Task 1

1. The adapter design pattern is related to polymorphisms, indirection, and protected variation. Adding a level of indirection to different APIs in other components with polymorphism. It supports Protected Variation with respect to changing external interfaces using an Indirection object that maintains interfaces and polymorphism.
2. Facade is a way to achieve Protected Variations.
3. Factory allows to introduction of object caching or recycling, and it is related to Pure Fabrication and High Cohesion.
4. Strategy defines a family of algorithms encapsulate each one and is related to Polymorphism, Protected Variations and Low coupling.
5. Composite represents part-whole hierarchies and is related to Protected Variation and Polymorphism.
6. Observer is based on Polymorphism and provides Protected Variation in terms of protecting the publisher from knowing the specific class of object, and number of objects, that it communicates with when the publisher generates an event.

Task 2

1. I would say RuleEngine should be assigned the responsibility of taking care of saving and loading the game's state. Because it knows the type of game for example thus the dimensions of the Board, what items can be on the board. Thus, it has the initializing data for the Board. The RuleEngine class has the information necessary to fulfill the responsibility through the Information Expert who has a responsible for assigning responsibilities to object, to distribute behavior across classes.

Task 3

We can take adapter and façade that can work well in this situation. We can take adapter because it adapts one interface for another and intended for this kind of application in our case a game which is written to use a one interface. We can take façade which brings us a simple interface to a complex system. Then it will be possible to reduce time and effort when it comes to changing and adding new features and functions to the game