Recursive Coordination Pattern

Recursive Coordination is learn from Chain Of Responsibility and Command Pattern.

Chain Of Responsibility：Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. **Chain the receiving objects and pass the request along the chain** until an object handles it.

Command Pattern：**Encapsulate a request as an object**, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.

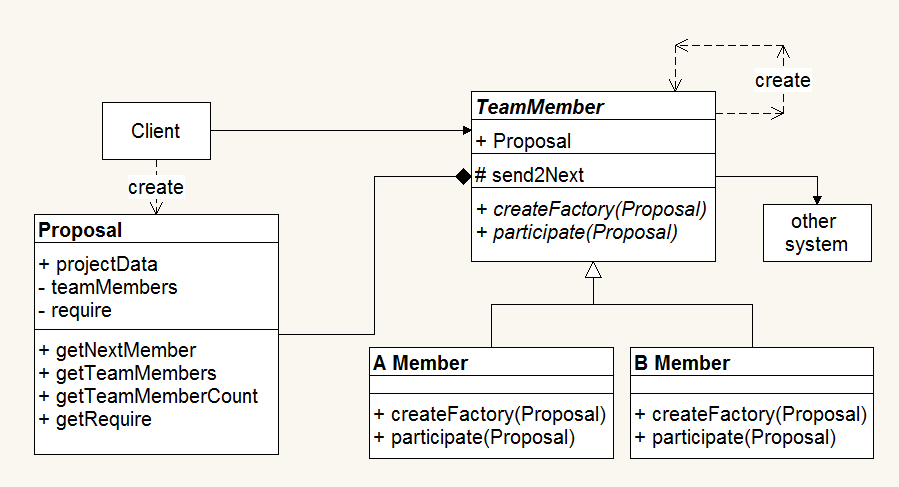
Recursive Coordination Pattern：**Encapsulate a request as an proposal object** and **choose one or more object(TeamMember), give them chance to participate the proposal request**. Final, **TeamMembers will return proposal** objects that Processed.

Scenes：

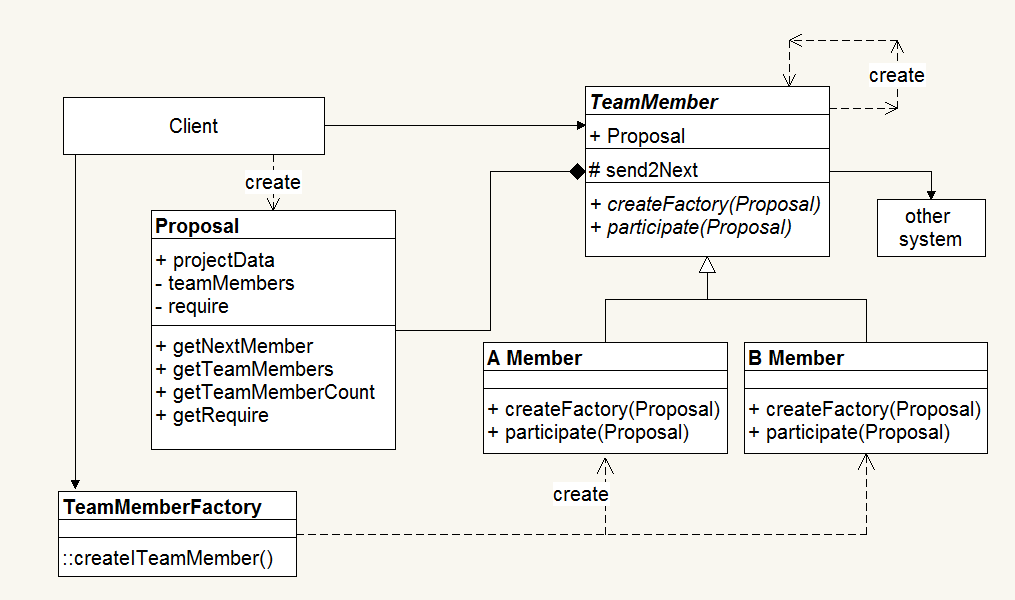
The Scenes that use Recursive Coordination Pattern is client send a request that it know which one or more objects should handles this request together, but the way those objects how to handle requests is hinges on some system that client don't know or can't get (ex : itself private method or other system).

UML(object model diagram)：

createFactory method exist because construct can't return object but self, so need it to new next member to participate the proposal and return the processed proposal.



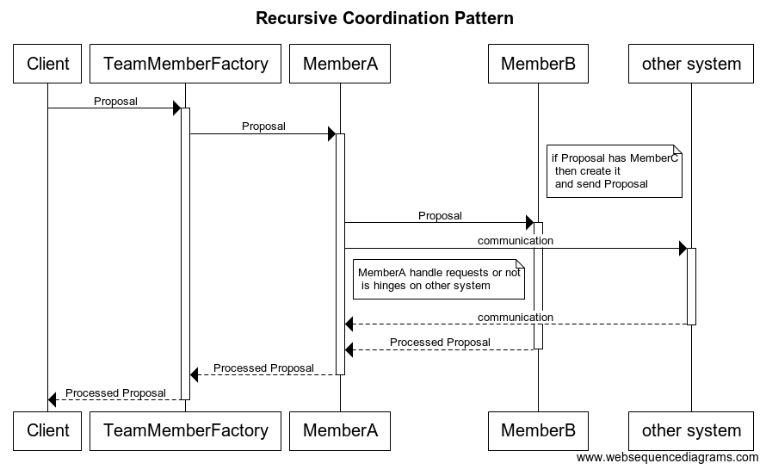
In real case createFactory method usually use a static factory to new members.



UML(timing diagram )：

It timing diagram is similar to Chain Of Responsibility. The difference are

1. member create(new) next one by itself createFactory and send proposal to it.
2. the next one is who that is setting by proposal
3. the processed proposal must return to client
4. members handle requests or not isn't fixed



Author：MazeR <mazer0701@gmail.com>