参考：<https://www.cnblogs.com/chenssy/p/3250409.html>

程序开发过程中，如果一个项目的不同的模块出于各种考虑提供了不同的集合对象，在一个总体的管理模块中如果要遍历这些不同的集合就会不方便。

例如，对电视机的电影、电视频道进行管理，分别让A和B来实现。

如：C:\study\C-sharpProject\iterator\_test1

public class MenuItem

{

public MenuItem(int channel, string name, string description)

{

Channel = channel;

Name = name;

Description = description;

}

public string Name { get; set; }

public string Description { get; set; }

public int Channel { get; set; }

}

public abstract class TelevisionMenu

{

public abstract void AddItem(int channel, string name, string description);

}

public class TVChannelMenu : TelevisionMenu

{

// 使用List集合

private *List*<MenuItem> menuItems = new *List*<MenuItem>();

public TVChannelMenu()

{

AddItem(1, "CCTV-1", "This is CCTV-1");

AddItem(2, "CCTV-2", "This is CCTV-2");

AddItem(3, "CCTV-3", "This is CCTV-3");

}

public override void AddItem(int channel, string name, string description)

{

MenuItem tvMenuItem = new MenuItem(channel, name, description);

menuItems.*Add*(tvMenuItem);

}

public *List*<MenuItem> GetMenuItems()

{

return menuItems;

}

}

public class FilmMenu : TelevisionMenu

{

const int MAX\_ITEMS = 5;

MenuItem[] menuItems = new MenuItem[MAX\_ITEMS];

int numberOfItems = 0;

public FilmMenu()

{

AddItem(1, "绝世天劫", "这是好电影");

AddItem(2, "达芬奇密码", "这是最喜欢的电影");

}

public override void AddItem(int channel, string name, string description)

{

MenuItem filmMenuItem = new MenuItem(channel, name, description);

if (numberOfItems < MAX\_ITEMS)

{

menuItems[numberOfItems] = filmMenuItem;

++numberOfItems;

}

}

public MenuItem[] GetMenuItems()

{

return menuItems;

}

}

class Program

{

static void Main(string[] args)

{

TVChannelMenu tvMenuItems = new TVChannelMenu();

FilmMenu filmMenuItems = new FilmMenu();

// 遍历电视频道

for (int i = 0; i < tvMenuItems.GetMenuItems().*Count*; ++i)

{

}

// 遍历电影频道

for (int j = 0; j < filmMenuItems.GetMenuItems().*Length*; ++j)

{

}

}

}

A使用List来实现电视频道的菜单集合，B使用数组来实现电影频道的菜单集合，这样在遍历时，就需要分开遍历。

使用迭代器模式来改进：

迭代器模式：

在实际的开发过程中，我们可能需要针对不同的需求，可能需要以不同的方式来遍历整个整合对象，但是我们不希望在集合对象的抽象接口层中充斥着各种不同的遍历操作。这个时候我们就需要这样一种东西，它应该具备如下三个功能：

1、能够遍历一个集合对象。

2、我们不需要了解集合对象的内部结构。

3、能够提供多种不同的遍历方式。

这三个功能就是迭代器模式需要解决的问题。作为一个功能强大的模式，迭代器模式把在元素之间游走的责任交给迭代器，而不是集合对象。这样做就简化了集合的接口和实现，也可以让集合更专注在它所应该专注的事情上，这样做就更加符合单一责任原则。

例：C:\study\C-sharpProject\iterator\_test2

public class MenuItem

{

public MenuItem(int channel, string name, string description)

{

Channel = channel;

Name = name;

Description = description;

}

public string Name { get; set; }

public string Description { get; set; }

public int Channel { get; set; }

}

// 抽象迭代器接口

public interface Iterator

{

bool HasNext();

object Next();

}

// 电视频道迭代器

public class TVChannelMenuIterator : Iterator

{

private *List*<MenuItem> menuItems;

private int position = 0;

public TVChannelMenuIterator(*List*<MenuItem> menuItems)

{

this.menuItems = menuItems;

}

public bool HasNext()

{

if ((position > menuItems.*Count* - 1) || (menuItems[position] == null))

{

return false;

}

return true;

}

public object Next()

{

MenuItem item = menuItems[position];

++position;

return item;

}

}

// 电影频道迭代器

public class FilmMenuIterator : Iterator

{

private MenuItem[] menuItems;

int position = 0;

public FilmMenuIterator(MenuItem[] menuItems)

{

this.menuItems = menuItems;

}

public bool HasNext()

{

if ((position > menuItems.*Length* - 1) || (menuItems[position] == null))

{

return false;

}

return true;

}

public object Next()

{

MenuItem item = menuItems[position];

++position;

return item;

}

}

public interface TelevisionMenu

{

void AddItem(int channel, string name, string description);

Iterator CreateIterator();

}

public class TVChannelMenu : TelevisionMenu

{

// 使用List集合

private *List*<MenuItem> menuItems = new *List*<MenuItem>();

public TVChannelMenu()

{

AddItem(1, "CCTV-1", "This is CCTV-1");

AddItem(2, "CCTV-2", "This is CCTV-2");

AddItem(3, "CCTV-3", "This is CCTV-3");

}

public void AddItem(int channel, string name, string description)

{

MenuItem tvMenuItem = new MenuItem(channel, name, description);

menuItems.*Add*(tvMenuItem);

}

public Iterator CreateIterator()

{

return new TVChannelMenuIterator(menuItems);

}

}

public class FilmMenu : TelevisionMenu

{

const int MAX\_ITEMS = 5;

MenuItem[] menuItems = new MenuItem[MAX\_ITEMS];

int numberOfItems = 0;

public FilmMenu()

{

AddItem(1, "绝世天劫", "这是好电影");

AddItem(2, "达芬奇密码", "这是最喜欢的电影");

}

public void AddItem(int channel, string name, string description)

{

MenuItem filmMenuItem = new MenuItem(channel, name, description);

if (numberOfItems < MAX\_ITEMS)

{

menuItems[numberOfItems] = filmMenuItem;

++numberOfItems;

}

}

public Iterator CreateIterator()

{

return new FilmMenuIterator(menuItems);

}

}

public class MainMenu

{

private TVChannelMenu tvMenu;

private FilmMenu filmMenu;

public MainMenu(TVChannelMenu tvMenu, FilmMenu filmMenu)

{

this.tvMenu = tvMenu;

this.filmMenu = filmMenu;

}

public void PrintMenu()

{

Iterator tvIterator = tvMenu.CreateIterator();

Iterator filmIterator = filmMenu.CreateIterator();

*Console*.*WriteLine*("电视节目有：");

PrintMenu(tvIterator);

*Console*.*WriteLine*("电影节目有：");

PrintMenu(filmIterator);

}

private void PrintMenu(Iterator iter)

{

while (iter.HasNext())

{

MenuItem item = (MenuItem)iter.Next();

*Console*.*Write*("channel: " + item.Channel + ", ");

*Console*.*Write*("name: " + item.Name + ", ");

*Console*.*WriteLine*("description: " + item.Description);

}

}

}

class Program

{

static void Main(string[] args)

{

TVChannelMenu tvMenu = new TVChannelMenu();

FilmMenu filmMenu = new FilmMenu();

MainMenu mainMenu = new MainMenu(tvMenu, filmMenu);

mainMenu.PrintMenu();

}

}

使用了迭代器模式之后，在MainMenu遍历电视频道和电影频道，就不用关心它们内部的集合对象是List还是数组，抑或是其他的集合类型了。

使用C#的特定语法进行改进：C:\study\C-sharpProject\iterator\_test3

public class MenuItem

{

public MenuItem(int channel, string name, string description)

{

Channel = channel;

Name = name;

Description = description;

}

public string Name { get; set; }

public string Description { get; set; }

public int Channel { get; set; }

}

// 电视频道枚举器

public class TVChannelMenuEnumerator : *IEnumerator*

{

private *List*<MenuItem> menuItems;

private int position = -1;

public TVChannelMenuEnumerator(*List*<MenuItem> menuItems)

{

this.menuItems = menuItems;

}

public object Current

{

get

{

if (position == -1 || position >= menuItems.*Count*)

{

throw new *InvalidOperationException*();

}

return menuItems[position];

}

}

public bool MoveNext()

{

if (position < menuItems.*Count* - 1)

{

++position;

return true;

}

return false;

}

public void Reset()

{

position = -1;

}

}

// 电影频道枚举器

public class FilmMenuEnumerator : *IEnumerator*

{

private MenuItem[] menuItems;

private int position = -1;

public FilmMenuEnumerator(MenuItem[] menuItems)

{

this.menuItems = menuItems;

}

public object Current

{

get

{

if (position == -1 || position >= menuItems.*Length*)

{

throw new *InvalidOperationException*();

}

return menuItems[position];

}

}

public bool MoveNext()

{

if (position < menuItems.*Length* - 1)

{

++position;

if (menuItems[position] != null)

return true;

else

return false;

}

return false;

}

public void Reset()

{

position = -1;

}

}

public interface TelevisionMenu : *IEnumerable*

{

void AddItem(int channel, string name, string description);

}

public class TVChannelMenu : TelevisionMenu

{

// 使用List集合

private *List*<MenuItem> menuItems = new *List*<MenuItem>();

public TVChannelMenu()

{

AddItem(1, "CCTV-1", "This is CCTV-1");

AddItem(2, "CCTV-2", "This is CCTV-2");

AddItem(3, "CCTV-3", "This is CCTV-3");

}

public void AddItem(int channel, string name, string description)

{

MenuItem tvMenuItem = new MenuItem(channel, name, description);

menuItems.*Add*(tvMenuItem);

}

public *IEnumerator* GetEnumerator()

{

return new TVChannelMenuEnumerator(menuItems);

}

}

public class FilmMenu : TelevisionMenu

{

const int MAX\_ITEMS = 5;

MenuItem[] menuItems = new MenuItem[MAX\_ITEMS];

int numberOfItems = 0;

public FilmMenu()

{

AddItem(1, "绝世天劫", "这是好电影");

AddItem(2, "达芬奇密码", "这是最喜欢的电影");

}

public void AddItem(int channel, string name, string description)

{

if (numberOfItems < MAX\_ITEMS)

{

MenuItem filmMenuItem = new MenuItem(channel, name, description);

menuItems[numberOfItems] = filmMenuItem;

++numberOfItems;

}

}

public *IEnumerator* GetEnumerator()

{

return new FilmMenuEnumerator(menuItems);

}

}

public class MainMenu

{

private TVChannelMenu tvMenu;

private FilmMenu filmMenu;

public MainMenu(TVChannelMenu tvMenu, FilmMenu filmMenu)

{

this.tvMenu = tvMenu;

this.filmMenu = filmMenu;

}

public void PrintMenu()

{

*Console*.*WriteLine*("电视节目有：");

PrintMenu(tvMenu);

*Console*.*WriteLine*("电影节目有：");

PrintMenu(filmMenu);

}

private void PrintMenu(TelevisionMenu menu)

{

foreach (MenuItem item in menu)

{

*Console*.*Write*("channel: " + item.Channel + ", ");

*Console*.*Write*("name: " + item.Name + ", ");

*Console*.*WriteLine*("description: " + item.Description);

}

}

}

class Program

{

static void Main(string[] args)

{

TVChannelMenu tvMenu = new TVChannelMenu();

FilmMenu filmMenu = new FilmMenu();

MainMenu mainMenu = new MainMenu(tvMenu, filmMenu);

mainMenu.PrintMenu();

}

}

根据C#的特定语法，最终改进的版本：

取消了自定义枚举器IEnumerator，利用yield关键字来实现编译器自动生成枚举器。

例：C:\study\C-sharpProject\iterator\_test4

public class MenuItem

{

public MenuItem(int channel, string name, string description)

{

Channel = channel;

Name = name;

Description = description;

}

public string Name { get; set; }

public string Description { get; set; }

public int Channel { get; set; }

}

public interface TelevisionMenu : *IEnumerable*

{

void AddItem(int channel, string name, string description);

}

public class TVChannelMenu : TelevisionMenu

{

// 使用List集合

private *List*<MenuItem> menuItems = new *List*<MenuItem>();

public TVChannelMenu()

{

AddItem(1, "CCTV-1", "This is CCTV-1");

AddItem(2, "CCTV-2", "This is CCTV-2");

AddItem(3, "CCTV-3", "This is CCTV-3");

}

public void AddItem(int channel, string name, string description)

{

MenuItem tvMenuItem = new MenuItem(channel, name, description);

menuItems.*Add*(tvMenuItem);

}

public *IEnumerator* GetEnumerator()

{

for (int i = 0; i < menuItems.*Count*; ++i)

{

yield return menuItems[i];

}

}

}

public class FilmMenu : TelevisionMenu

{

const int MAX\_ITEMS = 5;

MenuItem[] menuItems = new MenuItem[MAX\_ITEMS];

int numberOfItems = 0;

public FilmMenu()

{

AddItem(1, "绝世天劫", "这是好电影");

AddItem(2, "达芬奇密码", "这是最喜欢的电影");

}

public void AddItem(int channel, string name, string description)

{

if (numberOfItems < MAX\_ITEMS)

{

MenuItem filmMenuItem = new MenuItem(channel, name, description);

menuItems[numberOfItems] = filmMenuItem;

++numberOfItems;

}

}

public *IEnumerator* GetEnumerator()

{

for (int i = 0; i < numberOfItems; ++i)

{

yield return menuItems[i];

}

}

}

public class MainMenu

{

private TVChannelMenu tvMenu;

private FilmMenu filmMenu;

public MainMenu(TVChannelMenu tvMenu, FilmMenu filmMenu)

{

this.tvMenu = tvMenu;

this.filmMenu = filmMenu;

}

public void PrintMenu()

{

*Console*.*WriteLine*("电视节目有：");

PrintMenu(tvMenu);

*Console*.*WriteLine*("电影节目有：");

PrintMenu(filmMenu);

}

private void PrintMenu(TelevisionMenu menu)

{

foreach (MenuItem item in menu)

{

*Console*.*Write*("channel: " + item.Channel + ", ");

*Console*.*Write*("name: " + item.Name + ", ");

*Console*.*WriteLine*("description: " + item.Description);

}

}

}

class Program

{

static void Main(string[] args)

{

TVChannelMenu tvMenu = new TVChannelMenu();

FilmMenu filmMenu = new FilmMenu();

MainMenu mainMenu = new MainMenu(tvMenu, filmMenu);

mainMenu.PrintMenu();

}

}