入力練習用・毎回使う

C# 入力チェックシート

赤い波線を残さず、名前を認識させてから使う

合言葉	**定義する → IntelliSense で確認する → 使用する → 赤い波線がないことを確認する**

このチェックシートの目的

コードを上から順に写すことよりも、**Visual Studio が名前を認識できる順番**で入力することを優先します。途中の赤い波線を放置せず、一つずつ意味を確認して進みます。

1	2	3	4
先に定義	**候補を確認**	**名前を使用**	**赤線ゼロ**
名前の置き場所を作る	IntelliSense を使う	候補から選んで入力	確認して次へ進む

入力を始める前

- □ 今日入力する完成コードを一度、最後まで見る。
- □ **自分で作る名前に印を付ける。** 例：変数、メソッド、クラス、プロパティ
- □ どの名前が、どこから使われるかを見る。
- □ 依存される側（使われる側）を先に作る順番を決める。

絶対に守る 4 つのルール

1	**赤い波線を残したまま、次の行へ進まない。**
2	**教科書を上から順に写すことにこだわらない。**
3	**自作メソッドなどは、定義を先に作ってから呼び出す。**
4	**名前はできるだけ IntelliSense の候補から選ぶ。**

1　名前の種類ごとの入力順

名前を使う前に、その名前を Visual Studio が認識できる状態にします。

名前の種類	先にすること	先に作る例	その後で使う例
変数	宣言・初期化を作る	`int score = 80;`	`Console.WriteLine(score);`
自作メソッド	メソッド定義を作る	`static void ShowMessage() { … }`	`ShowMessage();`
クラス	class の定義を作る	`class Student { … }`	`new Student();`
プロパティ／フィールド	クラス内にメンバーを作る	`public int Score { get; set; }`	`student.Score`
自作メンバー	メソッドやプロパティを追加する	`public void ShowScore() { … }`	`student.ShowScore();`

2　自作メソッドは「定義→呼び出し」

おすすめの入力順

- □ Program クラスと空の Main メソッドを作る。
- □ Main の外側・Program クラスの内側に、ShowMessage メソッドを作る。
- □ メソッド定義に赤い波線がないことを確認する。
- □ Main に戻り、IntelliSense から ShowMessage を選んで呼び出す。
- □ コード全体に赤い波線が残っていないことを確認する。

完成形（入力するときは、ShowMessage の定義を先に作る）

```
class Program
{
    static void Main()
    {
        ShowMessage();
    }

    static void ShowMessage()
    {
        Console.WriteLine("こんにちは");
    }
}
```

3　IntelliSense を使って入力する

目的	**候補に名前が出ることは、その場所で Visual Studio が名前を認識しているかを確認する手掛かりです。**

基本の使い方

□ 数文字入力したら、候補一覧を見る。
□ 使いたい名前が候補に出たら、候補を選んで確定する。
□ 候補に出ないときは、最後まで手入力せず、一度止まる。
□ 大文字・小文字、スペル、定義した場所、static の有無を確認する。

候補に出ないときの確認順

順	考えられる原因	すること
A	名前の定義をまだ作っていない	先に定義を作る。
B	スペルや大文字・小文字が違う	定義側の名前と見比べる。
C	使う場所が違う（スコープ外）	波かっこ { } の範囲を確認する。
D	static の対応が違う	Main から呼べる形か確認する。
E	ピリオドの左側の型が違う	変数の型と作ったメンバーを確認する。

4　赤い波線が出たら、その場で止まる

止	見	比	直
入力を止める	**メッセージを見る**	**定義と比べる**	**直して再確認**
次の行へ行かない	カーソルを合わせる	名前・場所・型を確認	赤線ゼロで再開

□ 赤い波線の場所にマウスを合わせ、表示された内容を読む。
□ **エラー一覧の一番上から確認する。**　後ろのエラーは連鎖していることがある
□ 直前に入力した 1〜3 行と、対応する定義を見る。
□ 直せないときは、消してごまかさず、講師に見せる。
□ 直した後、赤い波線が消えたことを確認してから再開する。

5　入力中・入力後の最終チェック

入力中：1 つのまとまりを作るたびに確認

□ 今から使う名前は、すでに定義されている。

□ 名前は IntelliSense の候補から選んだ。

□ 波かっこ { } と丸かっこ ( ) の対応が取れている。

□ セミコロン ; の必要な場所を確認した。

□ 赤い波線が残っていない。

入力後：実行する前に確認

□ エラー一覧にエラーが残っていない。

□ 自作メソッドの定義と呼び出しの名前が同じ。

□ 変数は宣言した範囲の中で使っている。

□ Main から呼ぶメソッドの static が合っている。

□ 保存してから実行した。

□ 実行結果が完成例と同じ。

記録欄

日付	教材・課題	赤線ゼロで実行	気づいたこと／次回の注意
		□ はい　□ 要確認	
		□ はい　□ 要確認	
		□ はい　□ 要確認	

講師へ相談するときの伝え方

伝える　**「どの名前を使おうとしたか」「どこまで定義したか」「IntelliSense に出たか」「表示されたエラー」を順に説明する。**