# 第 12 章

# C#の布教

**74 回生 cobalt**

こんにちは、cobalt です。私は C#というプログラミング言語が大好きなので、C#について書いていこうと思います

## 12.1　まずプログラミング言語って何

プログラミング言語とは、コンピューター (この場合はスマートフォンなんかも含みます) に命令するために作られた人工言語です。長い歴史 (70 年ほど) と多くの種類があり、コンピューターの扱う 2 進数に近いものや、人間の扱う言葉に近いもの、3 種類の言葉しか使わないものなど色々な種類があります。

## 12.2　じゃあ C#って何

C#は、アンダース・ヘルスバーグというプログラマが設計した言語で、C という言語を改良して作られた C++ という言語に影響を受けて作られたのが C#です [C] → [C++] → [C+++] という理屈に基づき、プラスマーク四つを右上、右下、左上、左下に並べると#に見えることから、C++++ = C#という名前になったと言われています。

## 12.3　特徴

構文は C や、C++、それと Java などの影響を受けています。オブジェクト指向と呼ばれる部類の言語なので、クラスの概念があります。Microsoft .NET Framework の中心言語であり、.NET Framework において最も高い生産性を持つと言われています。Unity のスクリプトは C#で書く必要があります。Unity のようなゲーム製作にも向いている言語ということです。

## 12.4　実例

### オブジェクト指向の代名詞とも言えるクラスによって、コードをスマートに記述できる。

ゲーム制作を例に挙げましょう。今、ゲームの中に「小石」や「棒」と言ったアイテムを作るとします。クラスを用いない場合は、`StoneID` 、`StickName` と言った変数で ID や名前を管理するでしょう。「剣」を追加するとなれば、`SwordID` や `SwordName` などの変数に加え、`SwordAttack` などの関数も必要になります。これではどの変数がどのアイテムのものなのかが伝わりにくく、読みづらくて仕方ありません。

ここでクラスの出番です。クラスを使う一つ目の利点は、`StoneName` や `StoneID` を `Stone.Name`　`Stone.ID` のように纏める事ができることです。ID と名前を `Item` クラスに纏める事で、Item をベースに作った `Stone` や `Stick` から ID と名前にアクセスできるようになります。これなら変数も無闇に増えませんし、どのアイテムの ID なのかも一目でわかります。

二つ目の利点は、`Item` クラスを拡張したものをアイテムとして扱える事です。`Item` クラスを親とする `Weapon` クラスを作り、`Weapon` クラスに `Attack` 関数 (C#ではメソッドと呼びます) を定義します。`Weapon` クラスは `Item` クラスの特徴を自動で受け継いでいるので、`Weapon` クラスでも ID や名前を扱う事ができます。`Weapon` クラスを元に `Sword` を作れば、`Sword` から ID、名前に加え、`Attack` メソッドを扱う事が出来ます。

ピストルやマシンガンといった銃火器も追加してみましょう。`Weapon` をベースに `Pistol` と `MachineGun` を作って... ですが

Weapon クラスは剣や斧を想定したものなので、 Weapon クラスをベースとする銃は殴ることしかできません。 Weapon を書き換えるしかないのでしょうか。

そこで Weapon クラスをさらに拡張してみましょう。 Weapon クラスを親とする Gun クラスを作り、射程や弾速を Range 、BulletSpeed というように定義します。 Gun.Attack を遠隔攻撃に書き換えてしまいましょう。 override Attack というように、 override キーワードを使うことで Attack 関数の挙動を乗っ取って、遠隔攻撃に書き換えてしまいます。親である Weapon クラスには影響は及ばないので、安心して遠隔攻撃できます。

これで Weapon をベースに作った Sword の挙動を変えることなく、 Gun クラスをベースに作る Pistol や MachineGun を追加することができます。 Item クラスの要素である Pistol.ID 等もちゃんと使えます。

### アクセス修飾子、プロパティによってカプセル化ができる。

この機能は共同開発で役に立つものです。A さんと B さんによる共同開発を例にしましょう。A さんは Weapon に武器のリーチを示す変数と、リーチを設定するメソッドを作りました。このメソッドはリーチを負の値に設定できないようにするためのものでした。B さんはそのメソッドの存在を知らず、武器のリーチを直接 = で変更するコードを書きました。ここで B さんのコードに不備があり、武器のリーチが負の値に設定され、エラーが発生しました。

この事件の原因はリーチ変数がメソッドを介せずに変更できてしまうことです。C#(などの言語) にはこれを防ぐ仕組みとしてアクセス修飾子があります。クラス内で変数を宣言する際、前に private と記述することでクラス内からしかその値を扱えなくなります。この「クラス内」には継承先のクラスは含まれていません。継承先のクラスもその変数を扱えるようにするには protected と記述します。リーチ変数を変更するメソッドを作り、このメソッド内でリーチを負の値に設定することができないように記述すればエラーを防ぐことができます。どこからでも自由にアクセスできるようにするには public と記述します。

クラスが複雑になってくると、 private や protected の数も増えてきます。この変数全てに setID setName setReach とメソッドを書いていくのは手間がかかります。そこで C#(と VB) 独自の機能、自動実装プロパティの出番です。変数の宣言の後に中括弧で、変数の値を変更するメソッド set (自動で引数 value が用意されます) と変数の値を取得するメソッド get を書くことができます。

この set , get は = で変数の値を変更する/取得する場合にも適用されます。リーチ変数の set の中に負の値に設定できないようにすれば、 = を用いて変数の値を負の値に指定しようとしても無視されるようになります。また、 private set; public get; と書けば、値の変更はクラス内でのみ行え、値の取得は自由に行える「取得専用」の変数を作ることができます。

### ガベージコレクションが優秀

C++ で任意の長さの配列が欲しいときは new キーワードを用います。また、不要になった配列は、 delete キーワードを使って明示的に破棄する必要があります。

対して C#には、C++ の delete キーワードのようなものはありません。不要になった配列やクラスは、CPU が暇なときにまとめて破棄されます (ガベージコレクションと呼ばれます)。Python などにも同じような機能がありますが、C#(と Java) のガベージコレクションは他の言語より優秀で、不要なものを確実に破棄できるといわれています。

明示的な破棄が必要なクラスもあるでしょう。C#はそのようなクラスの為に、 using ステートメントというものを用意しています。 using ステートメントを使って宣言されたクラスは、ステートメントの終了と同時に破棄されます。この機能を使えば、delete を忘れた配列がメモリを圧迫するようなことはなくなります

### 多次元配列が高速

多次元配列は便利です。配列を入れ子状に宣言することで、書けるコードは大きく広がります。多くの言語の多次元配列は配列を配列にしてアクセスしています。多次元配列へのアクセスは、配列へのアクセスを次元数回繰り返す操作になります。100 回程度のアクセスならどうでもいいですが、10^7 以上になってくるとこの繰り返しが問題になってきます。

C#の多次元配列は、配列の配列ではありません。連続したメモリ領域として確保されます。これにより多次元配列へのアクセスは次元数の逆数倍に高速化されます。

### LINQ が最強

「配列に `for` を回して、中身が 0 に等しい要素の数を数える」「配列に `for` を回して、中身の要素を絶対値に書き換える」このような配列に対する成形、計測などの操作、頻繁に使いますよね。でも毎回 `for` 文を書くのは面倒... そこで C#の出番です。

この 2 つの操作は `Count(v=>v==0)` 、 `Select(v=>Math.Abs(v))` で出来ます。別の配列を作ったり `for` で index を気にする必要はありません。これが Language INtegrated Query、LINQ の力です。

LINQ にはこの 2 つの他にも様々な力を秘めています。条件を満たす要素の抽出や、重複要素の排除、条件を満たすはじめの要素の取得と、LINQ に含まれるメソッドの数は 50 を超えます。LINQ を使いこなすことで、C#のコードはより読みやすく、短くなるのです。

## 12.5 終わりに

C#には便利な機能が枚挙に暇がないほどあります。ここではその中でも特に便利な 5 つに絞って紹介しました。この部誌を読んで少しでも「C#って便利そうだな」と感じていただけたら幸いです。