インデクサ

配列的オブジェクト

C# 言語は、オブジェクトを配列のように扱う方法をサポートしています
これは、特定の処理において非常に強力な機能となるでしょう

オブジェクトを配列のように扱うこの機能を**インデクサ**と呼びます
ときに、プロパティが「スマートプロパティ」と呼ばれるのですが
これに対してインデクサを「スマート配列」と呼ぶことがあります

ここから連想できるように、インデクサの機能はプロパティの仕様に類似しています
つまり、配列のようなオブジェクト操作を暗黙的にメソッドに変換する機能がインデクサです

インデクサを使いこなすには、一定以上のオブジェクト指向の「設計」経験が必要になります
代表的な例としては、ベクトルのように内部で配列を扱うオブジェクトは
内部の配列とオブジェクトの操作をインデクサによって通信させるという方法をとります
こうすれば、メソッドの仕様を覚える必要なく、クラスの利用者は直感的にプログラムできるのです

インデクサは、プロパティ同様にクラスのメンバとして働きます
インデクサを宣言するには**インデクサ宣言子**で行います

[attributes] [modifiers] indexer-declarator {accessor-declarations}

attributes が属性、modifiers がインデクサ修飾子
indexer-declarator がインデクサ宣言子、accessor-declarations がアクセッサ宣言子群です
インデクサ宣言子は次のような構文を用います

type this [formal-index-parameter-list]

type はこのインデクサの型、formal-index-parameter-list は仮パラメータリストを指定します
プロパティは名前で識別されましたが、インデクサは**シグネチャで識別**されます

アクセッサ宣言子群はプロパティと基本的に同じです
get アクセッサはインデクサ型の戻り値と仮パラメータを持つメソッドに等しく
set アクセッサは void 型の戻り値と仮パラメータを持つメソッドに等しいです
さらに、set アクセッサは代入された値を暗黙的な変数 value で受け取っています

```
class Kitty {
 private string[] strName;
 private Kitty() {}

 public Kitty(string[] str) {
  strName = str;
 }
 public int Length {
  get { return strName.Length; }
 }

 public string this[int index] {
  get { return strName[index];  }
  set { strName[index] = value; }
 }

}
class Test {
 static void Main() {
  string[] str = new string[] {"Rena" , "Petit Charat" , "Mimi"};
  Kitty obj = new Kitty(str);
  GetKitty(obj);
  obj[1] = "Yuki";
  GetKitty(obj);
 }
 static void GetKitty(Kitty obj) {
  for (int i = 0 ; i < obj.Length ; i++)
   System.Console.WriteLine(obj[i]);
 }
}
```

このプログラムで作成した Kitty クラスは strName という文字列型の配列を扱います
この配列にアクセスする手段として、インデクサを採用しました

プログラムが長くなるために避けましたが
本来はアクセッサで取得した index が配列の範囲かどうかを確認し
もし不正なインデックスであれば例外を発生させるなどの処理を行うべきです