As you know, C# data types have a fixed range and are represented as a type in

the System namespace.

For example, the System.Boolean data type can be assigned a value from the set

{true, false}. Now, recall that all of the numerical data types (as well as the Boolean data type) are

*value types*. Value types can never be assigned the value of null, as that is used to establish an empty

object reference:

static void Main(string[] args)

{

// Compiler errors!

// Value types cannot be set to null!

bool myBool = null;

int myInt = null;

// OK! Strings are reference types.

string myString = null;

}

C# supports the concept of *nullable data types*. Simply put, a nullable type can represent all the

values of its underlying type, plus the value null.

Thus, if we declare a nullable bool, it could be assigned a value from the set **{true, false, null}.**

This can be extremely helpful when working **with relational databases**, given that it is quite common to encounter undefined columns in database tables. Without the concept of a nullable data type, there is no convenient manner in C# to represent a numerical data point with no value.

To define a nullable variable type, the question mark symbol (?) is suffixed to the underlying data

type.

Do note that this syntax is only legal when applied **to value types**. If you attempt to create a

nullable reference type (including strings), you are issued a compile-time error. Like a nonnullable

variable, local nullable variables must be assigned an initial value before you can use them:

static void LocalNullableVariables()

{

// Define some local nullable variables.

int? nullableInt = 10;

double? nullableDouble = 3.14;

bool? nullableBool = null;

char? nullableChar = 'a';

int?[] arrayOfNullableInts = new int?[10];

// Error! Strings are reference types!

// string? s = "oops";

}

In C#, the ? suffix notation is a shorthand for creating an instance