程序语言的常见设计错误(2) - 试图容纳世界

之前的一篇文章里，我谈到了程序语言设计的一个常见错误倾向：片面追求短小，它导致了一系列的历史性的设计错误。今天我来谈一下另外一种错误的倾向，这种倾向也导致了很多错误，并且继续在导致错误的产生。

今天我要说的错误倾向叫做“试图容纳世界”。这个错误导致了 Python，Ruby 和 JavaScript 等“动态语言”里面的一系列问题。我给 Python 写过一个静态分析器，所以我基本上实现了整个 Python 的语义，可以说是对 Python 了解的相当清楚了。在设计这个静态分析的时候，我发现 Python 的设计让静态分析异常的困难，Python 的程序出了问题很难找到错误的所在，Python 程序的执行速度比大部分程序语言都要慢，这其实是源自 Python 本身的设计问题。这些设计问题，其实大部分出自同一个设计倾向，也就是“试图容纳世界”。

在 Python 里面，每个“对象”都有一个“字典”（dictionary）。这个 dict 里面含有这个对象的 field 到它们的值之间的映射关系，其实就是一个哈希表。一般的语言都要求你事先定义这些名字，并且指定它们的类型。而 Python 不是这样，在 Python 里面你可以定义一个人，这个人的 field 包括“名字”，“头”，“手”，“脚”，……

但是 Python 觉得，程序应该可以随时创建或者删除这些 field。所以，你可以给一个特定的人增加一个 field，比如叫做“第三只手”。你也可以删除它的某个 field，比如“头”。Python 认为这更加符合这个世界的工作原理，有些人就是可以没有头，有些人又多长了一只手。

好吧，这真是太方便了。然后你就遇到这样的问题，你要给这世界上的每个人戴一顶帽子。当你写这段代码的时候，你意识中每个人都有头，所以你写了一个函数叫做 `putOnHat`，它的输入参数是任意一个人，然后它会给他（她）的头上戴上帽子。然后你想把这个函数 `map` 到一个国家的所有人的集合。

然而你没有想到的是，由于 Python 提供的这种“描述世界的能力”，其它写代码的人制造出各种你想都没想到的怪人。比如，无头人，或者有三只手，六只眼的人，…… 然后你就发现，无论你的 `putOnHat` 怎么写，总是会出意外。你惊讶的发现居然有人没有头！最悲惨的事情是，当你费了几个月时间和相当多的能源，给好几亿人戴上了帽子之后，才忽然遇到一个无头人，所以程序当掉了。然而即使你知道程序有 bug，你却很难找出这些无头人是从哪里来的，因为他们来到这个国家的道路相当曲折，绕了好多道弯。为了重现这个 bug，你得等好几个月，它还不一定会出现…… 这就是所谓 Higgs-Bugson 吧。

怎么办呢？所以你想出了一个办法，把“正常人”单独放在一个列表里，其它的怪人另外处理。于是你就希望有一个办法，让别人无法把那些怪人放进这个列表里。你想要的其实就是 Java 里的“类型”，像这样：

```
List<有一个头和两只手的正常人> normalPeople;
```

很可惜，Python 不提供给你这种机制，因为这种机制按照 Python 的“哲学”，不足以容纳这个世界的博大精深的万千变化。让程序员手工给参数和变量写上类型，被认为是“过多的劳动”。

这个问题也存在于 JavaScript 和 Ruby。

语言的设计者们都应该明白，程序语言不是用来“构造世界”的，而只是对它进行简单的模拟。试图容纳世界的倾向，没带来很多好处，没有节省程序员很多精力，却使得代码完全没有规则可言。这就像生活在一个没有规则，没有制度，没有法律的世界，经常发生无法预料的事情，到处跑着没有头，三只手，六只眼的怪人。这是无穷无尽的烦恼和时间精力的浪费。