HZ BOOKS
华章科技

计 算 机 科 学 丛 书

PEARSON

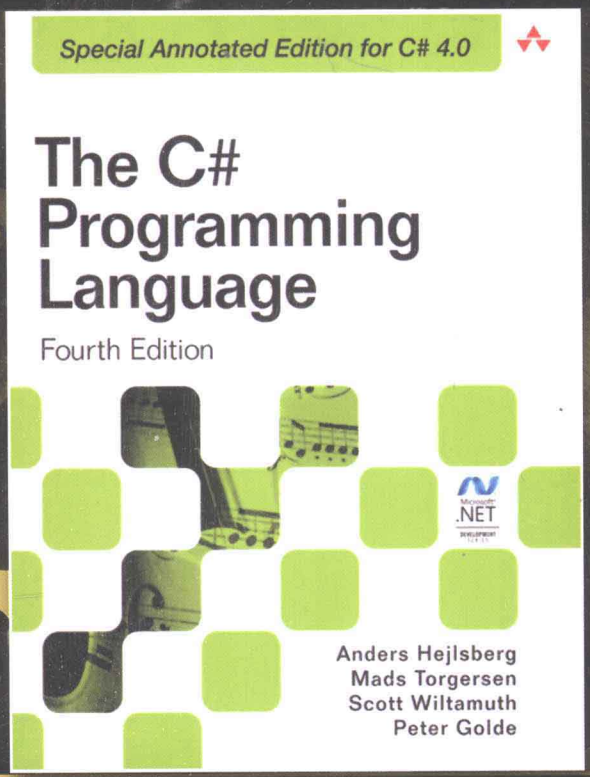
原书第4版

C#程序设计语言

(美) Anders Hejlsberg Mads Torgersen Scott Wiltamuth Peter Golde 著 陈宝国 黄俊莲 马燕新 译

The C# Programming Language

Fourth Edition



机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

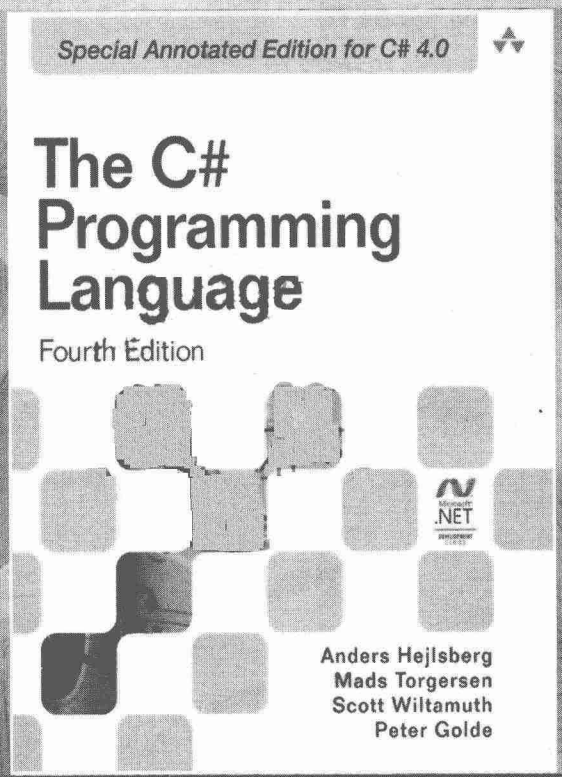
原书第4版

C#程序设计语言

(美) Anders Hejlsberg Mads Torgersen Scott Wiltamuth Peter Golde 著 陈宝国 黄俊莲 马燕新 译

The C# Programming Language

Fourth Edition



机械工业出版社
China Machine Press

C#语言结合了快速应用开发语言的高效和C/C++语言的强大。C# 4.0是对C# 3.0的进一步完善和扩展，它在C# 3.0的基础上引入了以动态语言为主的新特色。本书由C#的缔造者Anders Hejlsberg和他的同事们合著，全部内容都更新到了C# 4.0版。本书提供了C# 4.0语言完整的规格说明、参考资料、范例代码和来自12位卓越的C#大师的详细注解。这些注解所达到的深度和广度在其他书中难得一见。本书的正文介绍了C#的概念，而这些恰到好处的注解则解释了为什么这些特性是重要的，应该怎么使用它们，它们和其他语言的关系是什么，甚至它们是如何演化而来的。

对任何希望深入理解C#的程序员来说，这本书都是不容错过的经典参考书。

本书版权登记号：图字：01-2010-7270

图书在版编目（CIP）数据

C#程序设计语言（原书第4版）/（美）海杰尔斯伯格（Hejlsberg, A.）等著；陈宝国，黄俊莲，马燕新译. —北京：机械工业出版社，2011.6
（计算机科学丛书）
书名原文：The C# Programming Language, Fourth Edition

ISBN 978-7-111-34778-1

Ⅰ. C…　Ⅱ. ① 海…　② 陈…　③ 黄…　④ 马…　Ⅲ. C语言－程序设计　Ⅳ. TP312

中国版本图书馆CIP数据核字（2011）第093191号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：秦　健
北京瑞德印刷有限公司印刷
2011年8月第1版第1次印刷
185mm × 260mm · 36印张
标准书号：ISBN 978-7-111-34778-1
定价：99.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

译 者 序

The C# Programming Language, 4E

C#一直是程序员使用.NET的主要方式。如果离开了C#，.NET平台的强大也就无从谈起了，想要获得哪怕是今天一半的成功，可能也是根本没有希望的。C#从一开始就把.NET提升到了一个非常高的成功起点上。随着.NET的不断发展，C#的重要性也与日俱增。在共同发展的过程中，C#也开发出了大量优秀的与平台无关的语言特性，从而与.NET平台自身的创新交相辉映，成为一对最佳拍档。

C# 4.0是对C# 3.0的进一步完善和扩展，它在C# 3.0的基础上又引入了以动态语言为主的新特色，例如，C#4.0加入了dynamic关键字，可以将变量的static类型声明为dynamic，虽然名为dynamic，但它还属于一种静态类型，而把静态类型声明为dynamic之后，该类型的操作就只会在运行时进行解析，我们从中不难看出JavaScript的身影，这也标志着它离动态语言又近了一步，使得开发人员的编程效率得到进一步提高。这说明，虽然C#从本质上说仍然属于一种静态语言，但是对象的意义已经不再是纯粹的静态对象了，它们开始有了动态特征，这既体现在它们的结构上，也体现在它们的行为上，至少从编译器的角度来看是这样的。

本书无疑是讲解C# 4.0的经典之作，我想这一点各位读者早就赞同了，无须我再赘述。本书还有一大特色就是贯穿全书的注解，正如Don Box所说“每个.NET程序员在读这本书的时候都至少会有一次‘啊，原来如此’的感叹”，而我发现这些注解是引起我发出这样的感叹最多的地方。在本书中，各位C#专家提供了新一轮更深入的见解，堪称点睛之笔。

本书由黄俊莲、陈宝国、马燕新翻译完成。由于译者水平有限，在翻译过程中难免会有一些错误，恳请读者批评指正。

译 者

2011年4月

序　言

The C# Programming Language, 4E

自2000年夏.NET发布以来，已经过去十个年头了。在我看来，当时.NET最重要的两点就是结合了托管代码的本地执行和用于程序之间通信的XML消息机制。然而那个时候我还没有意识到C#会变得那么重要。

C#从一开始就是程序员理解和使用.NET的主要方式。如果你问一个普通的.NET程序员，一个值类型和一个引用类型的区别是什么，他会不假思索地回答“结构和类的区别”，而不会回答“一个是从System.ValueType继承的类型，而另一个不是从System.ValueType继承的类型”。原因何在？因为我们都是用语言，而不是通过API来和运行时（更重要的是，和其他人）交流想法和意图的。

如果没有一门出色的语言，一个平台要想成功是不可能的。C#最初就为人们如何看待.NET打下了坚实的基础。随着.NET的不断发展，C#的重要性也与日俱增，诸如迭代器和真正的闭包（也称为匿名方法），都是用C#编译器实现的纯语言特性，而不是平台自带的特性。C# 3.0的发布更意味着C#成为了.NET不断创新的主角，它引入了标准化的查询操作符，简洁的lambda表达式、扩展方法，以及在运行时访问表达式树的能力，而这些都是通过语言和编译器的开发实现的。C# 4.0最重要的特性（动态调用）也主要是语言和编译器的特性，而不是对CLR自身的改进。

说到C#就一定会提到它的缔造者Anders Hejlsberg，他像一位牧羊人一样坚守着他创造的这门语言。我非常荣幸地在C# 3.0设计阶段连续好几个月参加了C#的设计会议，Anders的工作让我大开眼界。他有着出色的天赋，深谙程序员喜欢什么和不喜欢什么，同时他又能和设计团队紧密合作，并最终获得最佳的设计方案。

特别是在C# 3.0上，在从函数式语言社区获取灵感并将它们带给广大开发者的过程中，Anders展现出了无与伦比的能力，要知道这绝对不是一件容易的事情。Guy Steele曾经在谈论Java时说道：“我们没打算要吸引Lisp程序员，我们的目标是C++程序员。我们成功地把他们从转向Lisp的路上吸引过来。”当我看到C# 3.0的时候，我就知道C#已经至少获得了一名C++程序员（就是我自己啦）的青睐。C# 4.0向着Lisp（以及JavaScript、Ptython、Ruby等）又靠近了一步，它增加了在不依赖静态类型定义的情况下编写程序的能力。

虽然C#很出色，但是为了理解它的精妙之处，并确保所有人都使用一个一致的C#，我们还是需要有一个参考文档——一个用自然语言（也就是英语）写成的并遵守一种统一范式（BNF范式）的文档。而你手中的这本书正是这样的一份文档。根据我的经验，我敢说每个.NET程序员在读本书时都至少会有一次“啊，原来如此”的感叹，它能让你的专业水平更上一层楼。

请享受阅读本书所带来的快乐吧。

Don Box

雷德蒙，华盛顿

2010年5月

前　言

The C# Programming Language, 4E

C#项目始于12年前的1998年12月，当初的目标是要为全新的（还未命名的）.NET平台创建一种简单、现代化、面向对象和类型安全的编程语言。一路走来，C#也算是历经坎坷。现在这门语言已经拥有超过一百万的程序员，发布了4个版本，其中每一个版本都加入了许多重要的新特性。

本书也同样出到了第4版。它提供了C#编程语言的完整技术规范，而且有两点内容是前几版没有的。当然，最显著的就是它涵盖了C# 4.0的新特性，包括动态绑定、指定和可选参数以及协变（covariant）与逆变（contravariant）泛型。这一版修订的主要目的是加强C#与.NET环境外部的对象之间的交互。C# 3.0中的LINQ使得用来访问外部数据源的代码的语言集成度更高，与之相比，C# 4.0的动态绑定使得C#与那些来自动态编程语言（例如Python、Ruby和JavaScript）的对象之间的交互更加自然，就好像是在与C#的本地对象进行交互一样。

在本书的前一版中，一些著名的C#专家提供了大量的注解。我们不断收到这方面的积极反馈，而且也很高兴在这一版中融入了一些新的和原来的注解者提供的新一轮更深入、更有趣的见解、指导、背景和观点。我们也非常高兴地看到这些注解和本书的核心内容相辅相成，让C#的特性跃然纸上。

创建C#语言是很多人共同努力的结果。C# 1.0的设计团队由Anders Hejlsberg、Scott Wiltamuth、Peter Golde、Peter Sollich和Eric Gunnerson组成。而C# 2.0团队的成员有Anders Hejlsberg、Peter Golde、Peter Hallam、Shon Katzenberger、Todd Proebsting和Anson Horton。

此外，C#和.NET公共语言运行时（Common Language Runtime）中泛型的设计和实现是基于微软研究院里Don Syme和Andrew Kennedy构建的“Gyro”原型之上。C# 3.0则是由Anders Hejlsberg、Erik Meijer、Matt Warren、Mads Torgersen、Peter Hallam和Dinesh Kulkarni负责设计。C# 4.0的设计团队则由Anders Hejlsberg、Matt Warren、Mads Torgersen、Eric Lippert、Jim Hugunin、Lucian Wischik和Neal Gafter组成。

要感谢所有影响了C#设计的人是不可能的，尽管如此我们还是要感谢你们。闭门造车是不会有好设计的，所以来自庞大和热情的程序员社区的意见和建议都是无价的。

C#是（并将继续是）我们工作过的最有挑战性和最令人振奋的项目之一。希望你们用得开心，我们也做得高兴。

Anders Hejlsberg
Mads Torgersen
Scott Wiltamuth
西雅图，华盛顿
2010年9月

作者简介

The C# Programming Language, 4E

Anders Hejlsberg是编程界的传奇人物。他是C#语言的架构师，同时也是微软技术专家。他于1996年加入微软，之前13年的职业生涯则是在Borland渡过，在那里他曾经是Delphi和Turbo Pascal的首席架构师。

Mads Torgersen是微软C#的项目经理，负责日常语言设计工作以及维护C#语言的规范。

Scott Wiltamuth是微软Visual Studio Professional团队的合作项目主管。他在微软参与过很多开发工具的工作，包括OLE Automation、Visual Basic，Visual Basic for Applications、VBScript、JScript、Visual J++和Visual C#。

Peter Golde是最初的Microsoft C#编译器的首席程序员。作为微软在ECMA委员会（这个委员会负责C#的标准化工作）的主要代表，他领导实现了编译器并参与了语言的设计工作。他目前在微软公司是负责编译器的架构师。

注解者简介

Brad Abrams是CLR和.NET架构团队在微软的创始人之一，现在他是微软WCF和WF的项目经理。他从1998年起就参与了.NET框架的设计，当时他最初的框架设计工作是构建.NET框架的核心组件BCL（基本类库）。Brad 1997年毕业于北卡罗来纳州立大学，获得计算机科学学士学位。Brad出版的书有《Framework Design Guidelines, Second Edition》（Addison-Wesley，2009）和《.NET Framework Standard Library Annotated Reference》（第1、2卷）（Addison-Wesley，2006）。

Joseph Albahari是《C# 3.0 in a Nutshell》（O'reilly，2007），《C# 3.0 Pocket Reference》（O'reilly，2008）和《LINQ Pocket Reference》（O'reilly，2008）的作者之一。他是在保健、教育和电信领域里有17年经验的资深程序员和软件架构师，他还是LINQPad的作者，这是一个可以用LINQ交互查询数据库的工具。

Krzysztof Cwalina是微软.NET框架团队的首席架构师，最初在微软负责框架的API设计。现在他主要领导开发和推广设计和架构标准的工作，并将这些标准应用到.NET框架的开发中。他是《Framework Design Guidelines》（Addison-Wesley，2005）的作者之一，他的博客是http://blogs.msdn.com/kcwalina。

Jesse Liberty（"Silverlight专家"）是微软的资深项目经理，还是多部畅销书的作者，他也写了数十篇非常吸引人的文章，同时也经常在世界各地的各种活动中发表演讲。他的博客是http://JesseLiberty.com，这个博客是Silverlight、WPF和Windows Phone 7开发人员的必读博客。Jesse有20多年的实践编程经验，曾担任Citi的副总裁，并在AT&T被评为杰出软件工程师。你可以通过他的个人博客联系他，也可以通过@JesseLiberty在Twitter上找到他。

Eric Lippert是微软C#编译器团队的资深程序员。他参与设计和实现了Visual Basic、VBScript、JScript、C#和Visual Studio Tools For Office。他的博客是http://blogs.msdn.com/EricLippert，这个博客上既有上述所有这些主题，也有一些其他内容。

Christian Nagel是微软的地区总裁和MVP。他是几本书的作者，包括《Professional C# 4 with .NET 4》（Wrox，2010）和《Enterprise Services with the .NET Framework》（Addison-Wesley，2005）。作为CN创新的创始者和thinktecture的合作者，他为软件开发人员讲授和辅导各种Microsoft .NET技术。你可以通过http://www.cninnovation.com联系他。

Vladimir Reshetnikov是微软Visual C#的MVP。他有超过8年的软件开发经验，以及6年Microsoft .NET和C#的经验。你可以通过他的博客http://nikov-thoughts.blogspot.com联系他。

Marek Safar是Novell C#编译器团队的首席程序员。过去5年来，他一直致力于开发Mono C#编译器的主要特性。你可以通过他的博客http://mareksafar.blogspot.com联系他。

Chris Sells是微软Business Platform分部（又名SQL Server分部）的项目经理。Chris写了好几本书，包括《Programming WPF》（O'Reilly，2007）、《Windows Forms 2.0 Programming》（Addison-Wesley，2006）和《ATL Internal》（Addison-Wesley，1999）。闲暇的时候，他还主持了很多研讨会，并且在微软内部的产品讨论组里也相当活跃。要想了解Chris和他的项目，请访问http://www.sellsbrothers.com。

Peter Sestoft是丹麦哥本哈根IT大学软件开发专业的教授。自2003至2006年，他是ECMA International C#标准化委员会的成员之一，他写的书包括《C# Precisely》（MIT Press，2004）和《Java Precisely》（MIT Press，2005）。你可以通过http://www.itu.dk/people/sestoft联系他。

Jon Skeet是《C# in Depth》（Manning，2010）的作者，他是C#的MVP。他在伦敦的Google公司工作，闲暇时间则写一些C#的文章，并经常发表演讲。他的博客是http://msmvps.com/jon.skeet，他也经常在http:// stackoverflow.com上回答有关栈溢出（Stack Overflow）的问题。

Bill Wagner是SRT Solutions的创始人、微软的地区总裁和C#的MVP。他的整个职业生涯都在编写代码。他是《Effective C#》（Addison-Wesley，2005）和《More Effective C#》（Addison-Wesley，2009）的作者，《Visual Studio Magazine》的专栏作家，同时还是一位C# Developer Center在MSDN上的贡稿人。你可以通过他的博客http://srtsolutions.com/blogs/billwagner了解他不断演变的思想和其他主题的内容。

目　录

The C# Programming Language, 4E

第1章

The C# Programming Language, 4E

介　绍

C#（读作“see sharp”）是一门简单、现代化、面向对象、类型安全的编程语言。C#属于C语言家族，任何C、C++或是Java程序员都不会对它感到陌生。C#在ECMA International的标准是ECMA-334，在ISO/IEC上的标准是ISO/IEC 23270。微软.NET Framework上的C#编译器的实现同时遵循了这两个标准。

C#不只是一门面向对象的语言，它还包含了对面向组件（component-oriented）编程的支持。现代的软件设计越来越依赖于自包含和自我描述的功能包形式的软件组件。这类组件的重要之处在于它们提供了一个带有属性（property）、方法（method）和事件（event）的编程模型，拥有提供了关于组件声明信息的特性（attribute），以及包含了自身的文档。C#在语言级别上直接支持这些概念，令C#可以很自然地创建和使用软件组件。

C#还提供了一些特性来帮助构建健壮、耐用的应用程序：垃圾收集（Garbage Collection）会自动回收不再使用的对象所占用的内存，异常处理（exception hardling）提供了一种结构化且可扩展的方式来检测错误和恢复，而语言的类型安全（type-safe）设计则可以防止读取未初始化的变量、数组越界或是进行未检查的类型转换。

C#拥有统一的类型系统（unified type system）。所有的C#类型，包括int和double这样的基础类型，都是从根类型object继承而来。所以所有的类型都具有一些通用的操作，任何类型的值都可以通过一致的方式进行保存、传递和操作。此外，C#还支持用户自定义引用类型和值类型，允许动态分配对象和轻型结构的内联存储。

为了保证C#程序和类库以兼容性的方式向前发展，C#在设计过程中非常注重版本控制（versioning）。很多编程语言都对这一点重视不够，所以当新版本的依赖库被引入时，用这些语言编写的程序都无端地失灵了。C#设计中的很多方面都直接受到了版本控制考量的影响，包括区分virtual和override修饰符、方法重载的规则，以及对接口成员显式声明的支持。

在本章的剩余部分里，我们将介绍C#语言的基本特性。后面的章节会更细致（更精确）地讲解这些规则和例外，然而这里先尽量做一个简明的介绍，让读者可以先对语言有一个初步的认识，以便尽早开始编写程序和理解后面的章节。

CHRIS SELLS

我绝对同意“现代化、面向对象和类型安全”的论述，然而C#已经不再是一门简单的语言了。但是，随着C# 2.0加入的泛型和匿名委托，C# 3.0引入的LINQ特性以及C# 4.0的动态值，其程序本身正在变得更加简单易读，维护起来也更加容易，而这些正是所有编程语言的梦想。

ERIC LIPPERT

C#越来越像一门函数式编程语言。诸如类型推导（type inference）、lambda表达式和一元查询推导式（monadic query comprehension）这样的特性让传统的面向对象程序员可以利用函数式编程的思想来增加语言的表达能力。

CHRISTIAN NAGEL

C#不是一种纯粹的面向对象语言，而是为了提高它在其主要应用领域中的生产效率而不断扩展所得到的一种语言。用C#3.0编写的程序与利用C#1.0的函数式编程结构编写的程序看起来可能完全不同。

JON SKEET

C#的某些方面确实使该语言的功能越来越强，但同时，C# 3.0中也有越来越多的属性和对象初始化程序是自动实现的，也就是说，固定不变的东西越来越多了。在将来的版本中是否会出现更多鼓励这种不变性的功能，以及它们是否支持其他领域（如元组、模式匹配和尾递归），这值得我们期待。

BILL WAGNER

这一部分自从C#第一版以来就没有改变过。显然，语言是不断发展的并且其中加入了新的术语，但C#仍然是一门简单易学的语言。这些高级特性很容易学会，但并不是每个程序中都需要使用。对于没有经验的开发人员来说C#仍然是一门简单易学的语言。

1.1 Hello，World

介绍编程语言时最经典的自然就是“Hello, World”程序了。这里是C#版：

```
using System;

class Hello
{
    static void Main() {
        Console.WriteLine("Hello, World");
    }
}
```

C#的源文件通常是以.cs结尾。假设这个“Hello, World”程序保存为hello.cs，那么就可以在命令行下用微软C#编译器这样编译程序：

```
csc hello.cs
```

它会生成一个可执行文件hello.exe。这个程序执行的输出为：

```
Hello, World
```

“Hello, World”程序的开始通过using指令引用了System命名空间。命名空间提供了一种层次化的方式来组织C#程序和类库。它可以包含类型和其他命名空间，例如，System命名空间包含了好几种类型，例如，在程序中引用的Console类的其他一些命名空间，例如IO和Collections。通过using指令引用某个命名空间时，你就可以不加前缀使用这个命名空间里的类型。using指令让程序可以把System. Console.WriteLine简写为Console.WriteLine。

“Hello, World”程序声明的Hello类只有一个成员方法Main。Main方法在声明时要带上static修饰符。实例方法可以通过关键字this来引用自身，而静态方法在使用时却无需引用某个特定对象。按照惯例，静态方法Main是程序的入口点。

程序的输出是由命名空间System里的Console类的WriteLine方法产生的。在默认情况下，微软的C#编译器会自动引用这个由.NET框架类库提供的类。注意C#本身并没有一个单独的运行库，.NET框架就是C#的运行库。

BRAD ABRAMS

请注意：这里的Console.WriteLine()其实是Console.Out.WriteLine的简写。Console.Out是一个属性，它返回了一个用来专门向控制台输出的System.IO.TextWriter基类的实现。上面的例子写成这样也是正确的：

```
using System;
class Hello
{
    static void Main() {
        Console.Out.WriteLine("Hello, World");
    }
}
```

在框架设计的初期，我们就注意到了在C#语言规范中的这节给语言所带来的复杂性。我们选择了方便的重载Console来让“Hello, World”更加容易编写。事实上，今天你会发现几乎用不着去调用Console.Out.WriteLine()了，大家也都很喜欢这个决定。

1.2 程序结构

在C#中关键的组织概念是程序、命名空间、类型、成员和汇编代码。C#的程序由一个或多个源文件组成。程序声明了类型，而类型包含了成员。类型可以组织到命名空间里，类和接口都是类型的例子，变量、方法、属性事件则是成员的例子。当编译C#程序时，它们通常都会打包到汇编代码里去。它们的文件扩展名可以是.exe或是.dll，分别取决于它们实现的是应用程序还是类库。

下面这个例子：

```
using System;

namespace Acme.Collections
{
    public class Stack
    {
        Entry top;

        public void Push(object data) {
            top = new Entry(top, data);
        }

        public object Pop() {
            if (top == null)throw new InvalidOperationException();
            object result = top.data;
            top = top.next;
            return result;
        }

        class Entry
        {
            public Entry next;
            public object data;

            public Entry(Entry next, object data) {
                this.next = next;
                this.data = data;
```