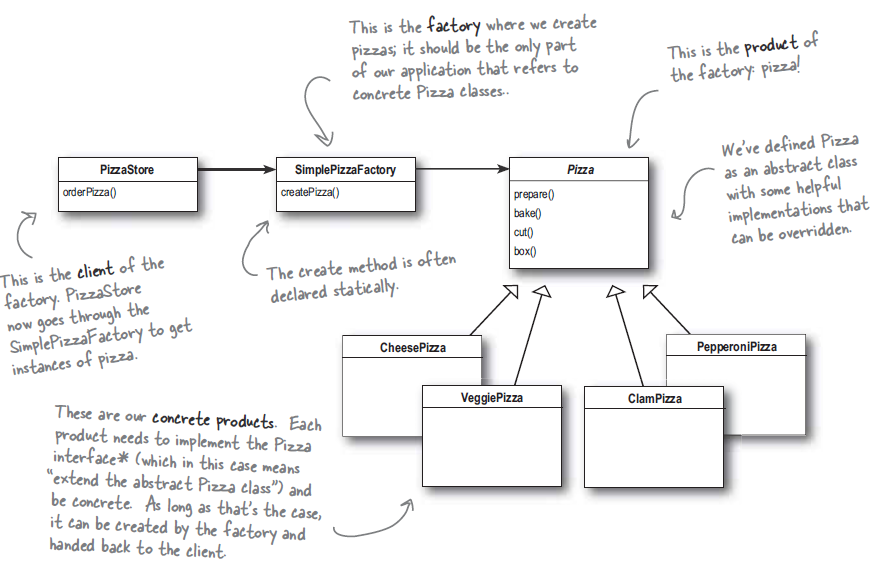
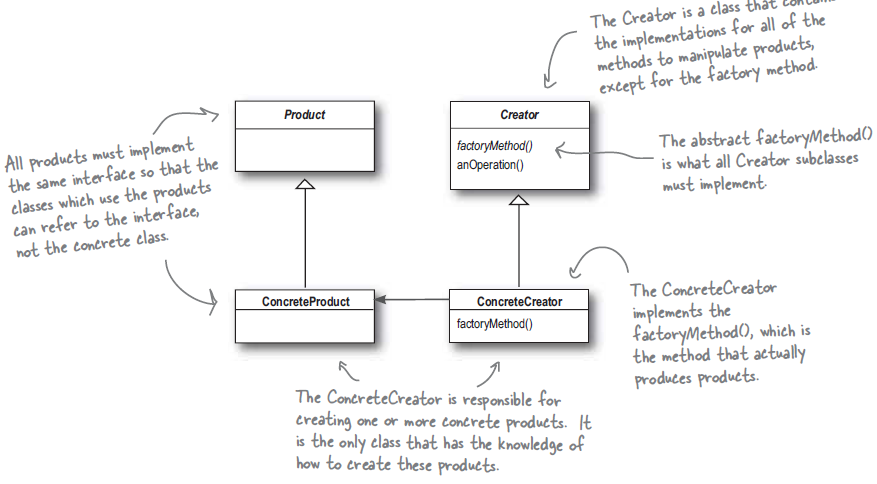
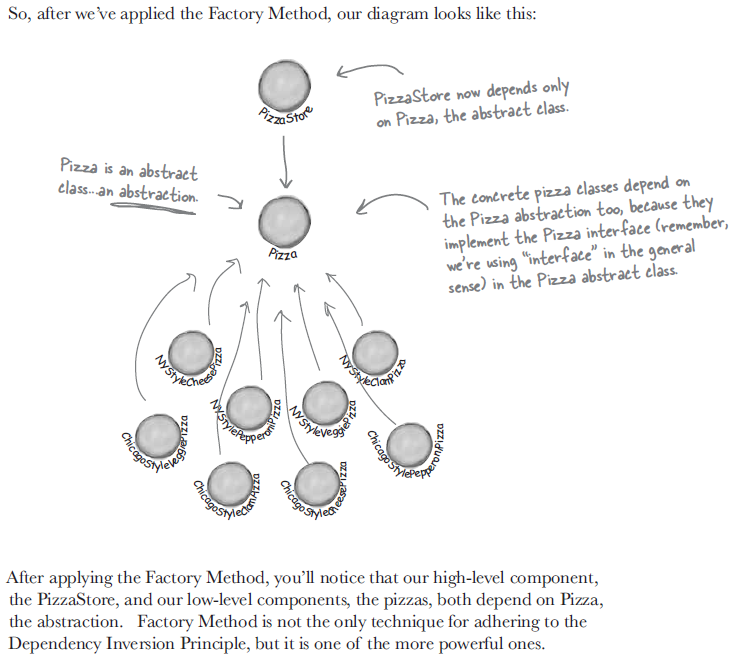
1. **Simple factory pattern**: It’s programming idiom which decouple object creation from its use by placing the creation logic in different object and then composing that object in the object that using it.
   1. In simple factory, factory method can be static called static factory. But we won’t able to extend it to modify object creation logic.



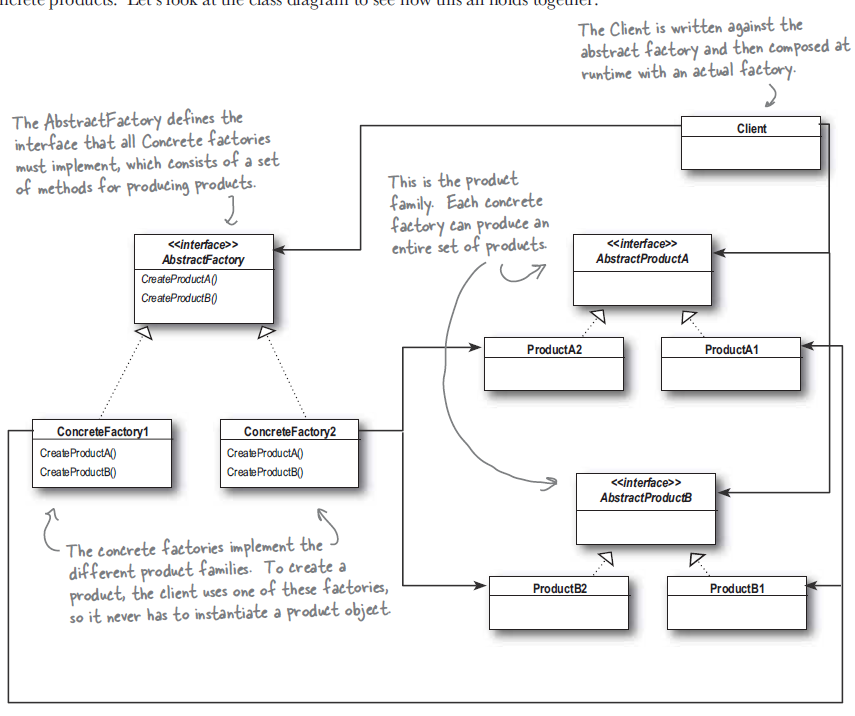
1. **Definition**: Factory patterns encapsulates the process of object creation.
   1. **Factory Method Pattern**: This pattern encapsulates the object creation by letting subclasses to decide what object to create.



* + 1. Abstract creator gives an interface with method to create object aka “factory Method”. Other methods written in the abstract creator supposed to work on the product created by “factory Method”. Subclasses implement factory method and decides which product to create.
    2. By decoupling the Product from Creator we’ve made Creator immune of the changes in any concrete Product.
    3. A factory method can be parameterized if needs to produce different concrete product depending on parameter or non-parameterized if it produces one type of object only.
    4. Use factory method pattern to decouple your client code from concrete classes if we don’t know ahead time all the concrete classes you’re going to need.
  1. **Dependency inversion principle**: Depend on abstractions do not depend on concrete classes.
     1. According to DI principle higher level component should not depend on lower level component they both depend on abstractions.



* + 1. DI guidelines:
       1. No variable reference to a concrete class
       2. No class should derive from a concrete class.
       3. No method should override an implemented method of its super class.
  1. **Abstract Factory Pattern**: The abstract factory pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes.



* + 1. Here client use abstract factory interface to create set of related products without knowing concrete product being produced. Client is decoupled from any details of concrete product.
    2. Often method of abstract factory is implemented as factory method pattern.
    3. Both factory method and abstract factory creates objects one through inheritance and other through composition.
    4. This factory pattern is used for creating a set of related products.
    5. Abstract factory interface needs to change if we required add creation of new product in our product set.

1. Java:
   1. Ex: **Map** synchronizedMap = **Collections**.synchronizedMap(**new** **HashMap**());