- ・ [Java 新人用 collection 学習資料](#)

【連載】コレクションフレームワーク

- ・ スタックアスタリスク

第 1 回：コレクションフレームワークの概要

- ・ http://www.stackasterisk.jp/tech/java/collection01_01.jsp

配列は関連するデータを扱うものとして大変便利ですが、配列の大きさを変更できないという制限があり、機能は単純なものに限られています。

Java ではオブジェクトのグループを操作できるよう、Java 開発当初から配列に加えて Vector や HashMap などのクラスが提供されてきました。

JDK1.2 以降はこれらのクラスよりもはるかに汎用化の進んだコレクションフレームワークというものがサポートされるようになりました。

- ・ 分類 List / Map / Set

よく使うもの

ArrayList

- ・ 作成 `ArrayList arraylist = new ArrayList();`
- ・ 追加

```
arraylist.add(追加したいオブジェクト);
```

- ・ 取得

```
Object obj = arraylist.get(i);
```

HashMap

- ・ 作成 `HashMap hashmap = new HashMap();`
- ・ `put(key,value);`

```
Object obj = hashmap.get(key);
```

- ・ キー一覧の取得

```
Set keys = hashmap.keySet();
```

Collection

- ・ コレクション間の変換が簡単
- ・ 反復子 iterator が共通で使える

配列と List の違い

	配列	ArrayList
要素の長さ	変更不可	変更可能
データ型	定義した型だけ	java.lang.Object
int や char などの基本データ型	配列作成が可能	Integer や String に変換して配列作成

第 2 回 : List の活用法

- http://www.stackasterisk.jp/tech/java/collection02_01.jsp

```
array=new ArrayList();
array.add(obj1);
array.add(obj2);
array.add(obj3);
int size=array.size();
Order od = array.get(1);
```

第 3 回 : Map の活用法

- http://www.stackasterisk.jp/tech/java/collection03_01.jsp

```
HashMap map = new HashMap();
map.put("09W0011",obj11);
map.put("09W0012",obj12);
map.put("09W0013",obj13);
int size=map.size();
```

キー値を重複して追加した場合、キー値に関連付けられた値が上書きされることがわかります。

HashMap を使用すれば「キーを付けて値を関連付ける」ことはできますが「順序の管理」はできません

TreeMap クラスは SortedMap インターフェースを実装したクラスで、マップが確実にキーの昇順でソートされます。

第 4 回 : Set の活用法

- http://www.stackasterisk.jp/tech/java/collection04_01.jsp

コレクション内のオブジェクトの順番を管理することができない。

List や Map にあった get() メソッドは Set にはない。

HashSet クラスは Set インターフェースを実装したクラスの中で最も高速に動作します。

HashMap クラスのメソッドには、Set オブジェクトを返すメソッドが用意されています。

Map に格納されているオブジェクトを Set に変換して、Set から iterator() メソッドを呼び出して全ての値を取得する目的で良く使われます。

第 5 回 : コレクションフレームワークで使用する様々な機能 (前編)

第 6 回 : コレクションフレームワークで使用する様々な機能 (後編)

第 7 回 : JDK5 でのコレクションフレームワークの変更点

- http://www.stackasterisk.jp/tech/java/collection07_01.jsp

コレクションフレームワークから要素を取り出すときにキャストが不要であったり、基本データ型のラッピングオブジェクトを使った変換が自動化されたりと、コードをシンプルに書くことができるようになりました。

- 1.4

```
ArrayList orders = new ArrayList();  
Order order1 = (Order) orders.get(1);
```

- 1.5

```
ArrayList <Order> orders = new ArrayList <Order>();  
Order order2 = orders.get(1);
```

拡張した for 文を使用すると、コードが更にシンプルになります。この for 文では Iterator を使用することなく、コレクションに入っている全ての要素を取得することができます。