**Design Patterns – Rules Pattern**

The Rules Pattern is a useful tool for when you encounter a problem that requires a large amount of if/else conditional logic. These scenarios can easily result in lengthy blocks of code where it’s difficult to debug, test, modify, or maintain the logic. The Rules Pattern provides a way to split this logic out into separate classes that follow best practices such as SOLID and DRY.

The Rules Pattern consists of the several components:

* An Interface defining a generic Rule, consisting of
  + A “run rule” method accepting inputs to be evaluated and the data return type
  + (Optional) A method or property defining the Rule’s priority (for cases where rules need to be evaluated in a particular order)
  + (Optional) A method to evaluate whether the rule applies (and therefore should be run) or does not apply.
* Classes implementing the Rule Interface
* A Rules Engine, consisting of
  + A constructor (typically accepting the list of rules to use)
  + A list of available rules
  + An evaluation method that accepts data, processes the data using the available rules, and returns a result

**Rules Pattern Kata**

In this Kata, we will be refactoring a solution to the Greed Kata using the Rules Pattern. You can find the Greed Kata instructions in the Design Patterns solution items.

The Start branch contains an implementation of the Greed Kata that should be refactored using the Rules pattern. The Hint branch contains a sample Rules interface, implementing class, and Rules Engine. Finally, the CompletedSolution branch contains a completed refactoring of the Greed Kata.

**Limitations/Caveats**

The Rules Pattern is a powerful tool. However, you should take care when using it to make sure it fits your use case. The Rules Pattern works best when Rules are not at risk of changing the state of the system in such a way as to alter the behavior of other Rules. In other words, running Rule A should not cause side effects that alter the outcome of Rule B – the behavior of both rules should depend strictly on the data they’ve been passed by the Rules Engine. If the behavior of the available rules cannot be decoupled, then the Rules Pattern may not be a good fit for your problem.