# 前言

三年前，CSDN推出了一個產品——ink，旨在提供一個高品質寫作環境。那時，我想寫一系列關於設計模式的文章，於是就在ink裡開始寫作，陸陸續續寫了三篇文章，後來不知道什麼原因這個產品下架了，我的三篇文章也沒了，這事也就一直被我擱置下來。直到2017年，知識付費盛行，各類付費的社區、產品如雨後春筍般崛起，而技術類的付費閱讀產品更是大行其道（GitChat便是其中一種）。在GitChat的盛情邀請之下，我寫作設計模式這一系列文章的想法又重新被點燃。2017年年底，我開始在GitChat上寫“從生活中領悟設計模式（Python）”課程。2018年，我對這一課程進行了一次升級。

隨著這一課程被越來越多的讀者熟知，不少出版社編輯找到我，他們覺得這一課程的內容非常有特色，希望能把它重新整理，仔細打磨，出版成書，於是便有了本書。

## 本書的特色

設計模式作為物件導向程式的設計思想和方法論，本身是非常抽象和難以理解的，需要有一定的代碼量和程式設計經驗才能更深入地理解。如果能用一種有趣的方式來講解設計模式，將會使這些枯燥乏味的概念變得更易於理解！

本書每一章以一個輕鬆有趣的小故事開始，然後用代碼來模擬故事劇情，再從模擬代碼中逐步提煉出設計模式的模型和原理，最後配合一個具體的應用案例，告訴你每一種模式的使用方法和應用場景。以由淺入深的方式帶你瞭解每一種模式，思考每一種模式，總結每一種模式。

本書力求用更通俗的語言闡述難懂的概念，用更簡單的語法實現複雜的邏輯，用更短小的代碼寫出強悍的程式！希望能帶給讀者一種全新的閱讀體驗和思考方式。

## 內容概述

本書分為3篇：

* “基礎篇”講解了23種經典設計模式，其中19 種常用設計模式分別用單獨的章節講解，其餘設計模式作為一個合集放在一章中講解；
* “進階篇”講解了由基礎設計模式衍生出的各種程式設計機制，包括篩檢程式模式、物件集區技術、回檔機制和MVC模式，它們在各大程式設計語言中都非常重要而且常見；
* “經驗篇”結合工作經驗和專案積累，分享了對設計模式、設計原則、專案重構的理解和看法。

## 讀者對象

### 一線互聯網軟體發展者

如果你想提升物件導向的思維方式，提高自己的軟體設計能力，本書會對你非常有幫助。本書每一章會抽象和總結出對應設計模式的模型和原理，並結合具體的應用案例告訴你該模式的應用場景、特點和注意事項。

### IT職場新人

如果你是IT新人，想通過學習設計模式來提升自己的技術能力和代碼理解能力，本書將非常適合你。本書每一章以一個輕鬆有趣的小故事開始，由淺入深地講述一個模式，讓你輕鬆愉悅地學會每一種模式。

### 對設計模式和程式設計思想感興趣的人士

設計模式能讓你的代碼具有更高的再使用性、更好的靈活性和可拓展性，更易被人閱讀和理解，因此學習設計模式是每一個程式師程式設計生涯中必不可少的一個環節。

## 為什麼叫設計模式

### 什麼是設計模式

設計模式最初是由GoF於1995年提出的。GoF全稱是Gang of Four（四人幫），即Erich Gamma、Richard Helm、Ralph Johnson和John Vlissides。他們四人於1995年出版了一本書*Design Patterns: Elements of Reusable Object-Oriented Software*（翻譯成中文是《設計模式：可複用物件導向軟體的基礎》），第一次將設計模式提升到理論高度，並將之規範化，該書提出了23種經典的設計模式。

設計模式是一套被反復使用、多數人知曉、無數工程師實踐的代碼設計經驗的總結，它是物件導向思想的高度提煉和範本化。使用設計模式是為了讓代碼具有更高的再使用性、更好的靈活性和可拓展性，更易被人閱讀和理解。

### 設計模式與生活有什麼聯繫

我一直堅信：程式源于生活，又高於生活！程式的靈魂在於思維的方式，而思維的靈感來源於精彩的生活。互聯網是一個虛擬的世界，而程式本身就是對生活場景的虛擬和抽象，每一種模式我都能在生活中找到它的影子。比如，說到狀態模式，我能想到水有固、液、氣三種狀態，而人也有少、壯、老三個階段；提起仲介模式，我能立刻想到房產仲介；看到裝飾模式，我能聯想到人的穿衣搭配……

設計模式是物件導向的高度抽象和總結，而越抽象的東西越難以理解。本書的寫作目的就是降低設計模式的閱讀門檻，以生活中的小故事開始，用風趣的方式，由淺入深地講述每一種模式。讓你再次看到設計模式時，不覺得它只是一種模式，還是生活中的一個“小確幸”！程式不是冷冰冰的代碼，它還有生活的樂趣和特殊意義。

## 為什麼要學設計模式

設計模式是軟體發展人員在軟體發展過程中面臨的一般問題的解決方案。這些解決方案是眾多軟體發展人員經過相當長的一段時間的試驗總結出來的。所以不管你是新手還是老手，學習設計模式對你都有莫大的幫助。

學習設計模式的理由有很多，這裡只列出幾個最現實的：

（1）擺脫面試的窘境，不管你是前端工程師還是後端工程師，或是全端工程師，設計模式都是不少面試官必問的。

（2）讓你的程式設計能力有一個質的提升，不再寫一堆結構複雜、難以維護的爛代碼。

（3）使你對物件導向的思想有一個更高層次的理解。

## 如何進行學習

### 熟悉一門物件導向語言

首先，你至少要熟悉一門物件導向的電腦語言。如果沒有，請根據自己的興趣、愛好或希望從事的工作，先選擇一門物件導向語言（C++、Java、Go等都可以）進行學習和實戰，對抽象、繼承、多態、封裝有一定的基礎之後，再來閱讀本書。

### 瞭解Python的基本語法

對Python的基本語法有一個簡單瞭解。Python的語法非常簡單，只要你有一定的（其他）程式設計語言基礎，通過“第0章 啟程之前，請不要錯過我”的學習就能很快地理解Python的語法。

### 學會閱讀UML圖

UML（Unified Modeling Language）稱為統一模組化語言或標準建模語言，是物件導向軟體的標準化建模語言。UML類圖表示不同的實體（人、事物和資料）如何彼此相關，換句話說，它顯示了系統的靜態結構。想進一步瞭解類圖中的各種關係，可參考閱讀“第0章 啟程之前，請不要錯過我”的“0.2　UML精簡概述”部分。

### 閱讀本書

通過閱讀本書內容，可以輕鬆愉快地學習設計模式和程式設計思想。本書“基礎篇”“進階篇”“經驗篇”的內容是逐步進階和提升的，但每一篇內的不同章之間是沒有閱讀的先後順序的（第0章和有特殊說明的除外），每一章都單獨成文，可從任意一章開始閱讀。例如，對於基礎篇的23種設計模式，你可以從中任意挑選一章開始閱讀。

## 為什麼選擇Python

雖然設計模式與程式設計語言沒有關係，它是對物件導向思想的靈活應用和高度概括，你可以用任何一種語言來實現它，但總歸是需要用一種語言進行舉例的。本書的所有示例代碼均使用Python編寫（有特殊說明的除外），選擇Python主要基於以下兩個原因。

### 彌補市場空缺

設計模式於1995由GoF提出，被廣泛應用於熱門的物件導向語言。目前用Java、C++描述的設計模式的書籍和資料已經非常多了，但用Python來描述的真是太少了；我在當當網上搜索“Python設計模式”，只有零星的幾本書。而對於程式設計語言中排名前三的Python語言，這明顯是不夠的。Python已經越來越成熟，也越來越多地被使用，作為一個有技術追求的IT人，有必要瞭解一下基於Python代碼的設計模式。

### 大勢所趨，Python已然成風

C語言誕生於1972年，卻隨著UNIX的誕生才深深根植於各大作業系統；C++誕生於1983年，卻因微軟的視覺化桌面作業系統才得以廣泛傳播；Java誕生於1995年，卻因互聯網的迅速崛起才變得家喻戶曉；Python誕生於1991年，而下一場技術革命已然開始，AI時代已然到來，在AI領域中已經被廣泛使用的Python必將成為下一個時代的第一開發語言！

最熱門的AI開源框架PyTorch和TensorFlow都已經採用了Python作為介面和開發語言。除此之外，還有一堆AI相關的框架庫，也都紛紛採用了Python，如SKlearn、PyML等。一門如此有前途的語言，我們必然是要去學習和使用的。

## 勘誤和支持

由於本人水準和經驗有限，書中難免會有一些錯誤或理解不準確的地方，懇請廣大讀者批評指正。

如果你在閱讀過程中發現錯誤，或有更好的建議，歡迎發郵件給我（E-mail：luoweifu@ 126.com，永久有效）。

最新的勘誤內容可通過以下方式查看：關注公眾號“SunLogging”，在功能表列中選擇“我的書箱”→“最新勘誤”。

## 致謝

從在GitChat上寫課程，到與出版社合作，寫完本書的書稿，大概經歷了一年半的時間，經過無數次與編輯的反復校對。寫作是一件非常考驗人耐心和細心的事，為了讓讀者更易理解，有些章節我進行了反復的推敲和修改。比如，為了講清楚單例模式的每一種實現方式的原理，硬是增加了兩個附錄，閱讀了十幾篇文章，並做了驗證性的實驗，整整花了三周時間才寫完。

感謝每一位在本書寫作過程中給予幫助的人，是你們的鼓勵和支持，才讓本書能順利完成。在此，要特別感謝電子出版社的首席策劃編輯董英，在寫書過中給予的諸多建議；也感謝GitChat的編輯馬翠翠，在寫線上課程“從生活中領悟設計模式（Python）”時給予的很多幫助；還要感謝Sophia“小朋友”，在封面設計過程中提出的非常細緻的改進意見！最後，我也要感謝我的朋友和同事對我寫書的鼓勵和支持。

還要感謝Sophia“小朋友”，在封面設計過程中提出的非常細緻的改進意見！