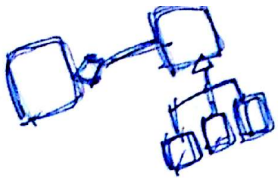
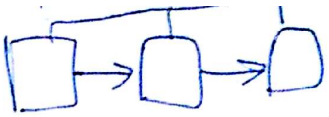
Strategy / Command.





OOSE Final Exam 6/9/2017

1. Please indicate the design pattern of the following description. 10%

Answer

(1) ● Use sharing to support large numbers of fine-grained objects efficiently. → prototype

(2) ● Provide an interface for creating families of related or dependent objects without specifying their concrete classes. → Abstract Factory.

(3) ● Define a family of algorithms, encapsulate each one, and make them interchangeable. The pattern lets the algorithm vary independently from clients that use it. → Strategy (?)

(4) ● Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later. → Memento

(5) ● Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation. → proxy?

(6) ● Attach additional responsibilities to an object dynamically. The pattern provides a flexible alternative to subclassing for extending functionality. → Decorator.

(7) ● You want to decouple event handlers from an event sources so handlers can be added or removed without impacting the event source. → chain of Responsibility (? Bridge

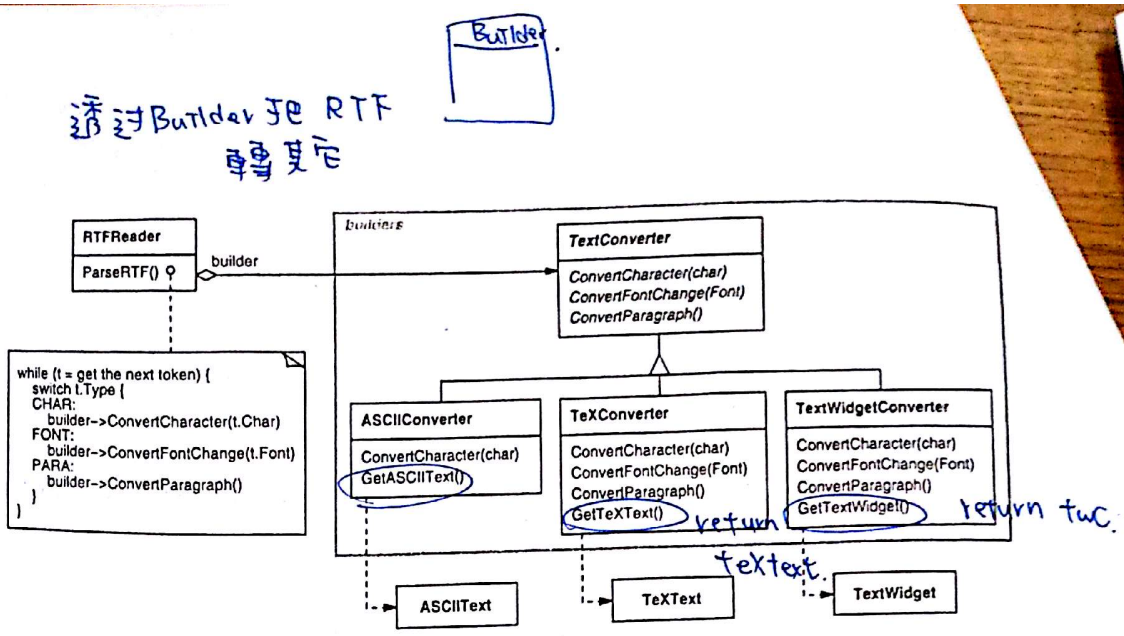
(8) ● Represent an operation to be performed on the elements of an object structure. This pattern lets you define a new operation without changing the classes of the elements on which it operates. → Command

(9) ● Provide a centralized communication medium between different objects in a system. → Mediator.

(10) ● Separate the construction of a complex object from its representation so that the same construction process can create different representations → Builder.

轉 RTF 格式 到 各式格式

2. We want to create a reader for the RTF (Rich Text Format) document exchange format which should be able to convert RTF to many text formats. The reader might convert RTF documents into plain ASCII text or into a text widget that can be edited interactively. The problem, however, is that the number of possible conversions is open-ended. So it should be easy to add a new conversion without modifying the reader. The diagram is shown as follows.



A solution is to configure the RTFReader class with a TextConverter object that converts RTF to another textual representation. As the RTFReader parses the RTF document, it uses the TextConverter to perform the conversion. Whenever the RTFReader recognizes an RTF token (either plain text or an RTF control word), it issues a request to the TextConverter to convert the token. TextConverter objects are responsible both for performing the data conversion and for representing the token in a particular format.

Subclasses of TextConverter specialize in different conversions and formats. For example, an ASCIIConverter ignores requests to convert anything except plain text. A TeXConverter, on the other hand, will implement operations for all requests in order to produce a TeX representation that captures all the stylistic information in the text. A TextWidgetConverter will produce a complex user interface object that lets the user see and edit the text.

Each kind of converter class takes the mechanism for creating and assembling a complex object and puts it behind an abstract interface. The converter is separate from the reader, which is responsible for parsing an RTF document.

Please implement it in Java code. **40%**

(1) method 是哪种 (6 method)

mix two Pattern (2) 写 Java code. 混合 factory Method.

(3) Implement Create 3种 product.

3. The template method pattern can be applied to improve the database access commands in the persistence framework. The following figure shows how to accomplish this. Based on the given diagram, please specify which shall be concrete operation, primitive operation, factory method and hook operation? Why do you think so? **10%** Please implement the template method in the example in Java code. **30%** Finally, please implement the factory method to vary three subclasses to create different types of products. **10%**

template.