11

Type Conversion

Understanding 'Type Conversion'

- 'Type Conversion' is a process of convert a value from one type (source type) to another type (destination type).
- Ex: int → long

Type Conversion

1. Implicit Casting (from lower-numerical-type to higher-numerical-type)
2. Explicit Casting (from higher-numerical-type to lower-numerical-type)
3. Parse / TryParse (from string to numerical-type)
4. Conversion Methods (from any-primitive-type to any-primitive-type)

Implicit Casting

The 'lower-numerical type' can be automatically (implicitly) converted into 'higher-numerical type'.

Conversion From		Conversion To
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, or decimal
float	→	double
double	→	[none]
decimal	→	[none]
char	→	ushort, int, uint, long, ulong, float, double, decimal

Explicit Casting

- We can manually convert a value from one data type to another data type, by specifying the destination data type within brackets, at left-hand-side of the source value.
- **Loosy conversion:** If the destination type is not sufficient-enough to store the converted value, the value may loose.
- Syntax: (DestinationDataType)SourceValue

When to use explicit casting?

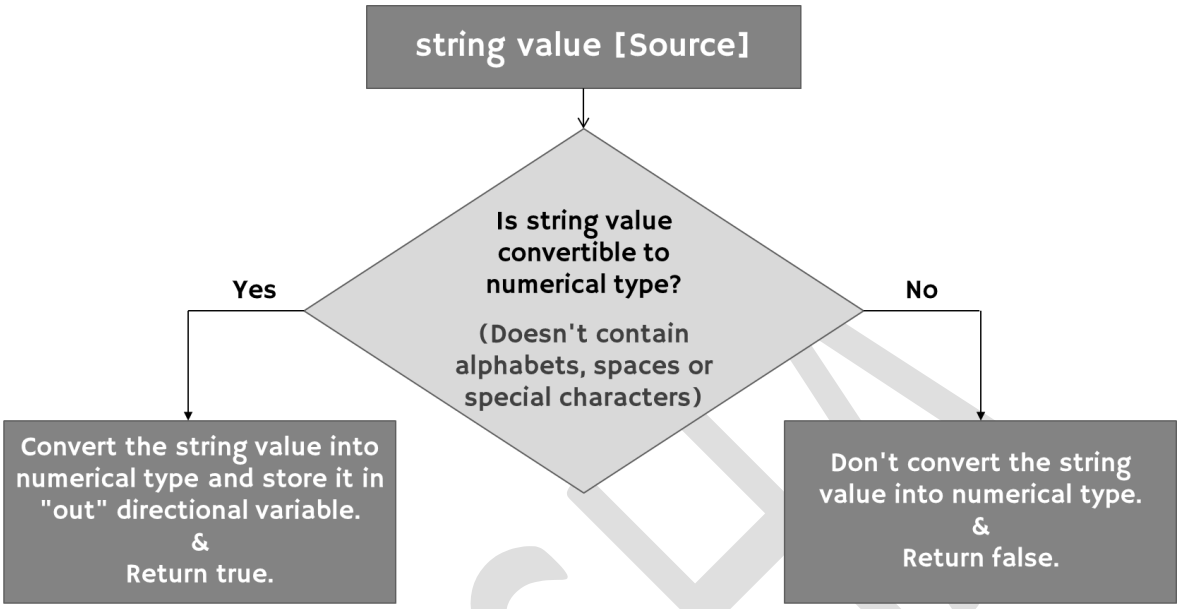
1. At all cases in the table of implicit casting.
2. At the case in the following table of explicit casting.
3. Child class to Parent class.

Conversion From		Conversion To
sbyte	→	byte, ushort, uint, ulong
byte	→	sbyte
short	→	sbyte, byte, ushort, uint, ulong
ushort	→	sbyte, byte, short
int	→	sbyte, byte, short, ushort, uint, ulong
uint	→	sbyte, byte, short, ushort, int
long	→	sbyte, byte, short, ushort, int, uint, ulong
ulong	→	sbyte, byte, short, ushort, int, uint, long
float	→	sbyte, byte, short, ushort, int, uint, long, ulong, decimal
double	→	sbyte, byte, short, ushort, int, uint, long, ulong, float, decimal
decimal	→	sbyte, byte, short, ushort, int, uint, long, ulong, float, double
char	→	sbyte, byte, short, ushort, int, uint, long, ulong, float, double, decimal
bool	→	[none]
string	→	[none]

Parse

- The string value can be converted into any numerical data type, by using "Parsing" technique.
- Ex: string → int
- The source value must contain digits only; shouldn't contain spaces, alphabets or special characters.
- If the source value is invalid, it raises FormatException.
- Syntax: DestinationDataType.Parse(SourceValue)

TryParse



- The string value can be converted into any numerical data type, by using "TryParse" technique (same as "parse"); but it checks the source value before attempting to parse.
- Ex: string → int
- If the source value is invalid, it returns false; It doesn't raise any exception in this case.
- If the source value is valid, it returns true [indicates conversion is successful]
- It avoids FormatException.

Syntax:

bool variable = *DestinationType*.TryParse(*SourceValue*, out *DestinationVariable*)

Conversion Methods

- Conversion method is a pre-defined method, which converts any primitive type (and also 'string') to any other primitive type (and also 'string').
- Ex: string → int and int → string
- The System.Convert is a class, which contains a set of pre-defined methods.
- It raises FormatException, if the source value is invalid.

- For each data type, we have a conversion method.
- All conversion methods are static methods.

Syntax:

- type destinationVariable = Convert.*ConversionMethod* (*SourceValue*)

Conversion To	Conversion Method
sbyte	System.Convert.ToSByte(value)
byte	System.Convert.ToByte(value)
short	System.Convert.ToInt16(value)
ushort	System.Convert.ToUInt16(value)
int	System.Convert.ToInt32(value)
uint	System.Convert.ToUInt32(value)
long	System.Convert.ToInt64(value)
ulong	System.Convert.ToUInt64(value)
float	System.Convert.ToSingle(value)
double	System.Convert.ToDouble(value)
decimal	System.Convert.ToDecimal(value)
char	System.Convert.ToChar(value)
bool	System.Convert.ToBoolean(value)
DateTime	System.Convert.ToDateTime(value)

Points to Remember

- For all the possible cases of 'implicit casting' and 'explicit casting', it is preferred to use 'explicit casting' or 'conversion methods' always.
- For conversion from 'string' to 'numerical type', use TryParse, instead of 'Parse'; as 'TryParse' avoids exceptions.
- For conversion of value from any-type to any-type, use conversion method.