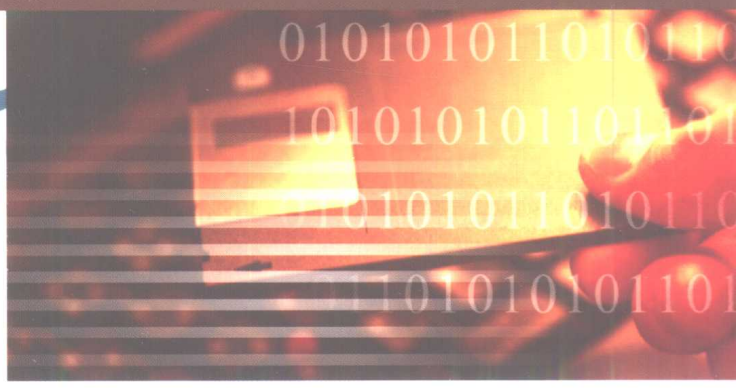
C#

程序设计案例教程



- C# 开发应用程序基本过程和编程要素案例剖析
- C# 的 Windows 应用程序设计及在数据库和网络中的应用

王　宏　编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

C# 程序设计案例教程

王 宏 编著

清 华 大 学 出 版 社

（京）新登字 158 号

内 容 简 介

C#是 Microsoft 新近推出的一门高级程序设计语言。本书以案例剖析的形式，对C#开放应用程序的基本过程、C#中比较特殊的编程要素、C#的 Windows 应用程序设计以及 C#在数据库和网络的应用等方面做了详细介绍。本书力求做到深入浅出，内容翔实。在介绍每一个例子的时候，不仅详细分析了例子本身，而且还对例子所涉及到的知识也做了阐述。

本书适合那些对C语言新技术有强烈兴趣的广大读者朋友，如果您有一定的C/C++基础，通过本书快速掌握 C#技术将会成为一件非常容易的事情。本书也适合高等院校及各类计算机培训班学员学习 C#语言。

书　名：C# 程序设计案例教程
作　者：王宏
出版者：清华大学出版社（北京清华大学学研大厦，邮编：100084）
http://www.tup.tsinghua.edu.cn
印刷者：北京鑫丰华彩印有限公司
发行者：新华书店总店北京发行所
开　本：787×1092　1/16　**印张**：19.25　**字数**：467 千字
版　次：2002 年 1 月第 1 版　2002 年 1 月第 1 次印刷
书　号：ISBN 7-302-04976-9/TP·2803
印　数：0001~ 4000
定　价：28.00 元

前　言

C#是 Microsoft 新近推出的一门高级程序设计语言。C#继承了 C++的大量语法要素和编程模式。在 C/C++的基础上也做了比较大的革新，这些革新措施使得 C#更简洁、更安全，由于采用了完全的面向对象的编程思想，使得 C#的代码的复用性更高。同时 C#通过和 Microsoft 同期推出的 CLI（Common Language Infrastructure，公共语言基础平台）的整合，能够实现最大程度上的语言之间的相互操作，解决了多年以来困扰软件产业的一大问题。C#还是一个开放的体系，它可以适应从桌面应用到网页浏览，从数据库到多媒体等多种领域。一名熟练的 C++程序员可以很容易地使用 C#进行程序设计。

本书以案例剖析的形式对 C#开放应用程序的基本过程、C#中比较特殊的编程要素、C#的 Windows 应用程序设计以及 C#在数据库和网络的应用等方面做了较为详细的介绍。

全书共由 15 个案例组成，根据它们涉及内容的侧重点，可以分成如下三个部分：

- 语言基础部分：这部分的程序包括案例 1~案例 7，程序都是采用 Console Application 程序模板生成，主要立足于介绍 C#中比较特殊的语言要素，如属性类的构造及其应用，同时对一些基本的程序设计技巧也做了介绍，如命令行分析等。
- Windows 程序设计：这部分内容包含案例 8～案例 13，程序都是采用 Windows Application 程序模板生成。这部分内容主要介绍如何通过 C#进行基于 Windows 用户界面的应用程序的设计，如何把用户界面和程序功能有机地结合在一起。
- 应用提高部分：这部分内容包含案例 14 和案例 15，其中案例 14 全面演示 C#开发数据库应用程序所涉及的内容，案例 15 则比较全面地说明了如何通过 C#开放基于网络的应用程序。

本书力求做到深入浅出，内容翔实。在介绍每一个案例时，不仅详细分析案例本身，而且还对案例涉及的知识进行了介绍。

本书由北京华源中科辅龙计算机技术有限责任公司策划，主要部分由王宏编写。另外参加编写的人员有：石利文、杨桂莲、邹杰、李炎、张玉玲、魏金兰、王艳燕、刘小华、余文娟、徐小宇、汪国华、刘树春、田剑等同志。全书由郭美山统稿，徐平校排。

本书所有的源代码和运行结果放在以下网址，读者可以自行下载。

URL：http://www.doudou.com.cn/html/Freesource/E-Tutorial.htm

由于时间仓促、作者水平有限，本书错漏之处在所难免，欢迎广大读者批评指正。

对本书内容有疑问的读者，可向抖斗书屋读者服务部提出咨询。

咨询电话：010-62565533　转 3308

E-mail：replybook @ 126.com

作者

2001 年 9 月于北京

目　录

引言　.Net 简介

C#是 Microsoft 推出的一门新的高级程序设计语言，它伴随着 Visual Studio.Net 一起发布。由于该语言充分吸取了计算机业界的最新成果，它的推出迅速引发了一股学习讨论 C#的热潮。本书在具体介绍 C#的编程技术之前，先对 C#的背景和一些基础知识做一个简要的介绍。

知识要点

- 什么是.Net
- .Net 的特点
- .Net 的组成
- C#的特点
- C#与 Java 的相同之处
- C#与 Java 的不同之处
- 基本数据类型
- C#中的基本运算符
- C#中的基本语句

Microsoft.Net 概述

2000 年 6 月，微软公司推出了“Microsoft.Net 下一代互联网软件和服务战略”，引起 IT 行业的广泛关注。2000 年 9 月，微软公司在旧金山发布了 Enterprise 2000。同月，微软原总裁兼首席执行官鲍尔默来到中国就“下一代互联网”的主题进行演讲，在中国掀起了一股“.Net 旋风”。2000 年 11 月，微软在 Comdex 计算机大展上发布了 Visual Studio.Net 软件，并展示了其.Net 发展战略的框架体系和开发工具的相关特性，全面加速了微软以.Net 技术进军市场的步伐。

什么是.Net

对于.Net 到底是什么这个问题，有着各种各样的说法。但当时作为首席执行官的鲍尔默应该最能代表微软公司的观点，他说：“Microsoft.Net 代表了一个集合、一个环境、一个可以作为平台支持下一代 Internet 的可编程结构。”确实，这句话基本上简单扼要地表述了.Net 的外特性。

.Net 首先是一个环境。这是一个理想化的未来互联网环境，微软的构想是一个“不再关注单个网站、单个设备与因特网相连的互联网环境，而是要让所有的计算机群、相关设备和服务商协同工作”的网络计算环境。简而言之，互联网提供的服务，要能够完成更高程度的自动化处理。未来的互联网，应该以一个整体服务的形式展现在最终用户面前，用户只需要知道自己想要什么，而不需要一步步地在网上搜索、操作来达到自己的目的。这是一种理想，但的的确确是互联网的发展趋势。

.Net 谋求的是一种理想的互联网环境。而要搭建这样一种互联网环境，首先需要解决的问题是针对现有因特网的缺陷，来设计和创造一种下一代 Internet 结构。这种结构不是物理网络层次上的拓扑结构，而是面向软件和应用层次的一种有别于浏览器只能静态浏览的可编程 Internet 软件结构。因此.Net 把自己定位为可以作为平台支持下一代 Internet 的可编程结构。

.Net 的最终目的就是让用户在任何地方、任何时间，以及利用任何设备都能访问他们所需要的信息、文件和程序。而用户不需要知道这些东西存在什么地方，甚至连如何获得等具体细节都不知道。他们只需发出请求，然后只管接收就是了，而所有后台的复杂性是完全屏蔽起来的。所以对于企业的 IT 人员来说，他们也不需要管理复杂的平台以及各种分布应用之间的工作是如何协调的。

.Net 的特点

.Net 包括四个重要特点，一是软件就是服务，二是基于 XML 的共同语言，三是融合多种设备和平台，四是新一代的人机界面。这四个特点基本上覆盖了.Net 的技术特征。

1. 软件就是服务

史蒂夫·鲍尔默在谈到软件服务时说道，“今天的软件产品仅仅是一张光盘，用户购买软件，亲自安装、管理和维护。但是软件服务是来自因特网的服务，它替用户安装、更新和跟踪这些软件，并让它们和用户一同在不同的机器间漫游。它为用户存储自己的信息和参考资料。这些就是软件和软件服务各自不同的风格。”

Orchestration 可视化编程工具产生基于 XML 的 XLANG 代码，它和 BizTalk 服务器、.Net Framework，以及 Visual Studio.Net 都曾是 Windows DNA 2000 战略的重要部分。

伴随着 ASP 产业的兴起，软件正逐渐从产品形式向服务形式转化，这是整个 IT 行业的大势所趋。在.Net 中，最终的软件应用是以 Web 服务的形式出现并在 Internet 上发布的。Web 服务是一种包装后的可以在 Web 上发布的组件，.Net 通过 WSDL 协议来描述和发布这种 Web 服务信息，通过 DISCO 协议来查找相关的服务，通过 SOAP 协议进行相关的简单对象传递和调用。

微软的.Net 战略意味着：微软公司以及在微软平台上的开发者将会制造服务，而不是制造软件。在未来几年之内，微软将陆续发布有关.Net 的平台和工具，用于在因特网上开发 Web 服务。那时，工作在.Net 上的用户、开发人员和 IT 工作人员都不再购买软件、安装软件和维护软件。取而代之的是，他们将定制服务，软件会自动安装，所有的维护和升

级也会通过互联网进行。

2. 基于 XML 的共同语言

XML 是从 SGML 语言演化而来的一种标记语言。作为元语言，它可以定义不同种类应用的数据交换语言。在.Net 体系结构中，XML 作为一种应用间无缝接合的手段，用于多种应用之间的数据采集与合并，用于不同应用之间的互操作和协同工作。具体而言，.Net 通过 XML 语言定义了简单对象访问协议（SOAP）、Web 服务描述语言（WSDL）、Web 服务发现协议（DISCO）。SOAP 协议提供了在无中心分布环境中使用 XML 交换结构化有类型数据的简单轻量的机制。WSDL 协议定义了服务描述文档的结构，如类型、消息、端口类型、端口和服务本身。DISCO 协议定义了如何从资源或者资源集合中提取服务描述文档、相关服务发现算法等。

3. 融合多种设备和平台

随着 Internet 逐渐成为一个信息和数据的中心，各种设备和服务已经或正在接入和融入 Internet，成为其中的一部分。.Net 谋求与各种 Internet 接入设备和平台的一体化，主要关注在无线设备和家庭网络设备及相关软件、平台方面。

4. 新一代的人机界面

新一代人机界面主要体现在“智能与互动”两个方面。.Net 包括通过自然语音、视觉、手写等多种模式的输入和表现方法；基于 XML 的可编辑复合信息架构——通用画布；个性化的信息代理服务；使机器能够更好地进行自动处理的智能标记等技术。

.Net 的组成

.Net 主要由 Windows.Net、.Net 框架、.Net 企业服务器、模块构建服务、Orchestration 和 Visual Studio.Net 构成。

1. Windows.Net

Windows.Net 是融入.Net 技术的 Windows，它将紧密地整合了.Net 的一系列核心构造模块，为数字媒体及应用间协同工作提供支持，是微软公司的下一代 Windows 桌面平台。

2. .Net 框架

.Net 框架(Framework)的目的是便于开发商更容易地建立网络应用程序和 Web 服务，它的关键特色是提供了一个多语言组件开发和执行的环境。从层次结构来看，.Net Framework 又包括三个主要组成部分：通用语言运行环境（Common LanguageRuntime）、服务框架（Services Framework）、上层的两类应用模板——面向 Web 的网络应用程序模板（Web Forms 或 Web Services）和 Windows 应用程序模板（Win Forms）。

其中通用语言运行环境在组件运行时，负责管理内存分配、启动和中止线程和进程、

强化安全系数，同时还调整任何该组件涉及到的其他组件的附件配置。在通用语言运行环境上是服务框架，它为开发人员提供了一套能够被任何现代编程语言调用的、统一的面向对象、异步、层次结构的可扩展类库，包括集合、输入/输出、字符串、图画、网络、线程、全球化、安全加密、数据库访问、调试相关服务等类库。在服务框架之上是两种应用类型的模板，一类是传统的 Windows 应用程序模板，另一类是基于 ASP+的 Web 网络应用程序模板。其中 ASP+以一组控件和体系结构的方式提供了一个 Web 应用模型，由.Net 框架提供的类库构建而成，通过它可以简化 Web 应用的实现过程。

3. .Net 企业服务器

在微软宣称的“第三代互联网”中，.Net 企业服务器是企业集成和管理所有基于 Web 的各种应用的基础，它提供企业未来开展电子商务的高可靠性、高性能、高可伸缩性以及高可管理性。

4. 模块构建服务

模块构建服务（Building Block Services）是.Net 平台中的核心网络服务集合，它主要包括以下几个组成部分：Internet XML 通信，使 Web 站点变成灵活的服务来交换和处理数据；Internet XML 数据空间，在 Web 商提供安全的和可编程的 XML 存储空间；Internet 动态更新，为快速开发和动态配置应用提供服务；Internet 日程安排，集成工作、社会和私人的日历；Internet 身份认证，提供从口令、钱包到生理数据等多级身份认证手段，还有 Internet 目录服务和 Internet 即时信息传递等服务。

5. Orchestration

Orchestration 是一种基于 XML 的面向应用的软件集成和自动化处理技术。它的目标是尽量不受时间、组织、应用及个人的限制，最大程度并最好地把集成技术和自动处理技术接合起来，以便商业事务能够交互、动态、可靠地进行下去。Orchestration 有三个基本要求：

- 处理与执行过程分离，即整个处理并不一定非要同执行的细节及途径绑定起来；
- 动态处理，即随着数据不同及交换的变化，整个操作过程必须随时动态更新改变；
- “Any to Any”集成，即整个处理过程不能对参与的平台、应用及协议等做出限制。

.Net 的 BizTalk Orchestration 是上述技术的一个实现，它包括一个可视化的设计环境、一套捆绑的工具和一个 Orchestration 引擎，用于业务流程处理、管理和调试。

6. Visual Studio.Net

Visual Studio.Net 是基于 XML 的编程工具和环境，它便于快速开发符合.Net 体系的软件服务，使其在独立设备、企业数据中心和因特网之间的传送更加容易。本书介绍的 C# 高级程序设计语言就是随着 Visual Studio 7.0 一起发布的一种最新的编程语言。

C#的特点

C#语言由C/C++演变而来。但是，它现代、简单、完全面向对象和类型安全。如果是C/C++程序员，学习曲线将会很平坦。许多C#语句直接借用程序员所喜爱的语言，包括表达式和操作符。关于C#最重要的一点：它是现代的编程语言。它简化和现代化了C++在类、命名空间、方法重载和异常处理等领域。摒弃了C++的复杂性，使它更易用、更少出错。对C#的易用有贡献的是减少了C++的一些特性，不再有宏、模板和多重继承。上述功能使编程更方便。

简单

C#具有C++所没有的一个优势就是学习简单。该语言首要的目标就是简单。在C#中，没有C++中流行的指针。默认地，所有的工作都放在受管理的代码中，在那里不允许如直接存取内存等不安全的操作。在C++中，有“::”、“.”和“->”操作符，它们用于命名空间、成员和引用。对于新手来说，操作符至今仍是学习的一道难关。C#弃用其他操作符，仅使用单个操作符“.”。

不必记住基于不同处理器架构的隐含的类型，甚至各种整型的变化范围。C#使用统一的类型系统，摒弃了C++多变的类型系统。这种系统允许程序员把各种类型作为一个对象，查看它是一个原始类型还是一个full-blown类。

现代

投入学习C#的努力是一笔大投资，因为C#是为编写NGWS（Next Generation Windows Service）应用程序的主要语言而设计。将会发现很多自己用C++可以实现或者很费力实现的功能，在C#中不过是一部分基本的功能而已。对于企业级的编程语言来说，新增的金融数据类型很受欢迎。程序员用到了一种新的十进制数据类型，它专用于金融计算方面。如果不喜欢这种现成简单的类型，根据应用程序的特殊需求，可以很容易地创建出新的一种数据类型。

前面已经提到，指针不再是程序员编程武器的一部分。全面的内存管理已经不是程序员的任务。运行时NGWS提供了一个垃圾收集器，负责C#程序中的内存管理。因内存和应用程序都受到管理，所以很必要增强类型安全，以确保应用的稳定性。

对于C++程序员，异常处理的确不是新的东西，但它是C#的主要功能。C#的异常处理与C++的不同点在于它是交叉语言的（运行时的另一个功能）。在没有C#之前，如果出现异常情况，必须由编程处理所有异常。但现在由于使用了基于异常的健壮的出错处理，已能处理大部分异常情况了。

面向对象

C#支持所有关键的面向对象的概念，如封装、继承和多态性。完整的 C#类模式构建在 NGWS 运行时的虚拟对象系统（VOS，Virtual Object System）的上层。对象模式只是基础的一部分，不再是编程语言的一部分。

程序员一开始必须关注的事，就是不再有全局函数、变量或者是常量。所有的东西都封装在类中，包括事例成员（通过类的事例，即对象可以访问）或静态成员（通过数据类型）。这些使 C#代码更加易读且有助于减少潜在的命名冲突。

定义类中的方法默认是非虚拟的（它们不能被派生类改写）。主要论点是，这样会消除由于偶尔改写方法而导致另外一些原码出错。要改写方法，必须具有显式的虚拟标志。这种行为不但缩减了虚拟函数表，而且还确保正确版本的控制。

使用 C++编写类，程序员可以使用访问权限（access modifiers）给类成员设置不同的访问等级。C#同样支持 private、protected 和 public 三种访问权限，而且还增加了第四种访问权限：internal。

程序员曾经创建了多少个类是从多基类派生出来的。大多数情况，仅需从一个类派生出。多基类惹出的麻烦通常比它们解决的问题还多，那就是为什么 C#仅允许一个基类的原因。如果程序员觉得需要多重继承，可以运用接口。

类型安全

在 C++中拥有一个指针，程序员能自由地把它强制转换成为任何类型，这并不是程序员所想象的企业级编程语言的类型安全。

和 C++不同，C#实施最严格的类型安全，以保护自己及垃圾收集器（garbage collector）。程序员不能使用没有初始化的变量。对于对象的成员变量，编译器负责清零。而局部变量，则由程序员负责清零。当程序员使用一个没有初始化的变量时，编译器会教程序员怎么做。这样做的目的在于：能够避免由于使用不经初始化的变量计算，结果导致错误，而程序员还不知道这些奇怪的结果是如何产生的。

C#取消了不安全的类型转换。不能把一个整型强制转换成一个引用类型（如对象），而当将引用类型转换成整型时，C#验证这种转换是正确的。边界检查是 C#的一部分。再也不会出现这种情况：当数组实际只定义了 n–1 个元素，却超额地使用了 n 个元素。

算术运算有可能溢出终值数据类型的范围。C#允许在语句级或应用程序级检测这些运算。在允许检测溢出的情况下，当溢出发生时将会抛出一个异常。在 C#中，被传递的引用参数是类型安全的。

版本控制

在过去的几年中，几乎所有的程序员都至少有一次不得不涉及到众所周知的“DLL 地狱”。该问题起因于多个应用程序都安装了相同 DLL 名字的不同版本。有时，老版本的

应用程序可以很好地和新版本的 DLL 一起工作，但是更多的时候它们会中断运行，现在的版本问题真是令人头痛。

NGWS runtime（runtime 是 C# 提供的采用动态链接库技术的运行环境）将对程序员所写的应用程序提供版本支持。C#可以最好地支持版本控制。尽管 C#不能确保正确的版本控制，但是它可以为程序员保证版本控制成为可能。有这种支持，一个开发人员就可以确保当他的类库升级时，仍保留着对已存在的客户应用程序的二进制兼容。

兼容

C#并没有存在于一个封闭的世界中。它允许使用最先进的 NGWS 的通用语言规定（CLS，Common Language Specification）访问不同的 API。CLS 规定了一个标准，用于符合这种标准语言内部之间的操作。为了加强 CLS 的编译，C#编译器检测所有的公共出口编译，并在编译通不过时列出错误。

当然，程序员能够访问旧一点的 COM 对象。NGWS 运行时提供对 COM 透明的访问。OLE 自动化是一种特殊的机制。任一个使用 C++创建 OLE 自动化项目的人已经喜欢上各种各样的自动化数据类型。有个好消息就是 C#支持它们，而没有繁琐的细节。

最后，C#允许程序员用 C 原型的 API 实现内部操作。可以从程序员的应用程序访问任何 DLL 中的入口点（有 C 的原型）。用于访问原始 API 的功能称做平台调用服务（PInovke，Plaform Invocation Services）。

灵活

尽管 C#代码的默认状态是类型安全的，但是程序员可以声明一些类或者仅声明类的方法是非安全类型的。这样的声明允许程序员使用指针、结构，静态地分配数组。安全码和非安全码都运行在同一个管理空间，这暗示着当从安全码调用非安全码时不会陷入列集（marshaling，这是安全码和非安全码的内存地址错乱的状态，是专用名称）。

C#与 Java

Java 是 Sun 公司推出的一款高级程序设计语言，Java 语言推出的时候也使计算机业界引起了空前的轰动，并迅速成为主流编程语言之一。由于 Java 语言也是以 C++为蓝本而发展起来的，同时 Java 语言也定位于网络应用，因此，不可避免，C＃和 Java 语言之间具有很多的相似之处，当然，作为两种各成体系的编程语言，它们之间的差别也是非常明显的。下面将 Java 和 C#之间的异同做一个简要的比较。

C#与 Java 的相同之处

由于 C#与 Java 都是基于 C++发展起来的，因此二者之间具有很多相似之处，这些相似之处包括：

- C#和 Java 语言的编译结果是独立于计算机和编程语言的，可执行文件可以在受管理的执行环境中执行；
- C#和 Java 语言都采用了自动的垃圾回收机制；
- C#和 Java 语言都取消了指针操作；
- C#和 Java 语言都没有头文件；
- C#和 Java 语言都只支持单重继承，要实现与多重继承类似的功能，必须通过接口来实现；
- 类都是从 Object 类派生而来，类的对象通过关键字 new 生成；
- C#和 Java 语言都支持线程；
- C#和 Java 语言都没有全局变量和全局函数，所有的变量和函数都属于某个类所有；
- C#和 Java 语言都支持对数组和字符串边界的严格检查，不会出现边界溢出的情况；
- C#和 Java 语言都使用“.”操作符，不再使用“->”和“::”操作符；
- C#和 Java 语言都将 null 和 bool 作为关键字；
- C#和 Java 语言中所有的值都必须先初始化后才能使用；
- C#和 Java 语言中的 if 语句都不允许采用整数作为判断条件；
- C#和 Java 语言的 try 语句块都可以后接 finally 语句块。

C#与 Java 的不同之处

尽管 C#和 Java 有很多相同之处，但是由于二者是两家不同公司开发的高级程序设计语言，它们又相对独立，自成体系，各自具有一些自己特有的特点。下面将 Java 和 C#之间的不同之处做一个简要的介绍。

1. 属性

对于那些经常使用快速开放工具，如 Delphi 或者 Visual Basic 的开放人员来说，属性是一个非常熟悉的概念。一般来说，通过 get×××可以读取属性的值，而通过 set×××方法则可以设置属性的值，下面是在 Java 中比较常见的属性操作语句：

```
foo.setSize(foo.getSize()+1);
label.getFont().setBold(true);
```

在 C#中可以通过如下语句来实现类似的功能：

```
foo.size++;
```

```
label.font.bold=true;
```

很明显，上述的属性设置方式较 Java 来说更为简洁，可读性也更强。这充分体现了C#简单的特点。

对于属性的定义，Java语言的实现方式如下：

```
public int getSize()
{
   return size;
}
public void setSize(int value)
{
    size=value;
}
```

C#对此也进行了简化，在C#中可通过如下方式来实现和上述程序类似的功能：

```
public int Size
{
    get
    {
        return size;
    }
    set
    {
        size=value;
    }
}
```

2. index

C#提供 index 来给对象加上索引的功能，从而用与处理数组类似的方式来处理对象，Java 语言则不支持 index。C#中定义 Index 的典型方式如下：

```
public Story this [int index]
{
    get
    {
        return stories [index];
    }
    set
    {
        if (value!=null)
        {
            stories [index]=value;
        }
    }
}
```

3. delegate

delegate 可以认为是一种类型安全、面向对象的函数指针。使用 delegate 可以通过一个名字访问不同的函数，它实现和 Java 中的 interface 类似的功能，但是它比 interface 更为好用。

4. event

C#提供对 event 的直接支持，它通过 delegate 和 event 关键字实现对事件的处理。event 关键字隐藏所有 delegate 方法，运算符“+=”和“-=”允许程序员自由加入或者删除时间处理程序。

5. enum

枚举用于指定一系列的对象，C#通过如下所示的语句来定义枚举：

```
public enum Direction{North,East,West,South};
```

在 C#中可以通过如下语句实现对枚举的使用：

```
Direction wall=Direction.North;
```

Java 不直接支持枚举，如果要实现和上述程序类似的功能，则 Java 必须通过如下的程序来实现：

```
public class Direction
{
    public final static int NORTH=1;
    public final static int EAST=2;
    public final static int WEST=3;
    public final static int SOUTH=4;
}
```

在定义了类 Direction 之后，Java 程序可以通过如下语句使用枚举：

```
int wall=Direction.NORTH;
```

6. foreach 语句

C#提供类标准的 for 循环，同时还提供了 foreach 语句来循环处理集合中的元素。Java 语言中，遍历集合中的所有元素的典型处理方式如下：

```
while (!collection.isEmpty())
{
    Object o=collection.get();
    collection.next();
    …
}
```

在 C#中则可以通过如下语句加以实现：

```
foreach(object o in collection)
{
    …
}
```

7．统一数据类型

大多数的高级程序设计语言都有基本数据类型，如整型、浮点类型等。同时，为了更好地满足实际的需要，这些高级语言大多也定义一些复杂数据类型，如结构体和类等。传统的高级语言中，对不同的数据类型有不同的处理方式，显然，如果能够将对简单数据类型的处理和对复杂数据类型的处理结合在一起，并用一种一致的方式加以处理的话，无疑会大大提升应用程序设计的效率，增强程序设计的灵活性。

Java 语言在处理基本数据类型的时候也采取分别处理的策略，但是在基本数据类型的基础上提供了一系列封装这些基本数据类型的类，例如，整型（int）被类 Integer 所封装，双精度浮点数（double）被类 Double 所封装。

C#提供了一种和 Java 不同的方式来实现数据类型的统一。事实上，在 C#中，即使是 int 这样的简单数据类型在 C#内部也是通过一个结构体 Int32 来实现的，在 C#中，可以这样认为，int 只是结构体 Int32 的一个别名。

由于 C#中的结构体也继承自类 Object，这样，Object 类中定义的方法，各个结构体也拥有，于是，在 C#中可以通过如下的方式来操纵整数：

```
int i=5;
System.Console.WriteLine(i.ToString());
```

8．操作符重载

通过操作符重载可以用一种比较自然的方式来操纵各种数据类型，从而大大提升程序的可读性和灵活性。C#中的“==”操作符在 Object 类中进行了定义，在 Object 中定义的==操作符通过比较两个值的引用来获得最后的结果。如果使用和集合相关的类，则必须在这样的类中实现 ICompar 接口，这个接口中定义了一个方法 CompareTo，该方法返回两个对象的比较结果，在此基础上，可以进一步定义各个实现比较的操作符，如“>”、“<”、“>=”和“<=”等。事实上，数字类型（如 int、long 等）可以直接使用这些比较操作符，它们的内部都实现了 ICompare 接口。

9．多态性

虚拟方法提供了多态性的支持。多态意味着派生类可以定义一个和基类中同名的方法。尽管 Java 和 C#都支持多态性，但是它们的具体实现方式还是有一定的差别的。

在 Java 语言中，默认情况下，基类的对象可以直接调用派生类中的虚拟方法，在 C#语言中，基类要调用派生类中的虚拟方法必须通过 virtual 关键字来实现。同时，在 C#语言中，一个方法要重载基类中的同名方法，还必须通过关键字 override 来实现。在 C#中实