
# Null Safety

2023年10月25日(水)

# アジェンダ

1. Nullとは？
2. Null Safetyとは？
3. 試してみた
4. まとめ

# NULLとは

プログラムの世界では、
「**何もない**」ということを表します。
“0”ではなく、”何もない”

1

0

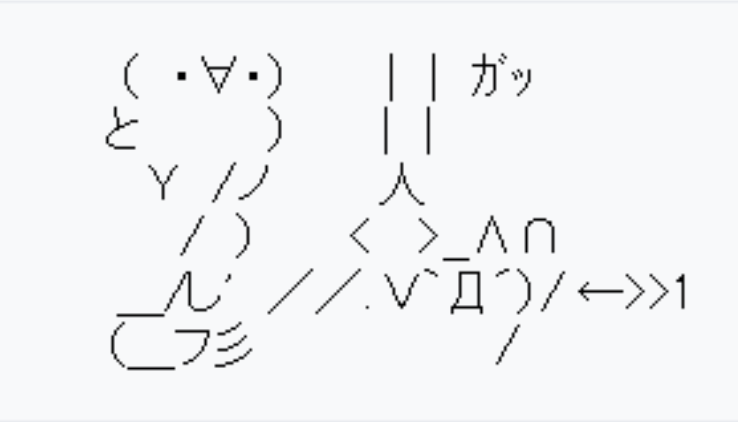
NULL

# よく見るやつ、NULL参照

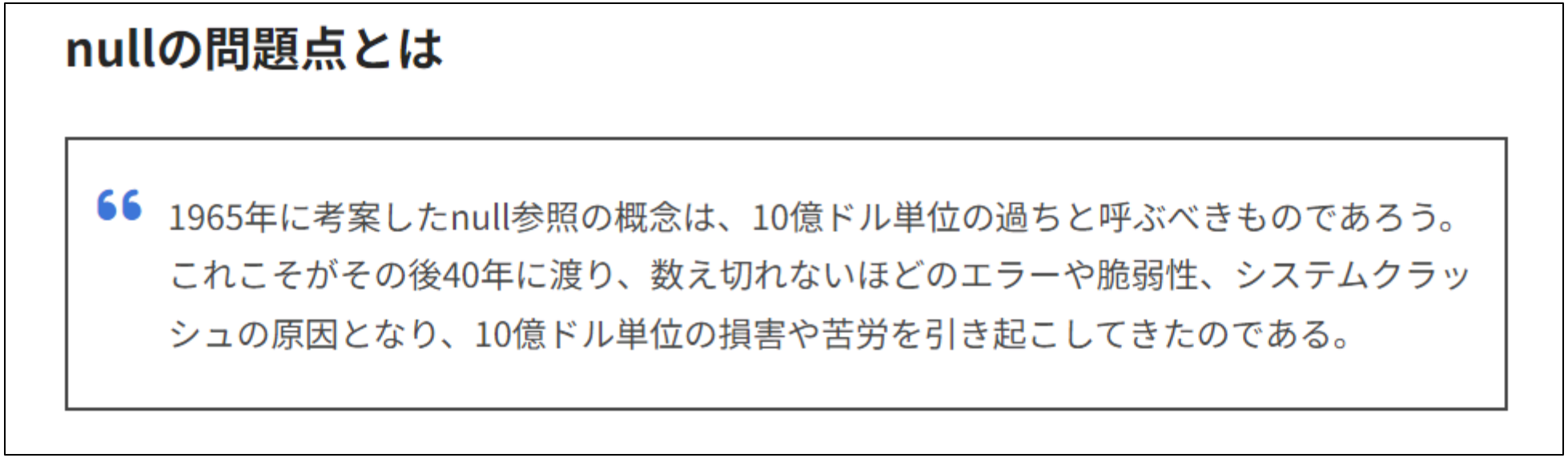
例) Java

**Java.lang.NullPointerException**

**→値がnullの変数を参照した時に発生する例外**

# よく見るやつ、NULL参照




**という言葉が飛び出すほど、影響を及ぼしてきた**

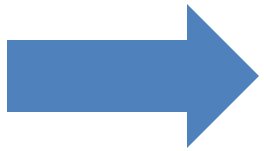
引用： https://genesis-tech.jp/blog/null-safety-of-csharp/

# NULLへの対策

```
1  var person = TryGetPerson(id);
2  string name = person.Name;
```

(^__^)
( ´∀`)＜ぬるぽ

変数・personがnullで返却されてきた場合、
personのNameにアクセスしようとすると、
いわゆる「null参照」で落ちることになる。



```
1  var person = TryGetPerson(id);
2  if( person == null )
3      return;
4  string name = person.Name;
```

回避するにはnull判定のコードを
書く必要がある。

上記のように、様々な場面でnullを想定して回避する実装をすれば対処できるが、
見落としなどにより、nullチェックができていない実装や、
nullが来ることを期待していないのに、nullが来てしまう実装が存在すると、
null参照で落ちる危険性がある。

引用： https://genesis-tech.jp/blog/null-safety-of-csharp/

このNULLにつきまとう問題を解決するため、
先人たちの努力により、新たな仕組みが作られました。

# Null Safety とは

※C#を例に

**NULLが原因で発生するエラーをコンパイル時に防止する仕組み**

**NULL安全 とも呼ばれる**

→NULLを扱える変数、扱えない変数を区別して
コンパイル時からNULLが原因で発生するエラーを防止する

# 参考「null許容値型」と「null参照許容型」

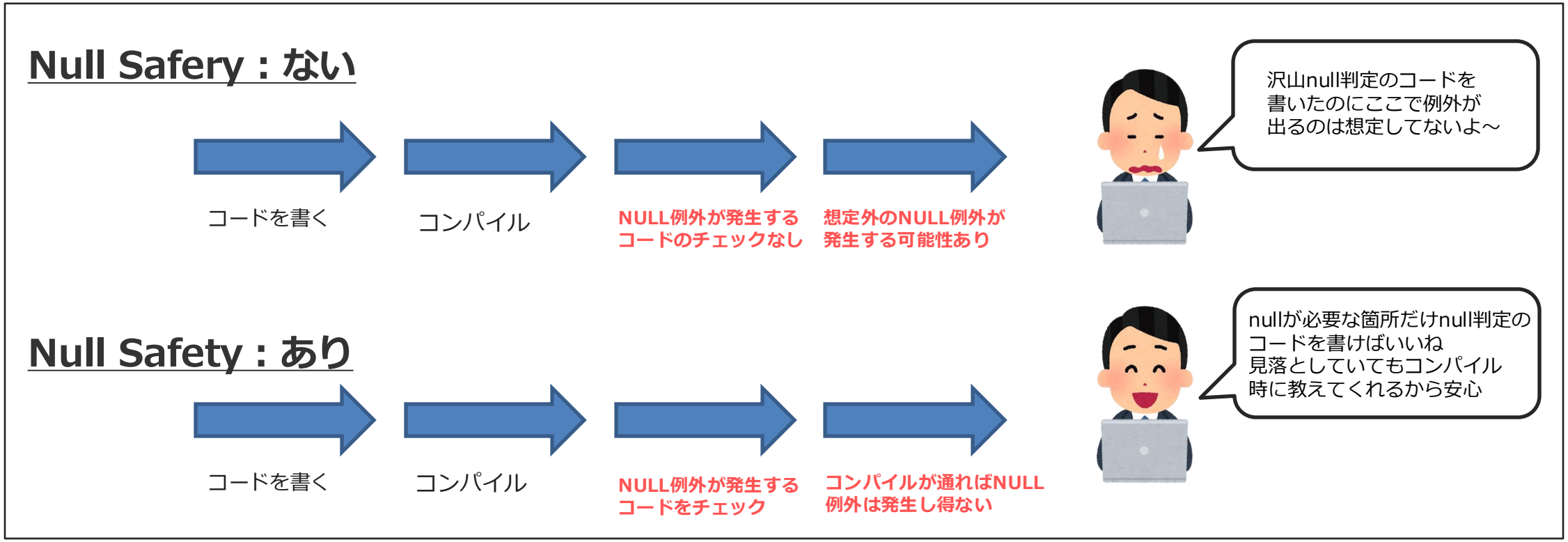
もともと、C#では値型（intやstructなど）は**null値が取れず**、
値型の変数でnull値を代入するには、**「null許容値型」**を使う必要があった。
(int? のように？を付けて宣言)
一方、参照型（stringやobjectなど）は、**デフォルトでnullが許容**されているので、
**許容・非許容が区別されていなかった。**

**「null参照許容型」**の登場により、 C#でも**参照型の「null許容 / null非許容」**の
区別が付くようになったので、 null参照例外が発生するコードは、
コンパイル時に検出してくれるようになり「null安全」が満たされるようになった。

引用： https://genesis-tech.jp/blog/null-safety-of-csharp/

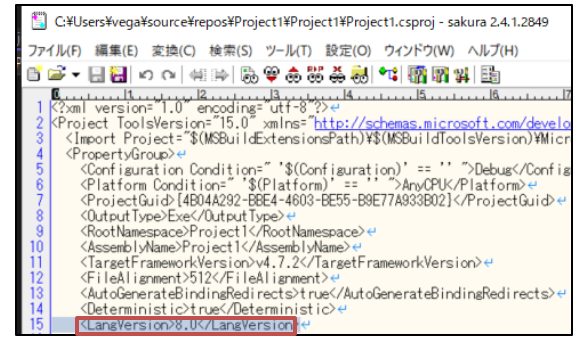
# よいところ

・想定外の箇所で変数がnullになり、不具合になってしまうことを防げる。
・nullの可能性を想定して、nullを判定するコードを書く必要がなくなる。
(nullが必要な箇所は区別して、nullを判定するコードを書く。)



VEGA Systems

# NULL SAFETY の使い方




→「**null参照許容型**」 **を有効にする**　※C# 8.0から導入

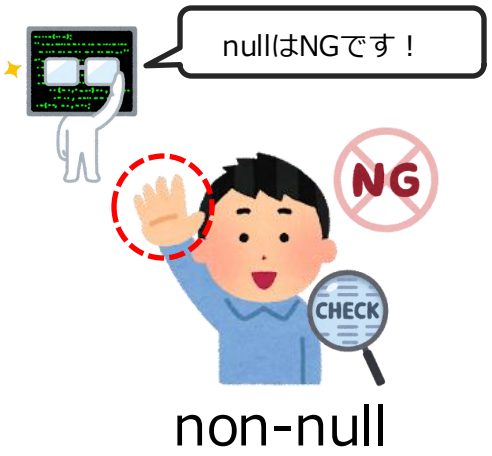
有効にすると、**参照型のT**に対して、**nullを認めなくなる**(null非許容 → non-null)
このため、nullを設定できない型にnullを設定しているコードは、
コンパイル時に検出されるようになった。

```
#nullable enable
        object obj2 = null; // warning CS8600  Null リテラルまたは Null の可能性がある値を Null 非許容型に変換しています。
        string s1 = null;  // warning CS8600  Null リテラルまたは Null の可能性がある値を Null 非許容型に変換しています。
        int i1 = null; // error CS0037: Null 非許容の値型であるため、Null を 'int' に変換できません。
```

修正しなきゃ

```
#nullable enable
        object obj2 = "hoge";
        string s1 = "hogeta";
        int i1 = 1;
```

コンパイル時に検出される




参考：https://ufcpp.net/study/csharp/resource/nullablereferencetype/

# NULL SAFETY の使い方

仕様や運用上で、nullを扱う場合には、
**T?** の形で記述することで、**nullを許容することができる**(null許容 → nullable)
※nullを許容してしまうので、従来通りnullチェックの構文は書く必要がある。

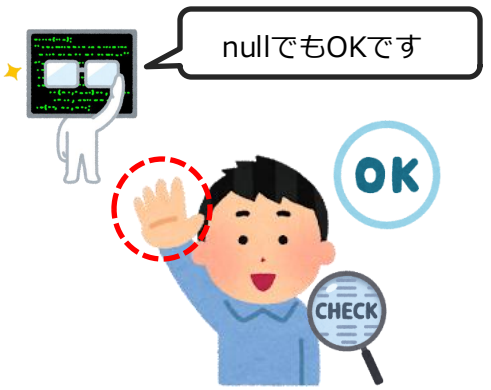
```
#nullable enable
        object? obj3 = null;

        string? s2 = null;

        int? i2 = null;
```

↓

コンパイルエラーにならない




nullable

# 3. 試してみた

■:警告　■:エラー　■:OK

```
// ※C#のバージョンは8.0以上にしておくこと

object? obj1 = null; // warning CS8632: '#nullable' 注釈コンテキスト内のコードでのみ、Null 許容参照型の注釈を使用する必要があります。

#nullable enable

object obj2 = null; // warning CS8600  Null リテラルまたは Null の可能性がある値を Null 非許容型に変換しています。
object? obj3 = null;


string s1 = null;  // warning CS8600  Null リテラルまたは Null の可能性がある値を Null 非許容型に変換しています。
string? s2 = null;


int i1 = null; // error CS0037: Null 非許容の値型であるため、Null を 'int' に変換できません。
int? i2 = null;
```

ディレクティブ外で “?” を定義すると警告が出る

ディレクティブ

コンパイル

```
ビルドを開始しました...
1>------ ビルド開始: プロジェクト: Project1, 構成: Debug Any CPU ------
1>C:¥Users¥vega¥source¥repos¥Project1¥Project1¥NullSafety.cs(18,19,18,20): warning CS8632: '#nullable' 注釈コンテキスト内のコードでのみ、Null 許容参照型の注釈を使用する必要があります。
1>C:¥Users¥vega¥source¥repos¥Project1¥Project1¥NullSafety.cs(29,22,29,26): error CS0037: Null 非許容の値型であるため、Null を 'int' に変換できません
1>C:¥Users¥vega¥source¥repos¥Project1¥Project1¥NullSafety.cs(21,27,21,31): warning CS8600: Null リテラルまたは Null の可能性がある値を Null 非許容型に変換しています。
1>C:¥Users¥vega¥source¥repos¥Project1¥Project1¥NullSafety.cs(25,25,25,29): warning CS8600: Null リテラルまたは Null の可能性がある値を Null 非許容型に変換しています。
========== ビルド: 成功 0、失敗 1、最新の状態 0、スキップ 0 ==========
========== ビルド は 5:06 PM に開始され、00.312 秒 かかりました ==========
```

VEGA Systems

## 下記の条件演算子や、合体演算子を組み合わせて使う

```
//null条件演算子 ns が null の場合は name1 は null。 null ではない
var ns = new Ns();
ns.name = null;
var name1 = ns?.name;

// null合体演算子 personNameがnullの場合、"is null 1" を代入する 。
string personName = null;
var name2 = personName ?? "is null 1";

// null条件演算子 + null合体演算子 personNameがnullの場合、"is null
var name3 = ns?.name ?? "is null 2";
```

参考： https://atmarkit.itmedia.co.jp/ait/articles/1607/07/news023.html

# まとめ

・NULLとは、”0”ではなく”何もない”
・NULLは昔から影響を及ぼしてきた

**★Null Safetyは、NULLが原因で発生するエラーをコンパイル時に防止する仕組み**

・NULLを扱える変数、扱えない変数を区別してプログラムの安全性を高める。
(null許容型とnull非許容型)
・想定外の箇所で変数がnullになり、不具合になってしまうことを防げる。
・nullの可能性を想定して、nullを判定するコードを書く必要がなくなる。
(nullが必要な箇所は区別して、nullを判定するコードを書くだけでよい。)

**→NULLになり得る状況を見逃さないようにして、そのための記述もシンプルに書きやすく！**