06讲理论三：面向对象相比面向过程有哪些优势面向过程真的过时了吗

在上两节课中，我们讲了面向对象这种现在非常流行的编程范式，或者说编程风格。实际上，除了面向对象之外，被大家熟知的编程范式还有另外两种，面向过程编程和函数式编程。面向过程这种编程范式随着面向对象的出现，已经慢慢退出了舞台，而函数式编程目前还没有被广泛接受。

在专栏中，我不会对函数式编程做讲解，但我会花两节课的时间，讲一下面向过程这种编程范式。你可能会问，既然面向对象已经成为主流的编程范式，而面向过程已经不那么推荐使用，那为什么又要浪费时间讲它呢？

那是因为在过往的工作中，我发现很多人搞不清楚面向对象和面向过程的区别，总以为使用面向对象编程语言来做开发，就是在进行面向对象编程了。而实际上，他们只是在用面向对象编程语言，编写面向过程风格的代码而已，并没有发挥面向对象编程的优势。这就相当于手握一把屠龙刀，却只是把它当作一把普通的刀剑来用，相当可惜。

所以，我打算详细对比一下面向过程和面向对象这两种编程范式，带你一块搞清楚下面这几个问题（前三个问题我今天讲解，后三个问题我放到下一节课中讲解）：

1. 什么是面向过程编程与面向过程编程语言？
2. 面向对象编程相比面向过程编程有哪些优势？
3. 为什么说面向对象编程语言比面向过程编程语言更高级？
4. 有哪些看似是面向对象实际是面向过程风格的代码？
5. 在面向对象编程中，为什么容易写出面向过程风格的代码？
6. 面向过程编程和面向过程编程语言就真的无用武之地了吗？

话不多说，带着这几个问题，我们就正式开始今天的学习吧！

什么是面向过程编程与面向过程编程语言？

如果你是一名比较资深的程序员，最开始学习编程的时候，接触的是Basic、Pascal、C等面向过程的编程语言，那你对这两个概念肯定不陌生。但如果你是新生代的程序员，一开始学编程的时候，接触的就是面向对象编程语言，那你对这两个概念可

能会比较不熟悉。所以，在对比面向对象与面向过程优劣之前，我们先把面向过程编程和面向过程编程语言这两个概念搞清楚。

实际上，我们可以对比着面向对象编程和面向对象编程语言这两个概念，来理解面向过程编程和面向过程编程语言。还记得我们之前是如何定义面向对象编程和面向对象编程语言的吗？让我们一块再来回顾一下。

- 面向对象编程是一种编程范式或编程风格。它以类或对象作为组织代码的基本单元，并将封装、抽象、继承、多态四个特性，作为代码设计和实现的基石 。
- 面向对象编程语言是支持类或对象的语法机制，并有现成的语法机制，能方便地实现面向对象编程四大特性（封装、抽象、继承、多态）的编程语言。

类比面向对象编程与面向对象编程语言的定义，对于面向过程编程和面向过程编程语言这两个概念，我给出下面这样的定义。

- 面向过程编程也是一种编程范式或编程风格。它以过程（可以理解为方法、函数、操作）作为组织代码的基本单元，以数据（可以理解为成员变量、属性）与方法相分离为最主要的特点。面向过程风格是一种流程化的编程风格，通过拼接一组顺序执行的方法来操作数据完成一项功能。
- 面向过程编程语言首先是一种编程语言。它最大的特点是不支持类和对象两个语法概念，不支持丰富的面向对象编程特性（比如继承、多态、封装），仅支持面向过程编程。

不过，这里我必须声明一下，就像我们在之前讲到的，面向对象编程和面向对象编程语言并没有官方的定义一样，这里我给出的面向过程编程和面向过程编程语言的定义，也并不是严格的官方定义。之所以要给出这样的定义，只是为了跟面向对象编程及面向对象编程语言做个对比，以方便你理解它们的区别。

定义不是很严格，也比较抽象，所以，我再用一个例子进一步解释一下。假设我们有一个记录了用户信息的文本文件users.txt，每行文本的格式是name&age&gender（比如，小王&28&男）。我们希望写一个程序，从users.txt文件中逐行读取用户信息，然后格式化成name\tage\tgender（其中，\t是分隔符）这种文本格式，并且按照age从小到大排序之后，重新写入到另一个文本文件formatted_users.txt中。针对这样一个小程序的开发，我们一块来看看，用面向过程和面向对象两种编程风格，编写出来的代码有什么不同。

首先，我们先来看，用面向过程这种编程风格写出来的代码是什么样子的。注意，下面的代码是用C语言这种面向过程的编程语言来编写的。

```
struct User {
  char name[64];
  int age;
  char gender[16];
};

struct User parse_to_user(char* text) {
  // 将text(“小王&28&男”)解析成结构体struct User
}

char* format_to_text(struct User user) {
  // 将结构体struct User格式化成文本 ("小王\t28\t男")
}

void sort_users_by_age(struct User users[]) {
  // 按照年龄从小到大排序users
}

void format_user_file(char* origin_file_path, char* new_file_path) {
  // open files...
  struct User users[1024]; // 假设最大1024个用户
  int count = 0;
  while(1) { // read until the file is empty
    struct User user = parse_to_user(line);
    users[count++] = user;
  }

  sort_users_by_age(users);

  for (int i = 0; i < count; ++i) {
    char* formatted_user_text = format_to_text(users[i]);
    // write to new file...
  }
  // close files...
}

int main(char** args, int argv) {
  format_user_file("/home/zheng/user.txt", "/home/zheng/formatted_users.txt");
}
```

然后，我们再来看，用面向对象这种编程风格写出来的代码是什么样子的。注意，下面的代码是用Java这种面向对象的编程语言来编写的。

```
 public class User {
  private String name;
  private int age;
  private String gender;

  public User(String name, int age, String gender) {
    this.name = name;
    this.age = age;
    this.gender = gender;
  }

  public static User praseFrom(String userInfoText) {
    // 将text(“小王&28&男”)解析成类User
  }

  public String formatToText() {
    // 将类User格式化成文本（"小王\t28\t男"）
  }
}

public class UserFileFormatter {
  public void format(String userFile, String formattedUserFile) {
    // Open files...
    List users = new ArrayList<>();
    while (1) { // read until file is empty
      // read from file into userText...
      User user = User.parseFrom(userText);
      users.add(user);
    }
    // sort users by age...
    for (int i = 0; i < users.size(); ++i) {
      String formattedUserText = user.formatToText();
      // write to new file...
    }
    // close files...
  }
}

public class MainApplication {
  public static void main(Sring[] args) {
    UserFileFormatter userFileFormatter = new UserFileFormatter();
    userFileFormatter.format("/home/zheng/users.txt", "/home/zheng/formatted_users.txt");
  }
}
```

从上面的代码中，我们可以看出，面向过程和面向对象最基本的区别就是，代码的组织方式不同。面向过程风格的代码被组织成了一组方法集合及其数据结构（struct User），方法和数据结构的定义是分开的。面向对象风格的代码被组织成一组类，方法和数据结构被绑定一起，定义在类中。

看完这个例子之后，你可能会说，面向对象编程和面向过程编程，两种风格的区别就这么一点吗？当然不是，对于这两种编程风格的更多区别，我们继续往下看。

面向对象编程相比面向过程编程有哪些优势?

刚刚我们介绍了面向过程编程及面向过程编程语言的定义，并跟面向对象编程及面向对象编程语言做了一个简单对比。接下来，我们再来看一下，为什么面向对象编程晚于面向过程编程出现，却能取而代之，成为现在主流的编程范式？面向对象编程跟面向过程编程比起来，到底有哪些优势?

1.OOP更加能够应对大规模复杂程序的开发

看了刚刚举的那个格式化文本文件的例子，你可能会有这样的疑问，两种编程风格实现的代码貌似差不多啊，顶多就是代码的组织方式有点区别，没有感觉到面向对象编程有什么明显的优势呀！你的感觉没错。之所以有这种感觉，主要原因是这个例子程序比较简单、不够复杂。

对于简单程序的开发来说，不管是用面向过程编程风格，还是用面向对象编程风格，差别确实不会很大，甚至有的时候，面向过程的编程风格反倒更有优势。因为需求足够简单，整个程序的处理流程只有一条主线，很容易被划分成顺序执行的几个步骤，然后逐句翻译成代码，这就非常适合采用面向过程这种面条式的编程风格来实现。

但对于大规模复杂程序的开发来说，整个程序的处理流程错综复杂，并非只有一条主线。如果把整个程序的处理流程画出来的话，会是一个网状结构。如果我们再用面向过程编程这种流程化、线性的思维方式，去翻译这个网状结构，去思考如何把程序拆解为一组顺序执行的方法，就会比较吃力。这个时候，面向对象的编程风格的优势就比较明显了。

面向对象编程是以类为思考对象。在进行面向对象编程的时候，我们并不是一上来就去思考，如何将复杂的流程拆解为一个一个方法，而是采用曲线救国的策略，先去思考如何给业务建模，如何将需求翻译为类，如何给类之间建立交互关系，而完成这些工作完全不需要考虑错综复杂的处理流程。当我们有了类的设计之后，然后再像搭积木一样，按照处理流程，将类组装起来形成整个程序。这种开发模式、思考问题的方式，能让我们在应对复杂程序开发的时候，思路更加清晰。

除此之外，面向对象编程还提供了一种更加清晰的、更加模块化的代码组织方式。比如，我们开发一个电商交易系统，业务逻辑复杂，代码量很大，可能要定义数百个函数、数百个数据结构，那如何分门别类地组织这些函数和数据结构，才能不至于看起来比较凌乱呢？类就是一种非常好的组织这些函数和数据结构的方式，是一种将代码模块化的有效手段。

你可能会说，像C语言这种面向过程的编程语言，我们也可以按照功能的不同，把函数和数据结构放到不同的文件里，以达到给函数和数据结构分类的目的，照样可以实现代码的模块化。你说得没错。只不过面向对象编程本身提供了类的概念，强制你做这件事情，而面向过程编程并不强求。这也算是面向对象编程相对于面向过程编程的一个微创新吧。

实际上，利用面向过程的编程语言照样可以写出面向对象风格的代码，只不过可能会比用面向对象编程语言来写面向对象风格的代码，付出的代价要高一些。而且，面向过程编程和面向对象编程并非完全对立的。很多软件开发中，尽管利用的是面向过程的编程语言，也都有借鉴面向对象编程的一些优点。

2.OOP风格的代码更易复用、易扩展、易维护

在刚刚的那个例子中，因为代码比较简单，所以只用到到了类、对象这两个最基本的面向对象概念，并没有用到更加高级的四大特性，封装、抽象、继承、多态。因此，面向对象编程的优势其实并没有发挥出来。

面向过程编程是一种非常简单的编程风格，并没有像面向对象编程那样提供丰富的特性。而面向对象编程提供的封装、抽象、继承、多态这些特性，能极大地满足复杂的编程需求，能方便我们写出更易复用、易扩展、易维护的代码。为什么这么说呢？还记得我们在上一节课中讲到的封装、抽象、继承、多态存在的意义吗？我们再来简单回顾一下。

首先，我们先来看下封装特性。封装特性是面向对象编程相比于面向过程编程的一个最基本的区别，因为它基于的是面向对象编程中最基本的类的概念。面向对象编程通过类这种组织代码的方式，将数据和方法绑定在一起，通过访问权限控制，只允许外部调用者通过类暴露的有限方法访问数据，而不会像面向过程编程那样，数据可以被任意方法随意修改。因此，面向对象编程提供的封装特性更有利于提高代码的易维护性。

其次，我们再来看下抽象特性。我们知道，函数本身就是一种抽象，它隐藏了具体的实现。我们在使用函数的时候，只需要了解函数具有什么功能，而不需要了解它是怎么实现的。从这一点上，不管面向过程编程还是是面向对象编程，都支持抽象特性。不过，面向对象编程还提供了其他抽象特性的实现方式。这些实现方式是面向过程编程所不具备的，比如基于接口实现的抽象。基于接口的抽象，可以让我们在不改变原有实现的情况下，轻松替换新的实现逻辑，提高了代码的可扩展性。

再次，我们来看下继承特性。继承特性是面向对象编程相比于面向过程编程所特有的两个特性之一（另一个是多态）。如果两个类有一些相同的属性和方法，我们就可以将这些相同的代码，抽取到父类中，让两个子类继承父类。这样两个子类也就可以重用父类中的代码，避免了代码重复写多遍，提高了代码的复用性。

最后，我们来看下多态特性。基于这个特性，我们在需要修改一个功能实现的时候，可以通过实现一个新的子类的方式，在子类中重写原来的功能逻辑，用子类替换父类。在实际的代码运行过程中，调用子类新的功能逻辑，而不是在原有代码上做修改。这就遵从了“对修改关闭、对扩展开放”的设计原则，提高代码的扩展性。除此之外，利用多态特性，不同的类对象可以传递给相同的方法，执行不同的代码逻辑，提高了代码的复用性。

所以说，基于这四大特性，利用面向对象编程，我们可以更轻松地写出易复用、易扩展、易维护的代码。当然，我们不能说，利用面向过程风格就不可以写出易复用、易扩展、易维护的代码，但没有四大特性的帮助，付出的代价可能就要高一些。

3.OOP语言更加人性化、更加高级、更加智能

人类最开始跟机器打交道是通过0、1这样的二进制指令，然后是汇编语言，再之后才出现了高级编程语言。在高级编程语言中，面向过程编程语言又早于面向对象编程语言出现。之所以先出现面向过程编程语言，那是因为跟机器交互的方式，从二进制指令、汇编语言到面向过程编程语言，是一个非常自然的过渡，都是一种流程化的、面条式的编程风格，用一组指令顺序操作数据，来完成一项任务。

从指令到汇编再到面向过程编程语言，跟机器打交道的方式在不停地演进，从中我们很容易发现这样一条规律，那就是编程语言越来越人性化，让人跟机器打交道越来越容易。笼统点讲，就是编程语言越来越高级。实际上，在面向过程编程语言之后，面向对象编程语言的出现，也顺应了这样的发展规律，也就是说，面向对象编程语言比面向过程编程语言更加高级！

跟二进制指令、汇编语言、面向过程编程语言相比，面向对象编程语言的编程套路、思考问题的方式，是完全不一样的。前三者是一种计算机思维方式，而面向对象是一种人类的思维方式。我们在用前面三种语言编程的时候，我们是在思考，如何设计一组指令，告诉机器去执行这组指令，操作某些数据，帮我们完成某个任务。而在进行面向对象编程时候，我们是在思考，如何给业务建模，如何将真实的世界映射为类或者对象，这让我们更加能聚焦到业务本身，而不是思考如何跟机器打交道。可以这么说，越高级的编程语言离机器越“远”，离我们人类越“近”，越“智能”。

这里多聊几句，顺着刚刚这个编程语言的发展规律来想，如果一种新的突破性的编程语言出现，那它肯定是更加“智能”的。大胆想象一下，使用这种编程语言，我们可以无需对计算机知识有任何了解，无需像现在这样一行一行地敲很多代码，只需要把需求文档写清楚，就能自动生成我们想要的软件了。

重点回顾

今天的内容就讲完了，我们来一起总结回顾一下，你需要重点掌握的几个知识点。

1.什么是面向过程编程？什么是面向过程编程语言？

实际上，面向过程编程和面向过程编程语言并没有严格的官方定义。理解这两个概念最好的方式是跟面向对象编程和面向对象编程语言进行对比。相较于面向对象编程以类为组织代码的基本单元，面向过程编程则是以过程（或方法）作为组织代码的基本单元。它最主要的特点就是数据和方法相分离。相较于面向对象编程语言，面向过程编程语言最大的特点就是不支持丰富的面向对象编程特性，比如继承、多态、封装。

2.面向对象编程相比面向过程编程有哪些优势？

面向对象编程相比起面向过程编程的优势主要有三个。

- 对于大规模复杂程序的开发，程序的处理流程并非单一的一条主线，而是错综复杂的网状结构。面向对象编程比起面向过程编程，更能应对这种复杂类型的程序开发。
- 面向对象编程相比面向过程编程，具有更加丰富的特性（封装、抽象、继承、多态）。利用这些特性编写出来的代码，更加易扩展、易复用、易维护。
- 从编程语言跟机器打交道的方式的演进规律中，我们可以总结出：面向对象编程语言比起面向过程编程语言，更加人性化、更加高级、更加智能。

课堂讨论

在文章中我讲到，面向对象编程比面向过程编程，更加容易应对大规模复杂程序的开发。但像Unix、Linux这些复杂的系统，也都是基于C语言这种面向过程的编程语言开发的，你怎么看待这个现象？这跟我之前的讲解相矛盾吗？

欢迎在留言区写下你的答案，和同学一起交流和分享。如果有收获，也欢迎你把这篇文章分享给你的朋友。

精选留言

相逢是缘

使用任何一个编程语言编写的程序，最终执行上都要落实到CPU一条一条指令的执行（无论通过虚拟机解释执行，还是直接编译为机器码），CPU看不到是使用何种语言编写的程序。对于所有编程语言最终目的是两种：提高硬件的运行效率和提高程序员的开发效率。然而这两种很难兼得。

C语言在效率方面几乎做到了极致，它更适合挖掘硬件的价值，如：C语言用数组char a[8]，经过编译以后变成了（基地址＋偏移量）的方式。对于CPU来说，没有运算比加法更快，它的执行效率的算法复杂度是O(1)的。从执行效率这个方面看，开发操作系统和贴近硬件的底层程序，C语言是极好的选择。

C语言带来的问题是内存越界、野指针、内存泄露等。它只关心程序飞的高不高，不关心程序猿飞的累不累。为了解脱程序员，提高开发效率，设计了OOP等更“智能”的编程语言，但是开发容易毕竟来源于对底层的一层一层又一层的包装。完成一个特定操作有了更多的中间环节, 占用了更大的内存空间, 占用了更多的CPU运算。从这个角度看，OOP这种高级语言的流行是因为硬件越来越便宜了。我们可以想象如果大众消费级的主控芯片仍然是单核600MHz为主流，运行Android系统点击一个界面需要2秒才能响应，那我们现在用的大部分手机程序绝对不是使用JAVA开发的，Android操作系统也不可能建立起这么大的生态。

2019-11-16 15:47

作者回复

2019-11-19 10:15

猫切切切切切

操作系统是业务无关的，它更接近于底层计算机，因此更适合用面向过程的语言编写。而接近业务的也就是接近人的软件，则更适合用面向对象的语言编写。

2019-11-15 01:09

辣么大

我们以历史的时间线看看这两种语言的演进过程。

1969年贝尔实验室提出Unix操作系统

1972年贝尔实验室的Dennis Ritchie开发C语言。
1973年他用C语言重写了Unix。
1991年Linus Torvalds提出Linux。

另外一条线面向对象语言的发展：
1972年第一个面向对象的编程语言是Simula发布。
1996年，Java1.0发布，流行的主要原因是jvm，Write Once, Run Anywhere（编写一次，到处安装JRE）

从时间上看，面向对象概念的提出晚于面向过程。C语言因商业应用成熟要比面相对象的编程语言早。

C语言的流行主要是因为Unix和Linux操作系统的实现基于C语言。类Unix系统可以运行在服务器，嵌入式设备，移动设备上。

一个东西的好坏要综合考虑：面向过程和面向对象各有各的优缺点。一门编程语言的提出是为了解决某些特定的问题。面向对象和面向过程在应用上各有各的位置。

2019-11-15 06:27

Paul Shan

思考题
大学学习操作系统的时候，大部分内容已经忘了，还记得老师说过，虽然操作系统是用C语言写的，但是面向对象的思想早已深入到操作系统的源代码中。

2019-11-15 05:09

LYy

操作系统虽然是用面向过程的C语言实现的 但是其设计逻辑是面向对象的。
C语言没有类和对象的概念，但是用结构体（struct）同样实现了信息的封装，内核源码中也不乏继承和多态思想的体现。
面向对象思想，不局限于具体语言。

2019-11-15 14:36

lijun

深夜追设计模式！

2019-11-15 00:05

李小四

设计模式_06
操作系统的源码一直没读过，但我认为如此复杂的系统设计，(站在现在的时间点)用面向对象风格(或实现相同目的其他方式)来编写代码是更合适的，而且从Linux的模块化分来看，推测有类似的实践。

老话说, 机器能读懂所有代码, 但人不一定. 对于机器来说, 每一次业务调用流程都是序列化的, 机器并不在乎面向对象, 但人在乎. 正如本文所讲, 编程语言离机器越来越远, 离人越来越近. 为了迁就人, 我们使用了执行效率更低的语言, 有了更多的中间环节, 占用了更大的内存空间, 换来的是这个行业的蓬勃发展以及让人类的便捷生活和能力延伸.

2019-11-15 09:37

墨雨

Java代码的注释是不是没改呀……为啥还是结构体呢……。感觉这几篇都比较偏理论，有点拖踏

2019-11-15 08:31

作者回复

注释忘了改 我改下 多谢提醒

2019-11-15 08:50

未未的未来

疑问：
老师举的文件那个例子，使用面向对象编程那个，不是封装了函数，用函数对操作过程进行了抽象了吗，为什么老师说没有用到封装、抽象这些特性?
思考题：
理解，C语言虽然是面相过程语言，但是面向过程语言也可以写面向对象的，另外，C语言更贴近底层一些，写操作系统的话还是有性能上的优势。

2019-11-15 08:28

作者回复

你指出的这点很好。关于封装，有两种理解，一种是狭义的面向对象特性：封装是一种信息隐藏，需要把数据和方法放到一起，而c语言实现的代码，数据和方法是分离的。封装的另一种广义的理解，可以包含你指的封装函数。抽象实际上我们前面章节中也讲到过，比较没有特异性，有的时候不看做面向对象的特性。

2019-11-15 10:45

寒江独钓者

读过linux内核源码和python解释器源码的应该都明白，所谓面向过程的C语言照样可以实现面向对象的思想，有很多设计都是非常优雅的，付出的代价并不高，我并不认为面向对象编程语言做相同的事情付出的代价会更低。编程思想、设计模式跟语言是没有关系的，编程思想和设计模式是指导我们编程的，而编程语言只是一种实现工具罢了。

2019-11-17 15:10

养成好习惯

go语言大力推举函数式编程，这是趋势吗老师

2019-11-18 16:48

作者回复

函数式编程让写代码更加简单些，封装了很多设计模式、并发处理，可能是个趋势。

2019-11-19 10:03

月坛小雨

这套课程的配图，没有《算法》那套用心呀

2019-11-15 09:15

作者回复

因为本身这门课的图就没有像算法那样多。这门课代码比较多。

2019-11-15 10:38

摸爬滚打三十年

老师前几节讲的都是基本概念，每读一遍就感觉加深了一层印象和理解。学习本节最大的感受，面向对象和面向过程之间相互对比更容易理解，他们最大的区别1.代码的组织单元:面向对象是类和对象，面向过程是函数和数据。2.是否支持四大特性以及对四大特性的支持程度，面向过程不支持继承和多态，在封装和抽象上，面向对象要更加深刻一些。面向对象对类抽象，面向过程抽象成方法，面条式的执行过程，流水线的操作方式。通过访问控制，面向对象对外暴露有限的成员属性和方法。总体来说，面向对象编程更适合人的思维习惯，能够客观反映代码和真实世界的关系。我的一点体会，不知道对不对，过一段时间再来回顾一次，看到时候的理解会不会更深。

2019-11-15 08:56

CHS

不矛盾，面向对象是一种思想，语言是一种工具，任何语言都可以使用面向对象的思想来写代码。系统使用C语言，是因为C语言更加底层，执行效率高。

2019-11-26 13:35

KaitoShy

我理解C++的出现主要是为了C语言中的泛型编程问题。而Java 的出现是为了抽象对计算机的依赖，更专注于业务。对于计算机底层 C/C++ 可能更适合，而 C 的对于程序员来说自由度是最大的。再看Linux的发展，1991年，林纳斯基于Unix的编写的，可能他更善于这个语言，他编写 git 的时候也是用的 C。Unix选择C也是由于自己擅长的原因

2019-11-15 09:44

null

OOP 更能应付复杂流程的程序开发，自己是深有体会。

去年做了一个功能，第一版着急上线，就照着流程图翻译，最后就是一组顺序执行的方法集。if 分支特别多。

上线后有空，自己重构了一版。这次是翻译了泳道图：data_provider、rule_filter_chain、data_consumer 和 data_writer。data_provider 提供 3 种类型的数据。每种类型数据有 2 种规则。匹配规则后，会有一些数据处理，如与订单绑定、返佣记录等。最后是将这些数据持久化到数据库。处理过程还使用了多态特性，因此在调用 rule_filter_chain、data_consumer、data_writer 处的代码都特别精简，都是从 Map 获取处理对象，然后直接调用方法。

现在回看，重构时自己关注最多也是思考最多的是每个阶段的类对象，它的职责是什么。如入参是什么，业务逻辑处理，处理后的出参是什么。实现完这些类后，就真的是像搭积木一样将这些类串起来就可以了。就跟文章说的一样：业务建模、翻译需求为类、类之间交互。
重构后的代码逻辑清晰了，也更简洁了。整个业务也更容易理解了。但是有一点，重构之后，类文件也增加了挺多。但是相比

一个类几百上千行代码，我更喜欢简短的类。适当的增加一些类文件，也是能接受的。

2020-01-09 10:05

Better me

讨论题发表一下自己的意见

用面向对象语言结合面向过程思想写程序，这就相当于手握一把屠龙刀，却只是把它当作一把普通的刀剑来用。相反我们也可以用面向过程语言结合面向对象思想来写程序，那也能达到很好的易复用、易扩展、易维护的效果，相对来说代价会比较高，毕竟语言本身没有提供这种语法机制，需要在我们编程中体现处理实属不易。

2019-11-19 20:38

苗

面向过程最主要的特点就是数据和方法相分离。我哭了，虽然一直用的是MVC框架，但是没有业务建模；基本上都是数据导向，妥妥的面向过程开发。

2019-11-19 18:58

筱乐乐哦

我的理解不矛盾，理由如下

1、操作系统更多的是和硬件打交道，需要考虑到语言本身翻译成机器语言的成本和执行效率，尤其总要。

2、如linux内核、总线、文件系统网络等的设计，也是具有面向对象过程的思想，很好的支持了常见的文件系统的挂在、内核的升级和对硬件的热插拔、网络等的处理，常见的服务器多数都是linux，为啥不用window，我的理解和设计实现有很大的关系，尤其是网络、安全、权限、标准等等

3、linux操作系统，本身就是一个大的抽象，属于一个硬件和高级软件连接的桥梁

也希望争哥发表下自己的看法，指点指点

2019-11-16 00:56

hudson

formatToText换成toString是不是更符合习惯?

2019-11-15 23:37

作者回复

会不会有歧义呢?

2019-11-19 10:16