Версионность

№ урока: 10 **Курс:** C# Professional

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Очень часто при проектировании класса, специально предназначенного служить базовым в иерархии, виртуальные методы объявляются таким образом, чтобы производные классы могли модифицировать поведение. На уроке рассматриваются возможные проблемы, связанные с неправильной реализацией иерархии, кроме того, описывается применимость шаблона NVI для решения проблемы «неправильных» базовых классов. Отдельное внимание уделяется рассмотрению понятия полиморфизм и его видам.

Изучив материал данного занятия, учащийся сможет:

- Понимать, что такое версионность
- Применять на практике шаблон NVI
- Реализовывать замещение, переопределение, перекрытие методов
- Понимать отличие между классическим и ad hoc полиморфизмом

Содержание урока

1. Шаблон NVI
2. Классический полиморфизм
3. Ad hoc полиморфизм
4. Замещение, переопределение и перекрытие методов

Резюме

- Шаблон NVI (Non-Virtual Interface) объявляет общедоступный интерфейс базового класса не виртуальным, но переопределяемое поведение перемещает в другой, защищенный метод, добавляя необходимый промежуточный уровень.
- Шаблон NVI подобен шаблону Template Method (Шаблонный метод), описанный Эрихом Гаммой, Ричардом Хелмом, Ральфом Джонсоном и Джоном Влиссидесом. Суть Шаблонного метода заключается в том, что он определяет основу алгоритма и позволяет наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом. Особенно эффективным его применение становится, когда нужно:
    - Однократно использовать инвариантную часть алгоритма, а изменение остальной части оставить на усмотрение наследникам
    - Локализировать и вычленить общий код для нескольких классов для избегания дублирования
    - Разрешить расширение кода наследникам только в определенных местах
- Шаблон NVI широко применяется в библиотеках .NET Framework и рекомендуется к использованию практически в каждом руководстве по проектированию от Microsoft.
- Полиморфизм относится к способности определять множество классов функционально разными, но одинаково названными методами или свойствами, которые попеременно могут использоваться кодом клиента во время выполнения. О полиморфизме часто говорят как о третьем базовом элементе объектно-ориентированного программирования, после инкапсуляции и наследования. Полиформизм — это греческое слово, означающее "наличие многих форм". Это понятие имеет два различающихся аспекта:
    - Во время выполнения объекты производного класса могут рассматриваться как объекты базового класса в таких местах как параметры метода и коллекции массивов. При этом объявленный тип объекта больше не идентичен его типу времени выполнения.
    - Базовые классы могут определять и реализовывать виртуальные методы, а производные классы могут переопределять их. Это означает, что они

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

предоставляют свои собственные определение и реализацию. Во время выполнения, когда клиентский код вызывает метод, среда CLR ищет тип времени выполнения объекта и вызывает это переопределение виртуального метода. Таким образом, в исходном коде можно вызвать метод в базовом классе и вызвать выполнение метода с версией производного класса.

- Если производный класс наследует от базового класса, то он приобретает все методы, поля, свойства и события базового класса. Проектировщик производного класса может выбирать из следующих возможностей:
  - переопределить виртуальные члены в базовом классе,
  - наследовать самый близкий метод базового класса без его переопределения,
  - определить новую не виртуальную реализацию этих членов, которая скрывает реализации базового класса.
- Производный класс может переопределить член базового класса, если только член базового класса объявлен как виртуальный или абстрактный. Производный член должен использовать ключевое слово `override`, чтобы явно указать, что метод должен участвовать в виртуальном вызове.
- Функции языка C# позволяют обеспечить и поддерживать обратную совместимость за счет управления версиями между базовыми и производными классами в различных библиотеках. Это означает, например, что если в базовом классе будет создан новый член, имя которого совпадает с именем члена в производном классе, язык C# обработает такую ситуацию, и она не приведет к непредвиденным результатам. Это также означает, что в классе должно быть явно указано, будет ли метод переопределять наследуемый метод, или это новый метод, который будет скрывать наследуемый метод с тем же именем.
- В C# производный класс может включать методы с теми же именами, что и у методов базового класса:
  - Метод базового класса может быть определен как виртуальный.
  - Если перед методом в производном классе не указано ключевое слово `new` или `override`, компилятор выдаст предупреждение, и обработка метода будет производиться как в случае наличия ключевого слова `new`.
  - Если перед методом в производном классе указано ключевое слово `new`, то этот метод определен как независимый от метода в базовом классе.
  - Если перед методом в производном классе указано ключевое слово `override`, то объекты производного класса будут вызывать этот метод вместо метода базового класса.
  - Базовый метод можно вызвать из производного класса с помощью ключевого слова `base`.
  - Ключевые слова `override`, `virtual` и `new` могут также применяться к свойствам, индексам и событиям.
- Существует две основные разновидности полиморфизма: классический полиморфизм и полиморфизм «для конкретного случая» или же ad hoc полиморфизм.
- Ad hoc полиморфизм позволяет обращаться схожим образом к объектам, не связанным классическим наследованием. Достигается это очень просто: в каждом из таких объектов должен быть метод с одинаковой сигнатурой (то есть одинаковым именем метода, принимаемыми параметрами и типом возвращаемого значения).
- В языках, поддерживающих такой тип полиморфизма, применяется технология позднего связывания, когда тип объекта, к которому происходит обращение, становится ясен только в процессе выполнения программы.

Закрепление материала

1. Что такое шаблон NVI?
2. Для чего применяется шаблон Template method (Шаблонный метод)?
3. Что такое полиморфизм?
4. Объясните назначение и применимость ключевых слов `new`, `override` и `virtual` в контексте полиморфизма.
5. Что такое ad hoc полиморфизм?

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

Дополнительное задание

Реализуйте шаблон NVI в собственной иерархии наследования.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные конструкции и понятия, рассмотренные на уроке.

Задание 2
Выучите описание шаблона Template method (Шаблонный метод). Обратите внимание на применимость шаблона, а также на состав его участников и связи отношения между ними. Напишите небольшую программу на языке C#, представляющую собой абстрактную реализацию данного шаблона.

Задание 3
Зайдите на сайт MSDN.
Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на уроке, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. Сохраните ссылки и дайте им короткое описание.

Рекомендуемые ресурсы

MSDN: Полиморфизм
http://msdn.microsoft.com/ru-ru/library/ms173152.aspx

MSDN: Управление версиями при помощи ключевых слов `new` и `override`
http://msdn.microsoft.com/ru-ru/library/6fawty39.aspx

MSDN: Использование ключевых слов `new` и `override`
http://msdn.microsoft.com/ru-ru/library/ms173153.aspx

MSDN: Модификатор `new`
http://msdn.microsoft.com/ru-ru/library/435f1dw2.aspx

MSDN: Модификатор `override`
http://msdn.microsoft.com/ru-ru/library/ebca9ah3.aspx

MSDN: Модификатор `virtual`
http://msdn.microsoft.com/ru-ru/library/9fkccyh4.aspx

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com