**1. Nullable value types (nullfähige Werttypen)?**

**Was sind noch einmal Werttypen und Verweistypen?**

Eine Variable eines Werttypes enthält eine Instanz des Typs. Dies unterscheidet sich von einer Variablen eines Verweistyps, die einen Verweis auf eine Instanz ihres Types enthält.

**Was sind Nullable Types (nullfähige Werttypen)?**

**Definition**

* **Value Types (Werttypen)** wie **int**, **double**, **bool, char,** können **keine null** annehmen. Man kann in diesen kein NULL speichern.  
  Mit **int?** , **double?**, **float?,** **decimal?** oder **bool?** , usw. ist es dies möglich.  
  Sie bleiben weiterhin Value Types (Werttypen). Genauer gesagt sind sie **nullfähige Werttypen** oder nullable value types.

Beispiel **int?**  
Wie speichert .NET den **int?** ?  
**int?** und alle anderen nullfähige Wertetypen speichern ihren eigentlichen Wert im Feld **Value**.  
Im boolschen Feld **HasValue**, wird anzeigt, ob die Variable überhaupt einen Wert (Value) hat, der nicht null ist.

**Info am Rande: Reference Types (Verweistypen)** (**string**, **object**, etc.) sind **von Haus aus** nullable (**nullfähig**), sprich: sie können automatisch (ohne die Setzung eines Fragezeichens) auch **null annehmen**.

**Beispiel**

int? x = null;

if (x.HasValue)

Console.WriteLine($"x hat den Wert {x.Value}");

else

Console.WriteLine("x ist null");

**Übung: Arbeiten mit int?, char? und bool?**

1. Erstellen Sie eine neue Konsolenanwendung.
2. Deklarieren Sie die folgenden Nullable-Typen:
   * int? nullableInt
   * char? nullableChar
   * bool? nullableBool
3. Weisen Sie jeder Variable den Wert null zu und geben Sie den die Werte in der Konsole aus.  
   Console.WriteLine($"nullableInt: {nullableInt}, nullableChar: {nullableChar}, nullableBool: {nullableBool}");
4. Schreiben Sie jetzt Code, der prüft, ob jede Variable einen Wert hat:

* Für nullableInt: Gib "Der Wert ist: X" aus, wenn ein Wert vorhanden ist, oder "nullableInt ist null".
* Für nullableChar: Gib "Der Buchstabe ist: X" aus, wenn ein Wert vorhanden ist, oder "nullableChar ist null".
* Für nullableBool: Gib "Die Bedingung ist: X" aus, wenn ein Wert vorhanden ist, oder "nullableBool ist null".

1. „Spielen Sie sich“ nun ein bisschen mit dem Programm, indem Sie den Variablen unterschiedliche Werte zuweisen und schauen, ob die Ausgabelogik (Console.WriteLine …) richtig reagiert.**2. Null conditional operator (?.) Null-Bedingungsoperator ?**

**Definition**

* Mit **?.** kann man prüfen, ob eine Variable **null** ist, **bevor** man auf ihre Eigenschaften oder Methoden zugreift.  
  Das verhindert **NullReferenceExceptions**.

**Beispiel mit Abbruch des Programms**

Person? person = null;

Console.WriteLine(person.Name);

Ergebnis: NullReferenceException

**Beispiel mit geschütztem Zugriff auf nullfähiges Objekt**

**Person? person = null;**

Console.WriteLine(person**?**.Name);

**Mit ? wird sicherer Zugriff auf Eigenschaften oder Methoden gewährleistet**

**Anwendungsfälle**

* Zugriff auf verschachtelte Objekte:

string city = person?.Address?.City;

* Aufrufen von Methoden, **nur** wenn das Objekt nicht null ist:

person?.PrintInfo();

**Übung**

Erstellen Sie eine neue Klasse **Person** mit folgenden Eigenschaften

public class Person

{

public string? Vorname { get; set; }

public string? Nachname { get; set; }

public Adresse? Wohnadresse { get; set; }

}

public class Adresse

{

public string? Stadt { get; set; }

public string? Straße { get; set; }

}

2. Erstellen eine neue Instanz der Klasse Person und lassen Sie die Eigenschaften Wohnadresse und Straße auf null.

3. Verwenden Sie den **Null Conditional Operator**, um **sicher** auf die Wohnadresse-Eigenschaft zuzugreifen

string strasse = ……………………………….;

Console.WriteLine($"Straße: {strasse ?? "Unbekannt"}");

**3.** **Nullable reference types (speziell Markierung nur nullfähige Verweistypen)**

**Definition**

* Reference Types (Verweistypen), wie Strings, Arrays, aller Art Objekte) können **intern** so markiert werden, dass sie **nicht null sein dürfen** (null zu sein können sie ja schon „von Haus aus“).  
  Die „Markierung“ besteht aus der Einstellung **<Nullable>enable</Nullable>** (global im Projekt-File) oder aus #**nullable enable** (auf Dokumentebene).
* Das aktiviert **Warnungen vom Compiler**, wenn versucht wird einer **nicht mit „?“** markierten Variable, **null zuzuweisen**.

**Beispiel**

#nullable enable

string name = null; // Compiler-Warnung

string? name2 = null; // Kein Fehler

**Vorteile**

* Reduziert die Anzahl von NullReferenceExceptions.
* Verhindert Bugs, die durch null-Zuweisungen entstehen.
* Selbstdokumentierender Code: Jeder sieht, dass string? null sein kann.
* Damit kann kann man NOT-Attribute (Felder) der Datenbanktabellen 1:1 mappen.

**Übung**

1. Stellen Sie sicher, dass das Nullable-Feature in der Projektdatei aktiviert ist. In der \*.csproj-Datei sollte der folgende Eintrag vorhanden sein:

<Nullable>enable</Nullable>

1. Erstellen Sie eine Klasse Produkt mit folgenden Eigenschaften:

public class Produkt

{

public string Name { get; set; } // Nicht-nullbare Referenz

public string? Beschreibung { get; set; } // Nullable Referenz

}

1. Schreibe den folgenden Code:

Produkt produkt = new Produkt

{

Name = "Laptop",

Beschreibung = null

};

Console.WriteLine($"Produktname: {produkt.Name}");

Console.WriteLine($"Beschreibung: {produkt.Beschreibung.Length}");

1. Beheben das Problem mit zuvor gelernten “Null Conditional Operator” und führen Sie das Programm noch einmal aus.

**3. Null Coalescing Operator (??) - NULL-Sammeloperator (??)**

**Definition**

* **??** wird verwendet, um einen **Standardwert** zu setzen, **wenn der linke Operand null ist**.

**Beispiele**

string? name = null;

string result = name ?? "Unbekannt";

Console.WriteLine(result); // Ausgabe: Unbekannt

Fallback-Werte für null-Variablen:

int? zahl = null;

int anzahl = zahl ?? 42; // Wenn zahl null ist, wird 42 verwendet

Null-Sicherer Zugriff mit ?. und ?? kombiniert:

string? city = person?.Address?.City ?? "Unbekannt";

**Übung**

Schreiben Sie ein Programm, das den Namen des Benutzers abfragt und geben Sie den eingegebenen Benutzernamen aus.  
Sollte kein Name übermittelt worden sein, muss der nicht eingegeben Werte auf null gesetzt werden. Verwenden Sie in Ihrer Logik den Null Coalescing Operator, um zw. diesen NULL-Wert und echten Namen zu unterscheiden.