ЛЕКЦИЯ 10

- Модуль 2
- 29.11.2023
- Упаковка и распаковка
- Nullable-типы

ЦЕЛИ ЛЕКЦИИ

- Получить представления о типе данных структура (struct)
- Получить представление о перечислениях (enum)

УПАКОВКА И РАСПАКОВКА

Boxing and Unboxing

УПАКОВКА И РАСПАКОВКА

Упаковка – процесс преобразования типа значения в объект с типом object или тип интерфейса им реализуемого

Операция присваивания:
```
L-value = R-value
```

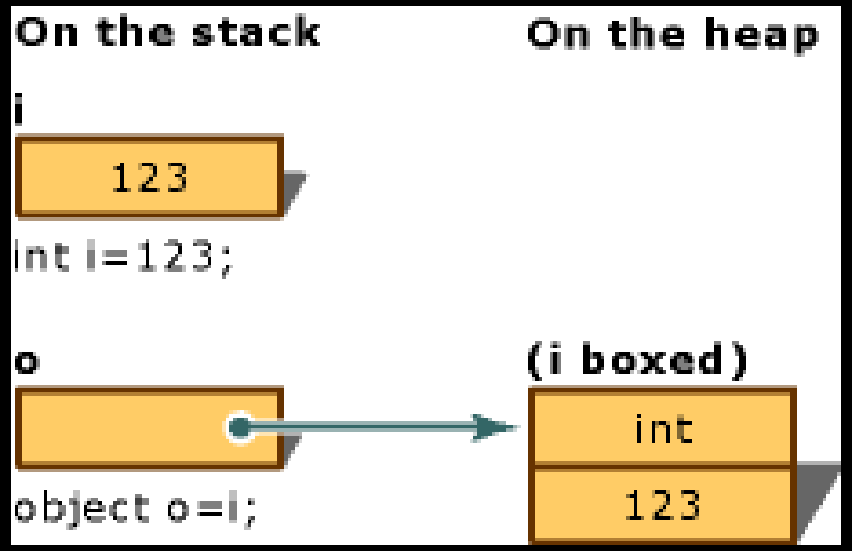
Явное приведение типов при распаковке:
```
L-value = (тип_L_value) R-value
```

L-value	R-value	Действие
Тип значения	Тип значения	Нет упаковки/распаковки
Тип ссылки	Тип ссылки	Нет упаковки/распаковки
Тип ссылки	Тип значения	Упаковка
Тип значения	Тип ссылки	Распаковка

УПАКОВКА (BOXING)

- Тип значений в тип Object
- Должен быть создан новый объект и размещен в куче
- Дорогая операция (в вычислительном смысле)

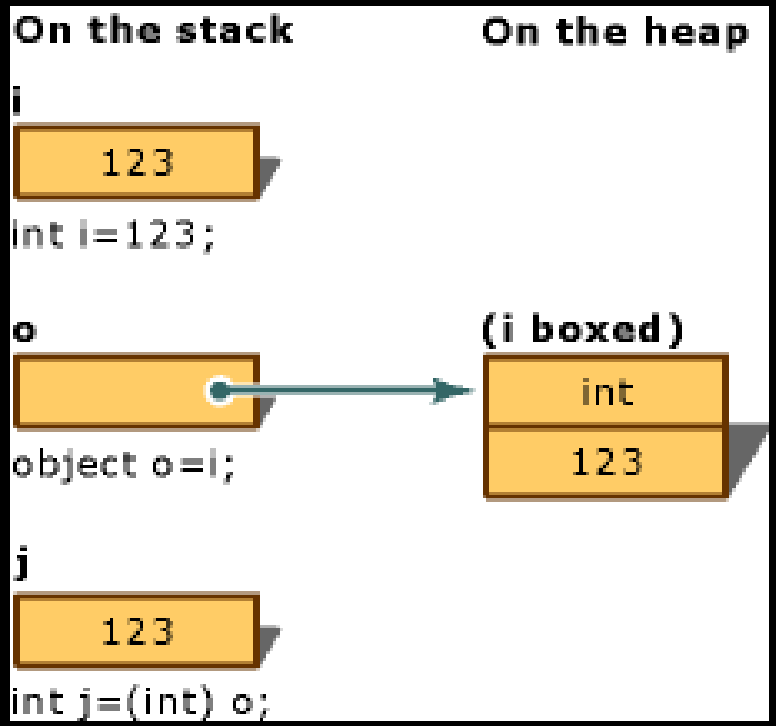
```
int i = 123;
object o = i; // неявная упаковка
```



РАСПАКОВКА (UNBOXING)

- Тип Object в тип значений
- Дорогая операция (в вычислительном смысле)

```
int i = 123;
object o = i; // неявная упаковка
int j = (int)o; // распаковка
```



ПОТЕРИ В ПРОИЗВОДИТЕЛЬНОСТИ ПРИ УПАКОВКЕ

```
const int count = 100_000_000;

Console.WriteLine("С упаковкой: {0} мс",          // 1072
DoTest(count, () => {
    int value = 123;
    object objValue = value;
}));

Console.WriteLine("Без упаковки: {0} мс",         // 520
DoTest(count, () => {
    int value = 123;
    int value2 = value;
}));

static long DoTest(int count, Action action) {
    var sw = Stopwatch.StartNew();
    for (int i = 0; i < count; i++) action();
    return sw.ElapsedMilliseconds;
}
```

ТИПЫ ЗНАЧЕНИЙ, ДОПУСКАЮЩИЕ ЗНАЧЕНИЕ NULL

ТИПЫ ЗНАЧЕНИЙ, ДОПУСКАЮЩИЕ NULL (NULLABLE<T>)

Тип?

- **Тип значения, допускающий null** предоставляет все значения типа и вдобавок значение `null`
- Значение типа значения, допускающего null имеет тип структуры `System.Nullable<T>`

Nullable-тип может потребоваться при работе с источниками данных, в которых присутствуют пропущенные и неопределённые значения

```
double? pi = 3.14;
char? letter = 'a';
int m2 = 10;
int? m = m2;
bool? flag = null;
int?[] arr = new int?[10];

int? b = 10;
if (b.HasValue)
    Console.WriteLine($"b is {b.Value}");   // b is 10
else
    Console.WriteLine("b does not have a value");
```

От какого типа унаследован Nullable<T>?

ОБЪЯВЛЕНИЯ ПЕРЕМЕННЫХ

```
int? x;

string? str;
```

Переменные nullable-типа

Ссылка на массив nullable-типа

```
double?[] dbls;
```

НАЗНАЧЕНИЕ ЗНАЧЕНИЙ

```
int? x = null;

string? str = "Test String";
```

```
int y = null;
```

Недопустимое типом значение

```
string[] strings = { "3,44", "", "0" };

double?[] dbls = new double?[strings.Length];
for (int i = 0; i < strings.Length; i++)
{
    try
    {
        dbls[i] = double.Parse(strings[i]);
    }
    catch (FormatException)
    {
        dbls[i] = null;
        continue;
    }
}
```

```
for(int i = 0; i < dbls.Length; i++)
{
    Console.WriteLine(dbls[i]);
}
```

ПРОВЕРКИ НАЛИЧИЯ ЗНАЧЕНИЯ

```
double?[] dbls = new double?[strings.Length];
for (int i = 0; i < strings.Length; i++)
{
    try
    {
        dbls[i] = double.Parse(strings[i]);
    }
    catch (FormatException)
    {
        dbls[i] = null;
        continue;
    }
}
```

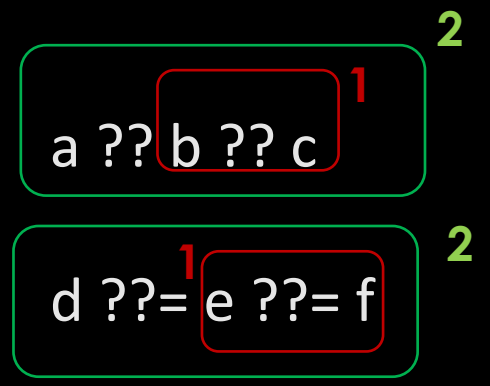
Проверяем наличие у экземпляра типа double? значения типа double

```
for(int i = 0; i < dbls.Length; i++)
 {
     Console.WriteLine(dbls[i].HasValue? dbls[i]:"N/A");
 }
```

```
Console.WriteLine(dbls[i].HasValue? dbls[i].Value:"N/A");
```

ОПЕРАЦИЯ ОБЪЕДИНЕНИЯ С NULL

- **op1 ?? op2** операция (оператор) объединения с NULL
  - Возвращает op1, если op1 != null и op2 в противном случае
  - Если op1 != null к op2 не оценивается
- **op1 ??= op2** операция (оператор) присваивания объединения значений с NULL
  - Если op1 == null выполняется присваивание в op1 значения op2
  - Если op1 != null к op2 не оценивается
- Обе операции правоассоциативны



ОПЕРАЦИЯ ОБЪЕДИНЕНИЯ С NULL: ПРИМЕР

```
int? a = 28;
int b = a ?? -1;
Console.WriteLine($"b is {b}");  // b is 28

int? c = null;
int d = c ?? -1;
Console.WriteLine($"d is {d}");  // d is -1

int? n = null;

//int m1 = n;     // не скомпилируется
int n2 = (int)n; // скомпилируется, но выбросит исключение
```

ОПЕРАЦИЯ ОБЪЕДИНЕНИЯ С NULL: ПРИМЕРЫ

```
string[] strings = { "3,44", "", "0" };

double?[] dbls = new double?[strings.Length];
for (int i = 0; i < strings.Length; i++)
{
    try
    {
        dbls[i] = double.Parse(strings[i]);
    }
    catch (FormatException)
    {
        dbls[i] = null;
        continue;
    }
}
```

Метод извлекает значение из nullable-типа, даже если **HasValue** – **false**, в таком случае вернётся значение по умолчанию базового типа (в примере – это **double**)

```
for(int i = 0; i < dbls.Length; i++)
{
    double buff = dbls[i].GetValueOrDefault();
    Console.WriteLine(buff);
}
```

УПАКОВКА И РАСПАКОВКА NULLABLE<T>

Механизм упаковки экземпляра типа значения, допускающего значение `null`:

- Если `HasValue` возвращает `false`, создается пустая ссылка
- Если `HasValue` возвращает `true`, упаковывается соответствующее значение базового типа T, а не экземпляр `Nullable<T>`

Механизм распаковки:

- При распаковке значения null, CLR создаст новый экземпляр типа, допускающего значение `null` и назначит полю `HasValue` значение `false`

УПАКОВКА И РАСПАКОВКА NULLABLE<T>

```
int a = 41;
object aBoxed = a;
int? aNullable = (int?)aBoxed;
Console.WriteLine($"aNullable: {aNullable}"); // 41

object aNullableBoxed = aNullable;
if (aNullableBoxed is int valueOfA)
{
    Console.WriteLine($"aNullableBoxed is boxed int: {valueOfA}"); // 41
}
```

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

- https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/types/boxing-and-unboxing
- https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/nullable-value-types
- https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/nullable-value-types
- Подводные камни при работе с enum в C# (https://habr.com/ru/companies/pvs-studio/articles/568928/)