NULLABLE TYPES

Value Types (structures, enumerations)

- Value Types are by default non-nullable types.
- Non-nullable types doesn't support 'null' values to be assigned to its variables.

Reference Types (classes, interfaces)

- Reference Types are by default nullable types.
- Nullable types support 'null' values assigned to its variables.
- They don't require the following syntax.

Cnverting Value-Types to Nullable-Types

```
Nullable<int> x = null;
[or]
int? x = null;
```

Accessing value of a nullable type:

```
variable.Value
```

```
variable.HasValue
```

NULLABLE TYPES

What is null?

- Represents 'blank' value.
- Eg: In Employee class, the 'int CreditCardNumber' can be 'null'.

Null coalescing operator

- The 'null coalescing operator' checks whether the value is null or not.
- It returns the left-hand-side operand if the value is not null.
- It returns the right-hand-side operand if the value is null.

Advantage

Simplifying the syntax of 'if statement' to check if the value is null.

Syntax:

variableName ?? valueIfNull

NULLABLE TYPES

Null Propagation Operator

- The "Null Propagation Operator (?.) and (? []) checks the value of left-hand operand whether it is null or not.
- It returns the right-hand-side operand (property or method), if the value is not null.
- It returns null, if the value is null.
- It accesses the property or method, only if the reference variable is "not null"; just returns "null", if the reference variable is "null".

Syntax

```
referenceVariable?.fieldName;
```

-- is same as --

```
(referenceVariable == null)? null :
referenceVariable.fieldName;
```

Advantage:

We can invoke desired member (property or method) after checking if null.