前言

我最初将Java视为“只是另一种编程语言”，在很多方面都是如此。

但是随着时间的流逝，我对它进行了更深入的研究，我开始发现这种语言与我到目前为止所见的其他语言不同。编程与管理复杂性有关：您要解决的问题的复杂性，取决于解决它的机器的复杂性。因为复杂性，我们的大多数编程项目都会失败。但是，据我所知在所有编程中，几乎没有一种语言全力以赴，并决定它们的主要语言设计目标将是克服开发和维护程序的复杂性。当然，许多语言设计决策都是考虑到复杂性的，但是在某些情况下要指出的是，总有其他问题被认为是必不可少的。不可避免地，那些其他问题是导致程序员最终“撞墙”的原因。这种语言。例如，C ++必须与C向后兼容（C程序员的迁移），以及高效。这些都是非常有用的目标，占了C ++成功的大部分，但是它们也暴露出额外的复杂性，从而阻止了一些项目的完成（当然，您可以责怪程序员和管理人员，但是如果一种语言可以帮助您发现错误，那为什么不呢？）再举一个例子Visual BASIC（VB）与BASIC捆绑在一起，而BASIC并不是真正设计成可扩展的语言，因此VB上的所有扩展都产生了一些真正无法维护的语法。 Perl向后兼容和代替awk，sed，grep和其他Unix工具，因此经常被指责产生“只写代码”（写完之后，过一会儿就看不懂了）。另一方面，C ++，VB，Perl和其他语言（如Smalltalk）他们的一些设计工作集中在复杂性问题上，结果是在解决某些类型的问题方面非常成功。

当我了解Java时，最让我印象深刻的是在Sun庞杂的设计目标中，似乎有一个目标是为程序员降低复杂度的。好像在说：“我们关心如何减少生产时间和复杂度的同时兼顾代码健壮性。”在早期，这个目标导致代码运行得不太快（尽管随着时间的推移，这种情况有所改善），但确实确实减少了开发工作时间-创建等效的C ++程序所需时间的一半或更少。这个结果仅此一项就可以节省大量的时间和金钱，但是Java并不仅限于此。继续包装许多已经变得重要的复杂任务，例如多线程和网络编程，其语言功能或库有时可以轻松地完成这些任务。最后，它解决了一些非常大的复杂性问题：跨平台程序，动态代码更改，甚至安全性，每种都可以满足您的复杂性需求，比如在任何地方“设置障碍”或“暂停程序进行观察”。因此，尽管存在性能问题，我们已经看到，Java的前景非常广阔：它可以使我们的生产力大大提高程序员。

各种方式创建程序，团队合作，建立用户界面与用户沟通，在不同类型的计算机上运行程序，并且轻松编写可通过Internet进行通信的程序-Java可增加人与人之间的通信。

我认为通信革命的结果可能不会从大量数据交互中展现。 但我们将看到真正的革命，因为我们将彼此之间更容易地进行交流：像星系中的一个个星群。