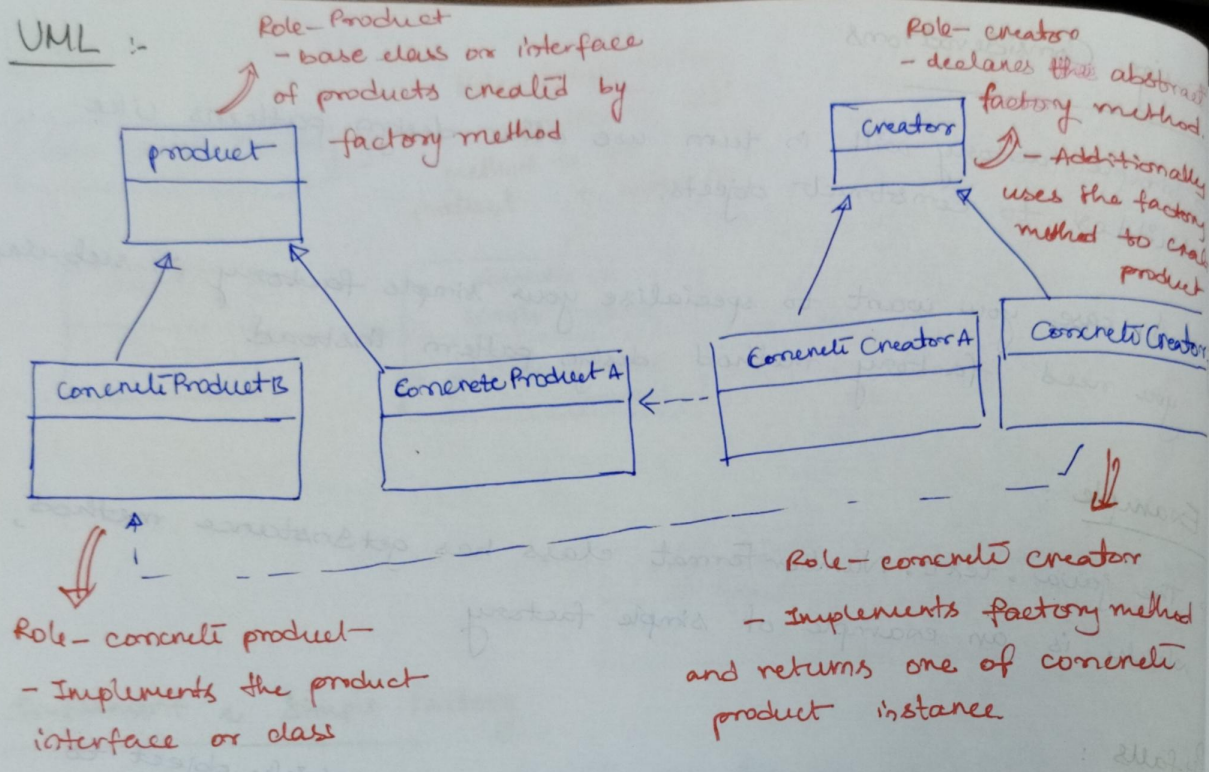
Factory :-

What is a Factory Method ?

- We want to move the object creation logic from our code to a seperate class.
- We use this pattern when we do not know in advance which class we may need to instantiate beforehand & also to allow new classes to be added to system and handle their creation without affecting client code
- We let subclasses decide which object to instantiate by overriding the factory method.

UML :-

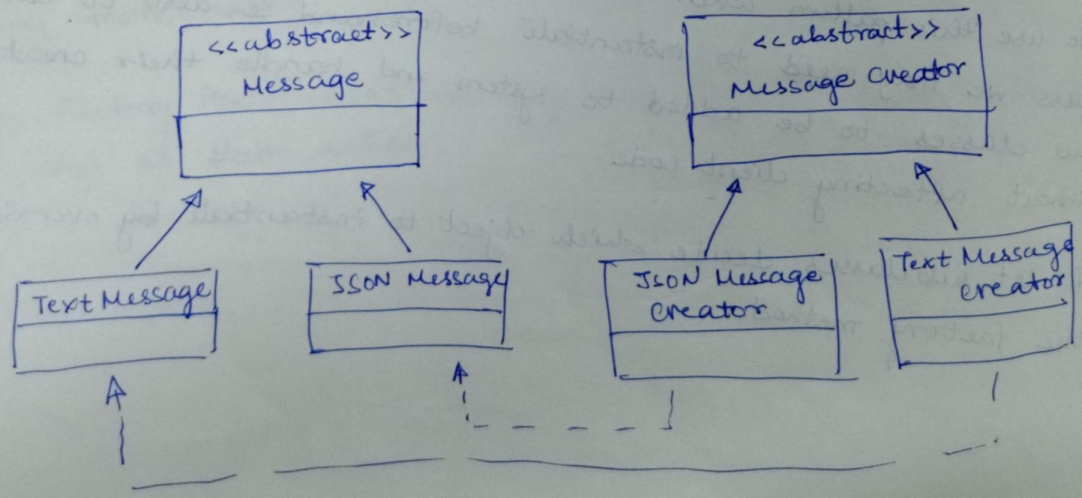


Implement a Factory Method

- We start by creating a class for our creator
  - Creator itself can be concrete if it can provide a default object or it can be abstract.
  - Implementation will override the method and return an object.

Example UML :-

class FactoryMethodEx



Implementation Considerations

- The creator can be a concrete class to provide a default implementation for the factory method. In such cases you'll create some "default" object in base creator.

- You can also use the simple factory way of accepting additional arguments to choose between different objects types. Subclasses can then override factory method to selectively create different objects for some criteria.

Design Considerations

- Creator hierarchy in factory method pattern reflects the product hierarchy. We typically end up with a concrete creator per object type.

- Template method design pattern often makes use of factory methods.

- Another creational design pattern called "abstract factory" makes use of factory method pattern

Examples :-

- The java.util.Collection has an abstract method called iterator(). This method is an example of factory method.

- Remember, the most defining characteristic of factory method pattern is "subclasses providing the actual instance". So static methods returning object instances are technically not GoF factory methods.

Pitfalls

- More complex to implement. More classes involved and need unit testing.
- You have to start with Factory method design pattern from the beginning. It's not easy to refactor existing code into factory method pattern.
- Sometimes this pattern forces you to subclass just to create appropriate instance.