継承禁止

シールドクラス

何らかの要因で、これ以上派生する必要のないクラスを生成した場合
不用意なプログラムがこのクラスを継承できないようにする方法があります

継承を禁止したクラスを シール付きクラスまたはシールドクラスと呼び
このクラスを基底クラスとした場合、コンパイルエラーを出すようにします
シールドクラスの宣言には sealed 修飾子をクラスに指定します

```
sealed class Kitty {
 public void Write() {
  System.Console.WriteLine("Kitty on your lap");
 }
}

//class Taruto : Kitty {} //エラー、継承できない

class Test {
 static void Main() {
  Kitty obj = new Kitty();
  obj.Write();
 }
}
```

シールドクラス Kitty を他のクラスに継承させることはできません
シールが意味するのはそれだけで、他にとくに影響は現れません

sealed 修飾子は abstract と同時に宣言することはできないので注意してください
abstract は何らかのクラスに継承されることが前提であるため
sealed とはまったく正反対の性質なので同時には宣言できないのです

シールドクラスは、宣言することによってクラスが最適化されます
これによって派生クラスを持たないことが保証されるので
仮想メソッドは非仮想メソッドに最適化されるなどのメリットがあります

シールドメソッド

クラスの継承を防ぐ方法と同様に、メソッドのオーバーライドを拒否することもできます
仮想メソッドのオーバーライドはクラス階層が何段続いても可能でしたが
特定のオーバーライド以降、メソッドをオーバーライドしてほしくない時に指定します

シールドメソッドの宣言は、オーバーライドの宣言と同時でなければなりません
シールドメソッドの宣言は sealed 修飾子をメソッドに指定します

```
class Kitty {
 public virtual void Write() {
  System.Console.WriteLine("Kitty on your lap");
 }
}

class Taruto : Kitty {
 public sealed override void Write() {
  System.Console.WriteLine("Magical nyan nyan TARUTO");
 }
}

class TokyoMM : Taruto {
 //public override void Write() {}//エラー、オーバーライドできない
}

class Test {
 static void Main() {
  Kitty obj = new Taruto();
  obj.Write();
 }
}
```

Taruto クラスの Write() は基底クラスの Write() メソッドをオーバーライドしています
と同時に、sealed 修飾子でこれ以上オーバーライドできないようにしています
このクラスの派生クラスは Write() メソッドをオーバーライドすることはできません

sealed

シールドを宣言します
これ以上の派生、オーバーライドを防ぎます
abstract 修飾子と同時に使用することはできません