of LuckCards, as the CardDeckFacade can take on responsibilities such as instantiating cards and drawing from cards. As a result, interactions with

all the cards are masked by the Facade to ensure minimal coupling between subsystems. This might be a somewhat trivial example, but you can imagine how the impact of the Facade pattern will be much greater in more complex system.

Part 2 Building a House

Please find the partial diagram for part 2 attached below.

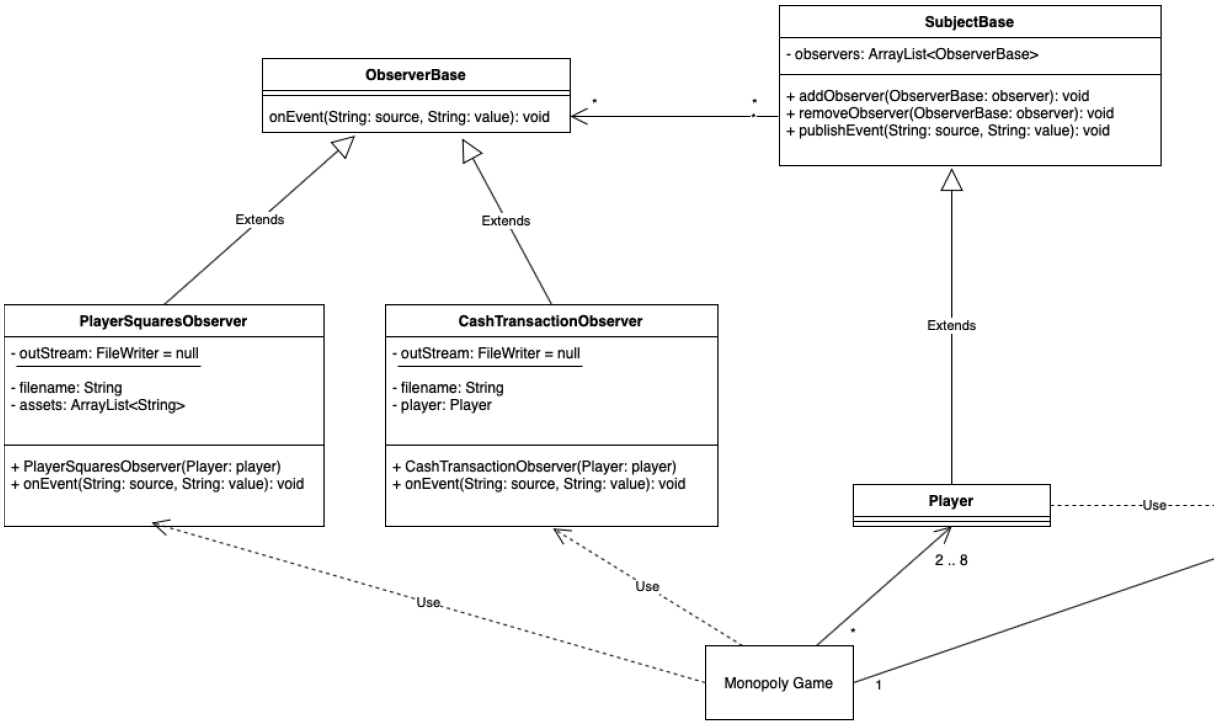


The Decorator pattern is appropriate, as we would like to enhance the behaviour of the RegularSquare at runtime with minimal changes to the code. Please bear in mind that for the decorator pattern to work, we can use either an interface OR an abstract class (Square in our case) at the top, as we just desire the polymorphic behaviour and don't mind if it's an interface or an abstract class at the top.

Another benefit is that should rules change (e.g. build two houses and three gardens on each square), we can very easily achieve this effect with minimal change to the code.

Part 3 Logging Player's Assets (Extra)

Please find the partial diagram for part 3 attached below.



It's very apparent that the Observer pattern should be used in this case. Please also bear in mind that you are welcome to use the inbuilt Observable class from java when you apply the Observer pattern yourself and the Observer pattern is used here to give you insight regarding how things work under the hood.

Part 4 Full Diagram