Тема 8. Использование переменных ссылочных типов данных

Парадигмы программирования. C#

План

- Основы ссылочных типов данных
- Использование встроенных ссылочных типов данных
- Иерархия классов
- Области имён в .NET Framework
- Приведение ссылочных типов данных

Основы ссылочных типов данных

Часть 1

Сравнение значимых типов со ссылочными типами

- Значимые типы
  - Переменные хранятся в стеке и содержат значения непосредственно внутри себя
  - Примеры: int, short, char
- Ссылочные типы
  - Переменные также хранятся в стеке, но содержат только ссылку на данные
  - Данные хранятся в отдельной области памяти – куче
- Пример: string

Выделение и освобождение памяти ссылочных переменных

- Объявление ссылочного типа данных

```
class Coordinate
{
    public double x = 0;
    public double y = 0;
}
```

- Выделение памяти для переменной

```
Coordinate coordinate;
coordinate = new Coordinate();
coordinate.x = 10.28;
coordinate.y = -8.52;
```

- Освобождение переменной

```
coordinate = null;
```

Неверные ссылки

- При неверной ссылке невозможно получить доступ к данным

```
Coordinate coordinate;
coordinate.x = 10.28; // Переменная не инициализирована
```

- Неверные ссылки обнаруживаются компилятором
- При попытке использования неверных ссылок во время выполнения программы будут возникать исключения типа `NullReferenceException`
- Если не уверены, инициализирована ли ссылочная переменная – проверьте, не равно ли её значение `null`

Сравнение значений и сравнение ссылок

- Для значимых типов
  - Использование == и != приводит к сравнению значений
- Для ссылочных типов
  - Использование == и != приводит к сравнению не значений, а ссылок
  - Даже если данные двух переменных совпадают, но ссылки разные, то их сравнение с помощью == даст `false`

```
Coordinate c1 = new Coordinate();
Coordinate c2 = new Coordinate();
c1.x = 1.0;
c1.y = 2.0;
c2.x = 1.0;
c2.y = 2.0;
if (c1 == c2)
    Console.WriteLine("Same");
else
    Console.WriteLine("Different");
```

- Использование операторов сравнения < > <= >= недопустимо для переменных ссылочных типов

Несколько ссылок на один объект

- Две ссылки могут указывать на один и тот же объект
- Все такие ссылки могут использоваться для чтения и записи

```
Coordinate c1 = new Coordinate();
Coordinate c2;
c1.x = 5.1; c1.y = 2.8;
c2 = c1;
c1.x = 10;
Console.WriteLine("{0}, {1}", c1.x, c1.y);
Console.WriteLine("{0}, {1}", c2.x, c2.y);
```

Использование ссылок в качестве параметров методов

- Ссылки могут использоваться в качестве параметров
  - При передаче по значению сами ссылки копируются в стек, но данные, на которые они ссылаются, фактически передаются по ссылке

```
static void PassCoordinateByValue(Coordinate c)
{
    c.x++; c.y++; // c ссылается на loc
}
…
Coordinate loc = new Coordinate();
loc.x = 5; loc.y = 7;
PassCoordinateByValue(loc);
Console.WriteLine(loc.x + " , " + loc.y);
```

Передача ссылочных переменных по ссылке и в качестве выходных

- Передача по ссылке

```
static void PassCoordinateByRef(ref Coordinate c)
{
    c = new Coordinate();
    c.x = c.y = 12.34;
}
…
Coordinate loc = new Coordinate();
PassCoordinateByRef(ref loc);
Console.WriteLine("{0}, {1}", loc.x, loc.y);
```

- Передача, как выходного параметра

```
static void PassCoordinateByOut(out Coordinate c)
{
    c = new Coordinate();
    c.x = c.y = 56.78;
}
…
Coordinate loc;
PassCoordinateByOut(out loc);
Console.WriteLine("{0}, {1}", loc.x, loc.y);
```

Использование встроенных ссылочных типов данных

Часть 2

Класс Exception

- Exception – это класс (если кто-то забыл ☺)
- Объекты класса Excpetion используются для выбрасывания исключений
  - Новый объект исключения создаётся с помощью оператора `new`
  - Исключение выбрасывается с помощью оператора `throw`
- Все типы исключений являются прямыми или опосредованными наследниками класса Exception

Класс string

- Это строка, состоящая из множества символов в кодировке `Unicode`
- `string` – это сокращённое представление `System.String`
- Сроки в C# являются неизменяемыми

```
string s = "Hello World!";
s[0] = 'c'; // Ошибка компиляции
```

- Для создания изменяемых строк используйте класс `StringBuilder`

Общие методы, операторы и свойства класса String (MSDN)

- Квадратные скобки `[]`
- Метод `Insert`
- Свойство `Length`
- Метод `Copy`
- Метод `Concat`
- Метод `Trim`
- Методы `ToUpper` и `ToLower`

Сравнение строк (MSDN)

- Метод Equals
  - Сравнение по значению на равенство
- Метод Compare
  - Сравнение строк больше – меньше – равно
  - Возможность сравнивать без учёта регистра
  - Сортировка символов при сравнении согласно словарю
- Учёт языковых национальных особенностей при сравнении
  - `System.Globalization namespace`
  - `CultureInfo class`

Операторы сравнения строк

- Операторы == и != переопределены для строк
- Эти операторы эквивалентны `String.Equals` и `!String.Equals`

```
string a = "Test";
string b = "Test";
if (a == b) ... // Истинно
```

Иерархия классов

Часть 3

Тип данных object

- Это синоним для класса `System.Object`
- Этот класс является базовым классом для всех типов данных в C#, даже для значимых типов данных
- К типу данных `object` может быть приведён любой тип данных
- В любом типе данных есть члены, определённые в `object`

Основные методы класса object

- Метод `ToString`
- Метод `Equals`
- Метод `GetType`
- Метод `Finalize`

Рефлексия

- Рефлексия – это механизм получения информации о любом типе данных
- Типы данных рефлексии объединены в область имён System.Reflection
- Оператор typeof возвращает объекта класса Type, хранящий информацию о интересующем классе

```
Type t = typeof(string);
MethodInfo[] mi = t.GetMethods();
foreach (MethodInfo m in mi)
{
    Console.WriteLine("Method: {0}", m);
}
```

- Для получения информации об объекте используйте метод GetType

Области имён в .NET Framework

Часть 4

Область имён System.IO

- Содержит доступ к операциям ввода/вывода файловой системы
  - `File, Directory`
  - `StreamReader, StreamWriter`
  - `FileStream`
  - `BinaryReader, BinaryWriter`
  - `TextReader, TextWriter`

Пример работы с файлами

```
StreamReader reader = new StreamReader("infile.txt");
// Text in from file
StreamWriter writer = new StreamWriter("outfile.txt");
// Text out to file
string line;
while ((line = reader.ReadLine()) != null)
{
    writer.WriteLine(line);
}
reader.Close();
writer.Close();
```

Область имён System.Xml

- Включает поддержку XML
- Предоставляет возможность использовать различные стандарты XML: XQuery, XPath, DOM, XSD, XSL/T
- Область имён System.Xml.Linq содержит классы работы с XML документами с помощью запросов, похожих на запросы SQL

Область имён System.Data

- Классы работы с данными реляционных баз данных
- Основа работы ADO.NET
  - ADO = ActiveX Data Objects
- Основные классы
  - `DataSet`
  - `DataTable`
  - `DataReader`
  - `DataAdapter`
  - `SqlCommand`
  - `SqlConnection`
- Область имён `System.Data.Linq` содержит классы, позволяющие на C# писать строго типизированные запросы, похожие на запросы `SQL`

Другие полезные области имён

- `System`
- `System.Net`
- `System.Net.Sockets`
- `System.Windows.Forms`
- `System.Web`

Приведение ссылочных типов данных

Часть 5

Преобразование переменных значимых типов

- Неявные преобразования
- Явные преобразования
  - Оператор приведения типа
  - Возможно возникновение исключений
- Класс `System.Convert`
  - Выполняет преобразования внутри себя

Преобразования предок/потомок

- Преобразование к ссылке на базовый класс
  - Неявные или явные
  - Всегда успешные
  - Всегда можно привести к типу `object`
- Преобразование к ссылке на класс-потомок
  - Обязательно наличие явного приведения
  - Будет осуществлена проверка правильности ссылки
  - Если проверка не пройдёт, будет выдано исключение `InvalidCastException`

Оператор is

- Возвращает `true`, если приведение типа допустимо

```
Camel c;
if (a is Camel)
    c = (Camel) a;
else
    Console.WriteLine("Not a Camel");
```

Оператор as

- Осуществляет преобразование между ссылочными типами, как оператор приведения типа
- При невозможности приведения
  - Возвращает `null`
  - Не выбрасывает исключение

```
Camel c = a as Camel;
if (c == null)
    Console.WriteLine("Not a Camel");
```

Преобразования и тип object

- Тип object является базовым для всех типов
- Любая ссылка может быть приведена к object

```
object ox;
ox = a;
ox = (object) a;
ox = a as object;
```

- Любая переменная object может быть приведена к любой ссылке
  - С явным преобразованием типа и проверками

```
b = (Camel) ox;
b = ox as Camel;
```

- Тип object и оператор is

```
if (a is object) // Всегда возвращает true
```

Преобразования типов и интерфейсы

- Интерфейс может использоваться только для доступа к его собственным членам
- Другие методы и переменные классов не доступны через интерфейс

```
interface IShape
{
    void Paint();
}

class Rectangle : IShape
{
    public void Paint() { }

    public void Reset() { }
}
Rectangle r = new Rectangle();
r.Reset();
r.Paint();
IShape s = (IShape)r;
s.Reset(); // Ошибка
s.Paint();
```

Упаковка и распаковка

- Унифицированная система типов
- Упаковка
- Распаковка

```
int p = 123;
object box;
box = p; // Неявная упаковка
box = (object) p; // Явная упаковка
```

- Вызов методов объектов для переменных значимых типов

```
object o = (object) 42; // Упаковка
Console.WriteLine(o.ToString( ));
```

Заключение

- Спасибо за внимание!