デザインパターンのデメリット

> 絶版に伴い、校正前の原稿テキストを公開したものです。基本的に原稿そのままをHTML形式に変換したものですので、誤字/脱字、説明不足の箇所もあるかも知れませんがご了承ください。初出:「PHPによるデザインパターン入門」(下岡秀幸/畑勝也/道端良 著, 秀和システム, ISBN4-7980-1516-4, 2006年11月23日発売)

ここまでデザインパターンを紹介してきて、「何だ。良いことずくめじゃないか」と感じられたのでないかと思います。しかし、デザインパターンは、万能薬でも銀の弾丸でもありません。やはり、デザインパターンにもデメリットはあります。

デザインパターンのデメリットについても、簡単に整理しておきます。

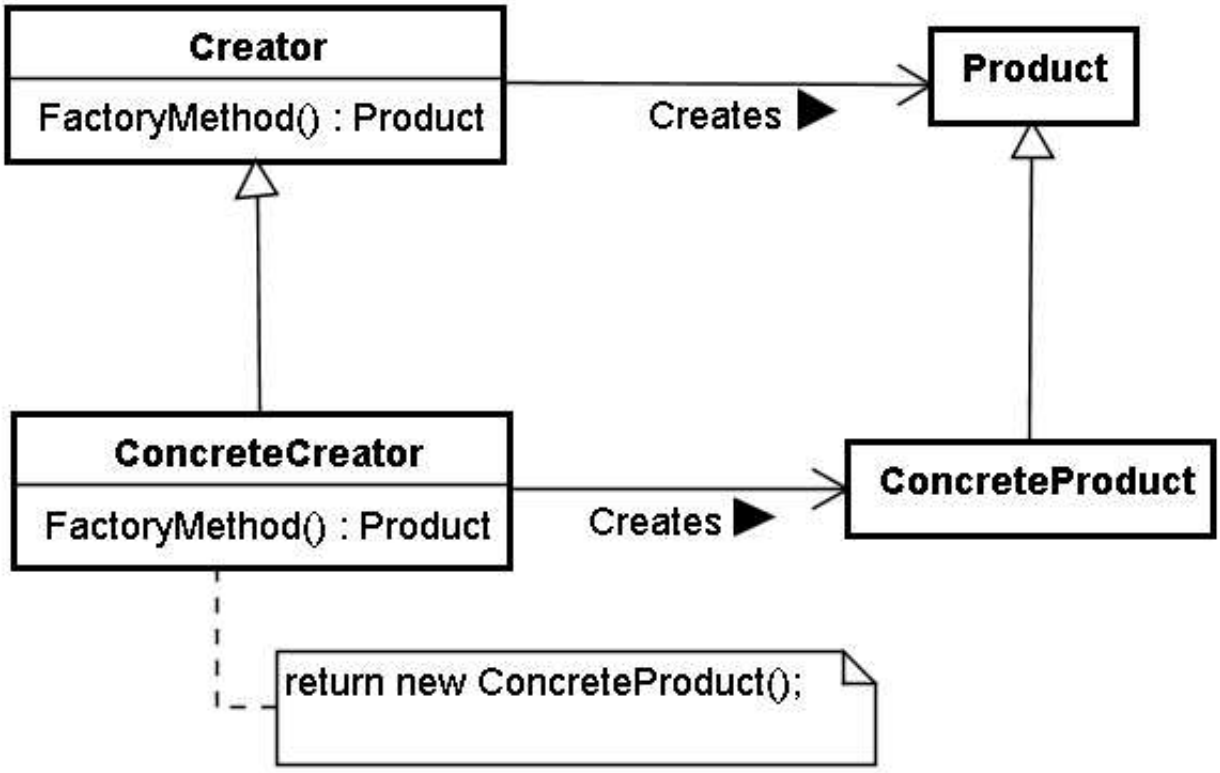
開発者どうしがデザインパターンを知らなければ、コミュニケーションが成立しない

コミュニケーションについても先のメリットに挙げましたが、当然コミュニケーションをとる相手もデザインパターンを知っている必要があります。そうでないと、一方通行な会話になってしまいます。

パターンのクラス図をそのまま適用しようとする

デザインパターンを学習している、もしくは一通り学習してから問題となる点です。

実践する際、UMLで描かれたクラス図をそのまま適用しようとしてしまいがちです。たとえば、Factory Methodパターンのクラス図があるとしましょう。



この場合ですと、「Creatorクラスはinterfaceにしてはいけない」とか「ProductクラスやConcreteCreatorクラスには、他のメソッドを追加してはいけない」といったものです。

デザインパターンは、法律や規則ではありません。問題解決のためのノウハウを提供しているものです。ひょっとすると、クラス図そのままで適用できる場面もあるかも知れません。しかし、ほとんどの場合はクラス図そのままではなく、何らかのアレンジを加えた状態で適用することがほとんどと思います。先のFactory Methodパターンの場合、Creatorクラスをinterfaceや抽象クラスで実装したりすることも問題ありません。当然、具象クラスで実装することもあるでしょう。また、Productクラスに多くのメソッドが用意されていることもあります。

このことが足かせになり、誤った適用を強いてしまう可能性があります。

無理にでもパターンを適用しようとする

これも学習中、学習後に問題となる点です。

人は新しく覚えた知識を使いたがります。このため、デザインパターンを無理矢理でも適用しようとする事があります。「金づちを持つとすべて釘に見える」と言ったコトバもありますね。

先の例でも挙げましたが、デザインパターンは、問題解決のためのノウハウを提供しているものです。あくまで「問題」を「解決」するのですから、無理矢理適用しても正しい使い方をしない限り、デザインパターンのメリットを教授することはできません。

使いこなすまでに時間がかかる

デザインパターン自体を学習するには、さほど時間はかからないと思います。しかし、実践してうまくパターンを適用できるまでには、それなりの時間がかかります。

デザインパターンに限らず、プログラミング言語や他の事項もそうですが、「知っていることとできることは違う」ということです。たとえ、23個のデザインパターンの構造をすべて覚えているとしても、実際に使えるかどうかは別問題です。これについては、何度もトライ＆エラーを繰り返し、パターン適用の「さじ加減」を体得する必要があると思います。

デザインパターンは諸刃の剣である

デザインパターンに限らない話ですが、デザインパターンも「諸刃の剣」の要素を持っています。デザインパターンのメリットにも挙げましたが、正しく使うことで再利用性や保守性が高いアプリケーションを作ることができます。逆に、デザインパターンの適用方法を誤ると、再利用性や保守性が低く、複雑なアプリケーションになる可能性があります。