Общая система типов CTS

Лекция №3

CTS (общая система типов)

CTS (общая система типов) представляет собой формальную спецификацию, в которой описано то, как должны быть определены типы для того, чтобы они могли обслуживаться в CLR-среде.

Тип Объект

Все объекты косвенно происходят от единого базового класса, определенного в составе CTS. Этот базовый класс — System.Object.

Значимые и ссылочные типы

Концепция создания языка, где любая сущность является объектом, не нова. Если попытаться сложить два значения типа double, при этом реально выделять объекты в куче, то выделение памяти будет чрезвычайно неэффективно.

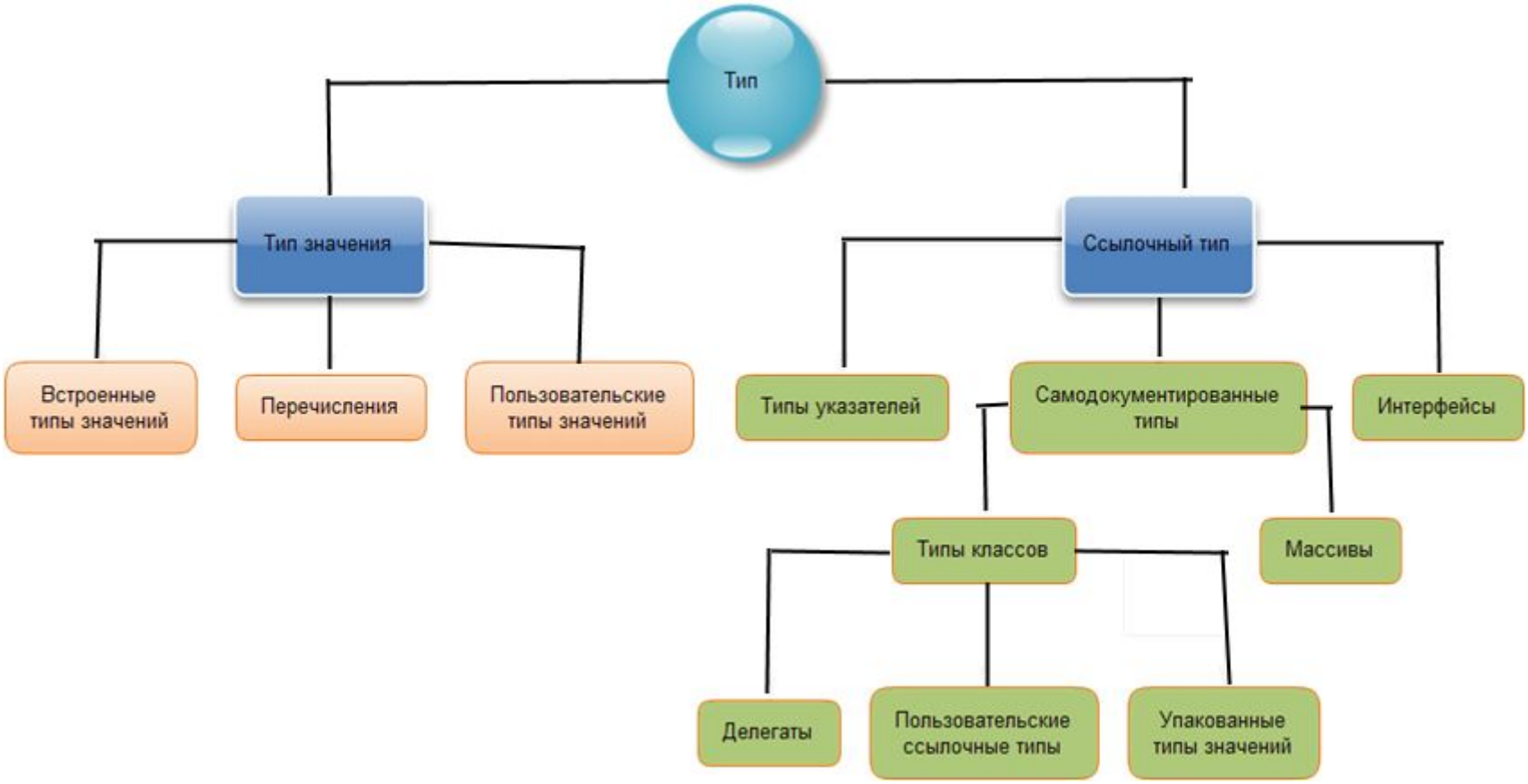
Значимые типы

Для значимого типа используется прямая адресация, значение хранит собственно данные, и память для них отводится, как правило, в стеке.

Ссылочные типы

Для ссылочного типа значение задает ссылку на область памяти в "куче", где расположен соответствующий объект.

Иерархия типов



Встроенные типы

Логический тип			
Имя типа	**Системный тип**	**Значения**	**Размер**
bool	System.Boolean	true, false	8 бит

Пример:

```
bool IsTrue = false;
```

Встроенные типы

Арифметические целочисленные типы			
Имя типа	**Системный тип**	**Диапазон**	**Размер**
sbyte	System.SByte	-128 — 127	Знаковое, 8 Бит
byte	System.Byte	0 — 255	Беззнаковое, 8 Бит
short	System.Short	-32768 —32767	Знаковое, 16 Бит
ushort	System.UShort	0 — 65535	Беззнаковое, 16 Бит
int	System.Int32	(-2*10^9 — 2*10^9)	Знаковое, 32 Бит
uint	System.UInt32	(0 — 4*10^9)	Беззнаковое, 32 Бит
long	System.Int64	(-9*10^18 — 9*10^18)	Знаковое, 64 Бит
ulong	System.UInt64	(0— 18*10^18)	Беззнаковое, 64 Бит

Пример:

```
int a = 123;
long b,c,d;
```

Встроенные типы

Арифметический тип с плавающей точкой			
Имя типа	**Системный тип**	**Диапазон**	**Точность**
float	System.Single	+1.5*10^-45 -/+3.4*10^38	7 цифр
double	System.Double	+5.0*10^-324 -/+1.7*10^308	15-16 цифр
Арифметический тип с фиксированной точкой			
Имя типа	**Системный тип**	**Диапазон**	**Точность**
decimal	System.Decimal	+1.0*10^-28 - +7.9*10^28	28-29 значащих цифр

Встроенные типы

Символьные типы			
Имя типа	**Системный тип**	**Диапазон**	**Точность**
char	System.Char	U+0000 - U+ffff	16 бит Unicode символ
string	System.String	Строка из символов Unicode	
Объектный тип			
Имя типа	**Системный тип**	**Примечание**	
object	System.Object	Прародитель всех *встроенных* и пользовательских типов	

Пример:

```
object a = 123;
```

Упаковка и распаковка

Как же эти различные категории типов обеспечивают более эффективную работу системы?

Это делается с помощью *упаковки* (boxing). В простейшем случае при упаковке размерный тип преобразуется в ссылочный. В обратном случае ссылочный тип *распаковывается* (unbox) в размерный.

Упаковка и распаковка

Объект лишь тогда является объектом, когда это необходимо.

- int foo = 42; // Размерный тип.
- object bar = foo; // Переменная foo упакована в bar.

А теперь выполним явное приведение типов, чтобы преобразовать *bar* обратно в размерный тип:

- int foo = 42; // Размерный тип.
- object bar = foo; // Переменная foo упакована в bar.
- int foo2 = **(int)** bar; // Распаковка и приведение к типу int.

Тип Объект: открытые методы

Метод	Описание
bool Equals()	Сравнивает две ссылки на объекты в период выполнения, чтобы определить, указывают ли они в точности один и тот же объект. Если две переменные ссылаются на один и тот же объект, возвращается *true*. В случае размерных этот метод возвращает *true,* если типы переменных идентичны и их значения равны.
Type GetType()	Используется с методами отражения для получения информации о типе данного объекта.

Тип Объект: открытые методы

Метод	Описание
int GetHashCodeO	Возвращает заданный для объекта хэш-код. Хэш-функции используются в реализации класса, когда хэш-код объекта нужно поместить в хэш-таблицу для повышения производительности.
string ToString()	Используется по умолчанию для получения имени объекта. Его можно переопределить в производных классах, чтобы они возвращали понятное пользователю текстовое представление объекта.

Тип Объект: защищенные методы

Метод	Описание
void Finalize()	Вызывается в период выполнения для освобождение ресурсов перед сбором мусора. Этот метод можно вызывать, а можно и не делать этого. Поэтому не помещайте в него подлежащий исполнению код..

Тип Объект: защищенные методы

Метод	Описание
Object MemberwiseClone	Представляет *ограниченную копию* (shallow copy) объекта. Под этим понимают копию объекта, содержащую ссылки на другие объекты, но не копии этих объектов. Если ваши классы должны поддерживать *полную копию* (deep copy), которая действительно включает копии объектов, на которые она ссылается, то вам нужно реализовать интерфейс *ICloneable* и самому вручную производить клонирование или копирование.

Приведение типов

Приведение типов – это преобразование значения одного типа в значение другого типа.

Приведение типов

Выделяют приведения типов:

- явные (англ. *explicit*);

```
int num = 2147483647;
long bigNum = num;
```

- неявные (англ. *implicit*).

```
double x = 1234.7;
int a;
a = (int)x; // a = 1234
```

Приведение типов

```
class Employee { }
class ContractEmployee : Employee { }
class CastExampleI
{
public static void Main () {
Employee e = new ContractEmployee(); } }
```

Этот код будет работать, поскольку всегда подразумевается восходящее приведение (upcast) производного класса к его базовому классу.

Приведение типов

А вот такой код недопустим, так как компилятор не предоставляет неявное нисходящее приведение (downcast).

```
…
class CastExample2 {
public static void Main ()
{
ContractEmployee ce = new Employee (); // Не будет
// компилироваться.
} }
```

Приведение типов

Вы не сможете выполнить нисходящее приведение объекта типа Employee к объекту типа ContractEmployee, поскольку нет гарантии, что этот объект поддерживает интерфейс, определенный классом ContractEmployee. Поэтому в случае нисходящего приведения используется явное приведение:

```
...
class CastExampleS {
public static void Main ()
{// Нисходящее приведение не сработает.
ContractEmployee ce = (ContractEmployee)new Employee(); }}
```

Приведение типов

А давайте обманем CTS путем явного приведения базового класса к производному:

…

```
class CastExample4 {
public static void Main ()
 { Employee e = new Employee ();
ContractEmployee c = (ContractEmployee)e; } }
```

Эта программа компилируется, но генерирует исключение периода выполнения.