コレクション

応用編6日目

配列変数の問題点

- あらかじめ格納できるデータの数が決まっている
- データ数がわからない大量のデータを扱うのに不向き

コレクションとは

- 大量のデータを扱うことが出来るクラスライブラリ
- あらかじめ格納できるデータサイズがわからなくてもよい
- 様々なタイプがあり目的に応じて選択できる
- 英語で、**Collection(コレクション)**と書く

コレクションを使うには

- Collectionは、様々なデータ群を扱うクラス群
- ネームスペース「**System.Collections.Generic**」に含まれる

System.Collections.Genericの利用

```
using System.Collections.Generic;
```

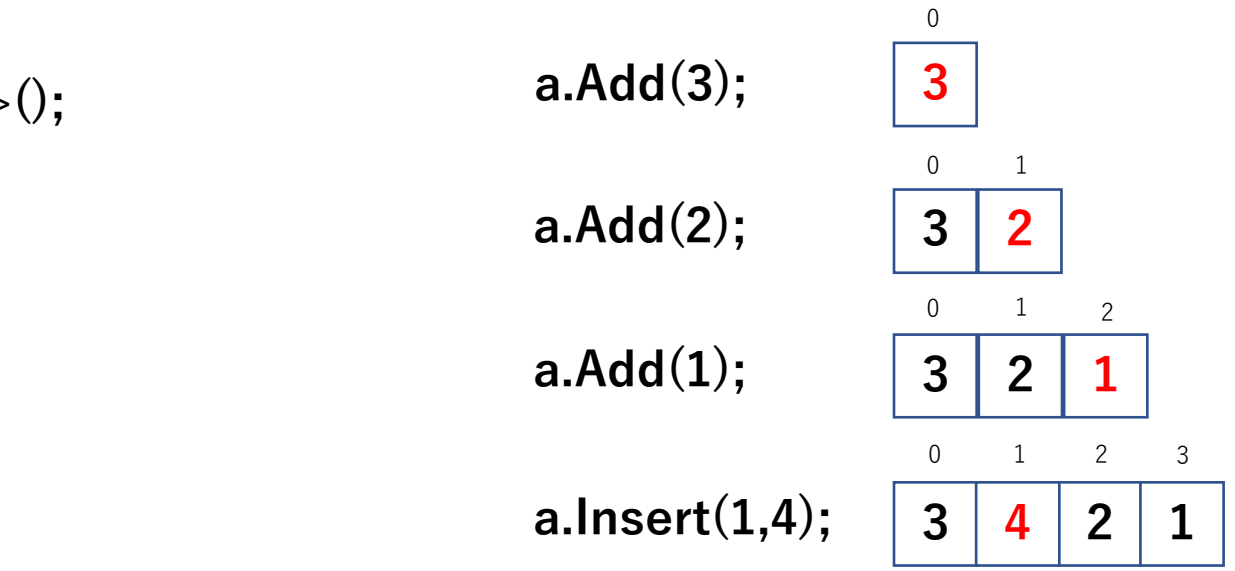
Listクラス

- 動的な配列
- 長さを自由に変えることが出来る
- データの挿入・削除が可能

Listクラスのデータの追加

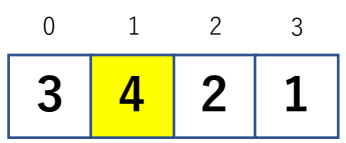
- データの追加は、**Add()**メソッドで行う
- データは、追加した順番に、0,1,2,…と、番号が割り振られる

```
List<int> a = new List<int>();
```

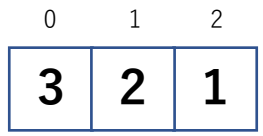


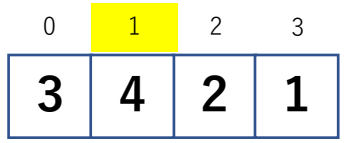
Listのデータの削除

- データの削除は、**Remove()/RemoveAt()**メソッドで行う
- その際、引数として与えた番号のデータがなくなるが、それ以降の番号は、繰り上がっていく

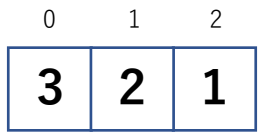


a.Remove(4);





a.RemoveAt(1);



Listクラスの主要メソッド

メソッド	働き
Add()	コレクションに要素を追加します
Insert()	コレクション内の指定した場所に要素を挿入します
Remove()	指定された番号の要素をコレクションから削除します
RemoveAt()	指定された番号の要素をコレクションから削除します
Clear()	中身をクリアする

様々な種類のコレクション

- Dictionary
- HashSet
- その他にも様々なコレクションがある

Dictionary①

- **Collection(コレクション)**はListばかりではない
- **Dictionary**は、キーと値を１セットとした要素の集まりを管理するクラス
- 通常の配列変数では、キーとして必ず数値をとるが、**Dictionary**は、キーと値の組み合わせを自由に設定することが可能
- **SampleEx603**参照

Dictionary ②

Dictionaryの書式

```
Dictionary<(キーのクラス),(値のクラス)>
```

Dictionaryにデータを格納

```
(変数名)[キー]=値;
```

Dictionaryからデータの取り出し

```
(変数名)[キー]
```

Dictionaryの構造

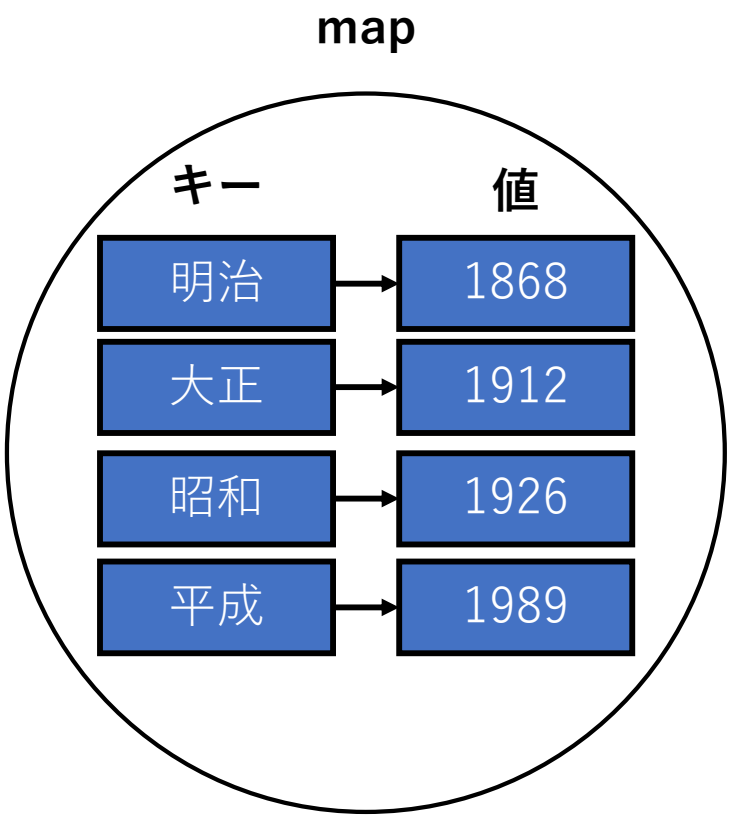
Dictionary<string,int> map = new Dictionary<string,int> ();

map["明治"] = 1868;

map["大正"] = 1868;

map["昭和"] = 1868;

map["平成"] = 1868;



HashSet

- **重複なくデータを格納できる**コレクション
- **Add**メソッドを用いて、データを格納する
- 追加される要素に同じものが含まれていても一つとみなす
- SampleEx604参照

HashSetの構造

HashSet<string> s = new HashSet<string>();