Introducción a C#

1

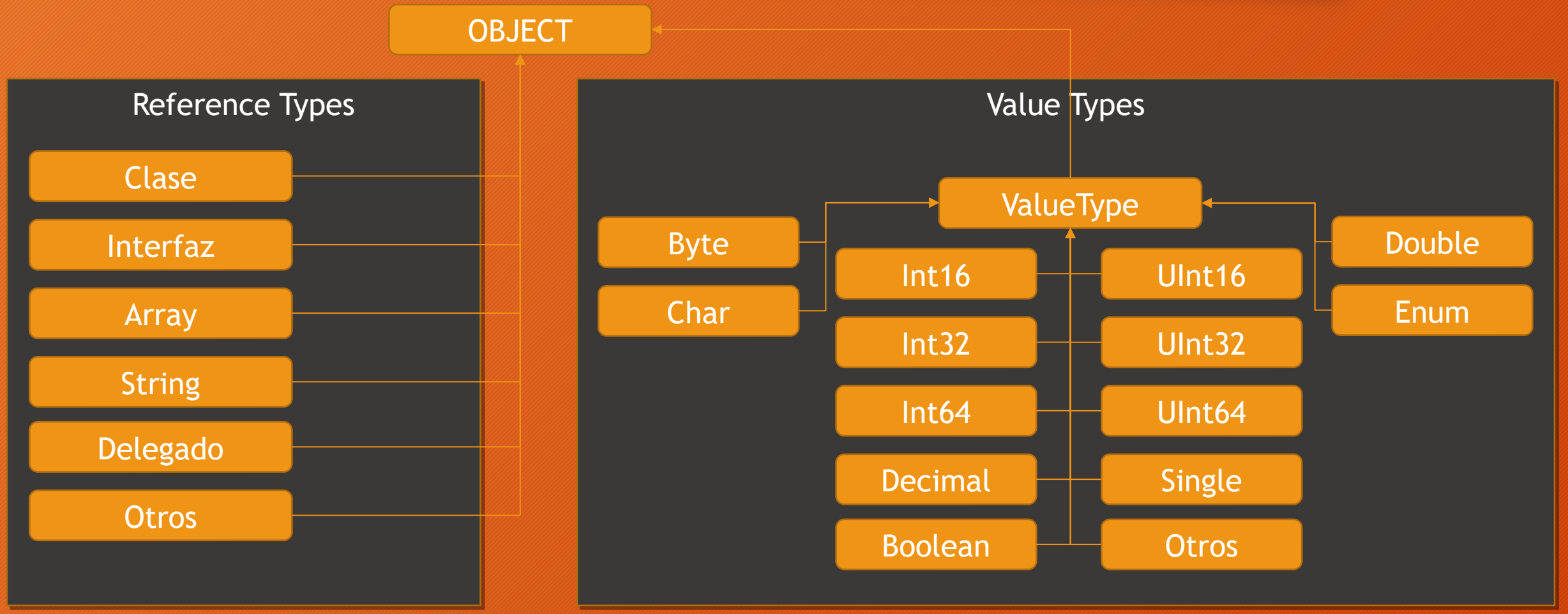
Programación II y Laboratorio de Computación II

Edición 2018

Common Type System (CTS)

- Define un conjunto común de “tipos” de datos orientados a objetos.
- Todo lenguaje de programación .NET debe implementar los tipos definidos por el CTS.
- Todo tipo hereda directa o indirectamente del tipo System.Object.
- El CTS define tipos de VALOR y de REFERENCIA

Common Type System (CTS)



Categoría	Clase	Descripción	C# Alias
Enteros	Byte	Un entero sin signo (8-bit)	byte
	SByte	Un entero con signo (8-bit)	sbyte
	Int16	Un entero con signo (16-bit)	short
	Int32	Un entero con signo (32-bit)	int
	Int64	Un entero con signo (64-bit)	long
Punto Flotante	Single	Un número de punto flotante de simple precisión (32-bit)	float
	Double	Un número de punto flotante de doble precisión (64-bit)	double
	Decimal	Un número decimal de 96-bit	decimal
Lógicos	Boolean	Un valor booleano (true o false)	bool
Otros	Char	Un caracter Unicode (16-bit)	char

Tipos de Datos

- Las variables escalares son constantes o variable que contiene un dato atómico y unidimensional.

- Las variables no escalares son array (vector), lista y objeto, que pueden tener almacenado en su estructura más de un valor.

Valores Predeterminados (atributos de clase)

- Enteros
  - 0 (cero)
- Punto flotante
  - 0 (cero)
- Lógicos
  - False
- Referencias
  - Null

Conversiones Básicas

- Implícitas: no interviene el programador

```
float flotante = 15;
```

- Explícitas: interviene el programador, ya que puede haber perdida de datos.

```
int entero = (int)15.2;
```

Conversiones Implícitas

Tipo	Conversiones permitidas
sbyte	short, int, long, float, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long	float, double, decimal
ulong	float, double, decimal
float	double
double	Ninguna
decimal	Ninguna

Operadores Aritméticos

Descripción	C#
Asignación	=
Adición	+
Sustracción	-
Multiplicación	*
División	/
Negación	!
Módulo (Parte entera de la división)	%
Mayor	>
Menor	<
Mayor o Igual	>=
Menor o Igual	<=

Operadores Lógicos

Descripción	C#
Operador lógico Y	&&
Operador lógico O	||
Negación lógica	!
Igualdad	==
Desigualdad	!=

En C# todas las evaluaciones se hacen por “cortocircuito”

```
//Si Hacer1() es True,
//entonces NO se evalua Hacer2()
if (Hacer1() || Hacer2())
{ }
```

```
//Si Hacer1() es False,
//entonces NO se evalua Hacer2()
if (Hacer1() && Hacer2())
{ }
```

Sentencias Condicionales

```
if (x > 10)
    Hacer1();
```

```
if (x > 10)
{
    Hacer1();
    Hacer2();
}
```

```
if (x > 10)
{
    Hacer1();
}
else
{
    Hacer2();
}
```

```
if (x > 10)
{
    Hacer1();
}
else if (x <= 20)
{
    Hacer2();
}
else
{
    Hacer3();
}
```

Sentencias Condicionales

```
int a = 0;

switch (a)
{
    case 1:
        // Código 1
        break;
    case 2:
        // Código 2
        break;
    default:
        // Código DEFAULT
        break;
}
```

Sentencias Repetitivas

```
// Partes: declaración, prueba, acción
for (int i = 1; i < 10; i++)
{
}
```

La sentencia **foreach** permite recorrer arreglos y colecciones

```
string[] nombres = new string[5];

foreach (string auxNombre in nombres)
{
    //auxNombre es un elemento de nombres.
}
```

Sentencias Repetitivas

```
bool condicion = true;

while (condicion == true)
{
    //En algún momento poner condicion = false
}
```

```
bool condicion = true;

do
{
    //En algún momento poner condicion = false
} while (condicion == true);
```

Entry Point

```
class HolaMundo
{
    static void Main(string[] args)
    {
        System.Console.WriteLine("Hola mundo C#");
        // nombre completamente cualificado.

        System.Console.ReadKey();
    }
}
```

Entry Point

El punto de entrada para los programas en C# es la función Main

- static: Es un modificador que permite ejecutar un método sin tener que instanciar a una variable (sin crear un objeto). El método Main() debe ser estático.
- void: Indica el tipo de valor de retorno del método Main(). No necesariamente tiene que ser void.
- string [] args: Es un Array de tipo string que puede recibir el método Main() como parámetro. Este parámetro es opcional.

Console

- Es una clase pública y estática.
- Representa la entrada, salida y errores de Streams para aplicaciones de consola.
- Es miembro del NameSpace **System**.

Métodos

- Clear()
  - Limpia el buffer de la consola. Equivalente a **clrscr()** de C.
- Read()
  - Lee el próximo carácter del stream de entrada. Devuelve un entero.
- ReadKey(bool)
  - Obtiene el carácter presionado por el usuario. La tecla presionada puede mostrarse en la consola. Equivalente a **getch()** / **getche()** de C.

Métodos

- ReadLine()
  - Lee la siguiente línea de caracteres de la consola. Devuelve un string. Equivalente a **gets()** de C.
- Write()
  - Escribe el string que se le pasa como parámetro a la salida estándar. Equivalente a **printf()** de C.
- WriteLine()
  - Ídem método Write, pero introduce un salto de línea al final de la cadena.

Propiedades

- BackGroundColor: Obtiene o establece el color de fondo de la consola.
- ForeGroundColor: Obtiene o establece el color del texto de la consola.
- Title: Obtiene o establece el título de la consola.

Formato de salida de Texto

Con los marcadores (“{}”), además de indicar el número de parámetro que se usará, podemos indicar la forma en que se mostrará.

Cuantos caracteres se mostrarán y si se formatearán a la derecha o la izquierda o también se pueden indicar otros valores de formato.

```
string miTexto = "Hola!";


Console.Write("Acá iría mi texto: {0}", miTexto);
```

Formato de salida de Texto

Fomato completo: { N [, M][: Formato] } (*)

- **N** será el número del parámetro, empezando por cero.
- **M** será el ancho usado para mostrar el parámetro, el cual se rellenará con espacios. Si M es negativo, se justificará a la izquierda, y si es positivo, se justificará a la derecha.
- **Formato** será una cadena que indicará un formato extra a usar con ese parámetro.

Formato de salida de Texto

```
Console.WriteLine("{0,10}{1,-10}{2}", 10, 15, 23);
// Salida:
//         10 15        23

Console.WriteLine("{ 0,10:#,###.00}{1,10}", 10.476, 15.355);
// Salida:
//        10.48         15.355
```