


# 配列の仕組み

## 配列のコピー

```
1 public class Main {
2
3     public static void main(String[] args) {
4         // 大学のテストの点数を格納する配列を作成
5         int[] scores = { 72, 85, 95, 77 };
6         /*
7          * scores配列の内容をコピーして保持するcopyScoresを宣言して scoresを代入する
8          */
9         int[] copyScores = scores;
10        // copyScoresのインデックス番号0に100を代入する
11        copyScores[0] = 100;
12        // scoresのインデックス番号0の内容を表示
13        System.out.println("scores[0]の値:" + scores[0]);
14        // copyScoresのインデックス番号0の内容を表示
15        System.out.println("copyScores[0]の値:" + copyScores[0]);
16    }
17 }
```

実行結果



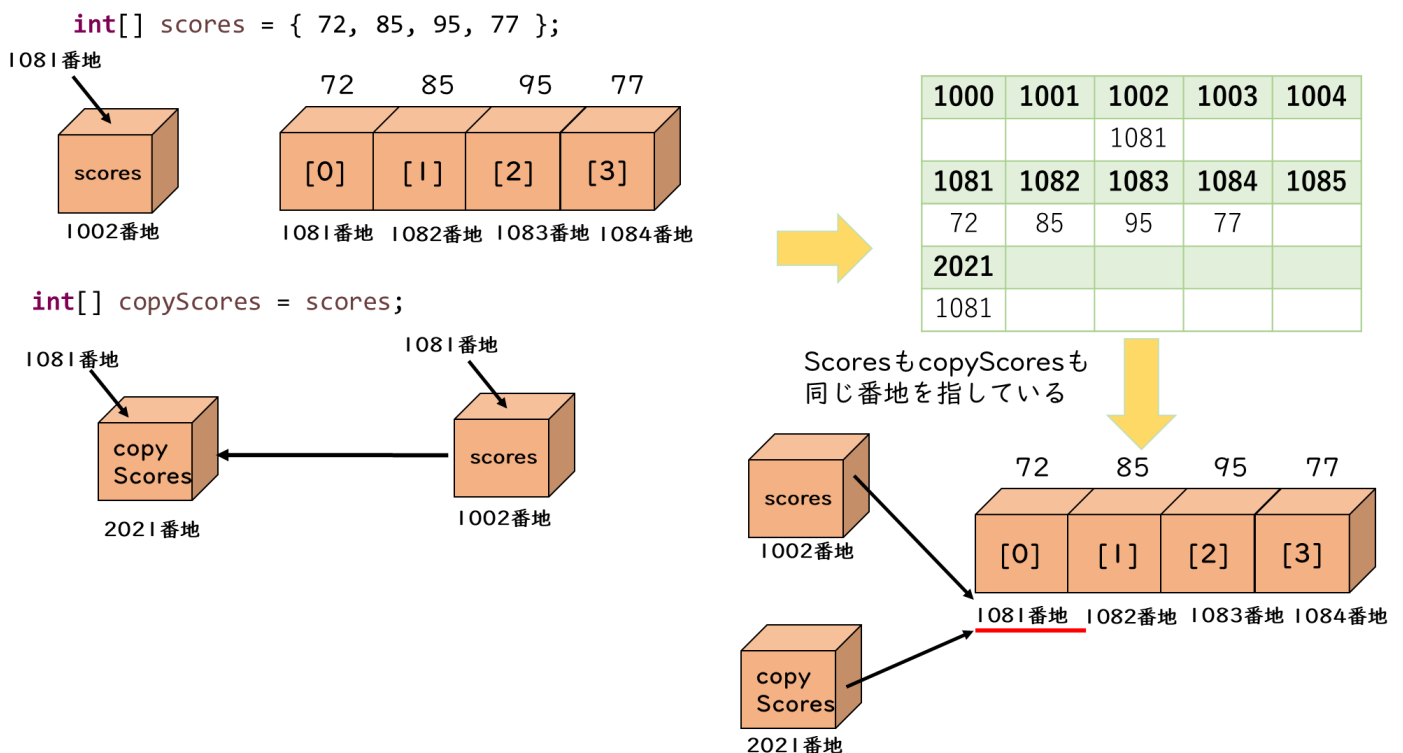
コンソール ×

<終了> Main (24) [Java アプリケー  
scores[0]の値:100  
copyScores[0]の値:100

で

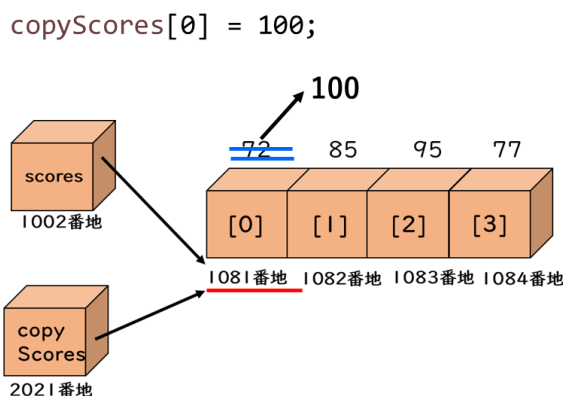
は、本題だ！

ここでは結果がどうしてこうなったのか、先ほどの参照の考え方を踏まえて考えていこう



状況を確認していくと、「scores」も「copyScores」も同じ 1081 番地を参照している状態つまり 2 つとも同じところを見ている状態になってしまうんだ。そうするとどうなるかみんなはもう想像がつくよね。

同じところをみてるのだから、値が変更になったら scores も copyScores の値も両方とも変わってします。だから実行結果が両方とも 100 になるんだ。



実行結果

scores[0]の値:100  
copyScores[0]の値:100

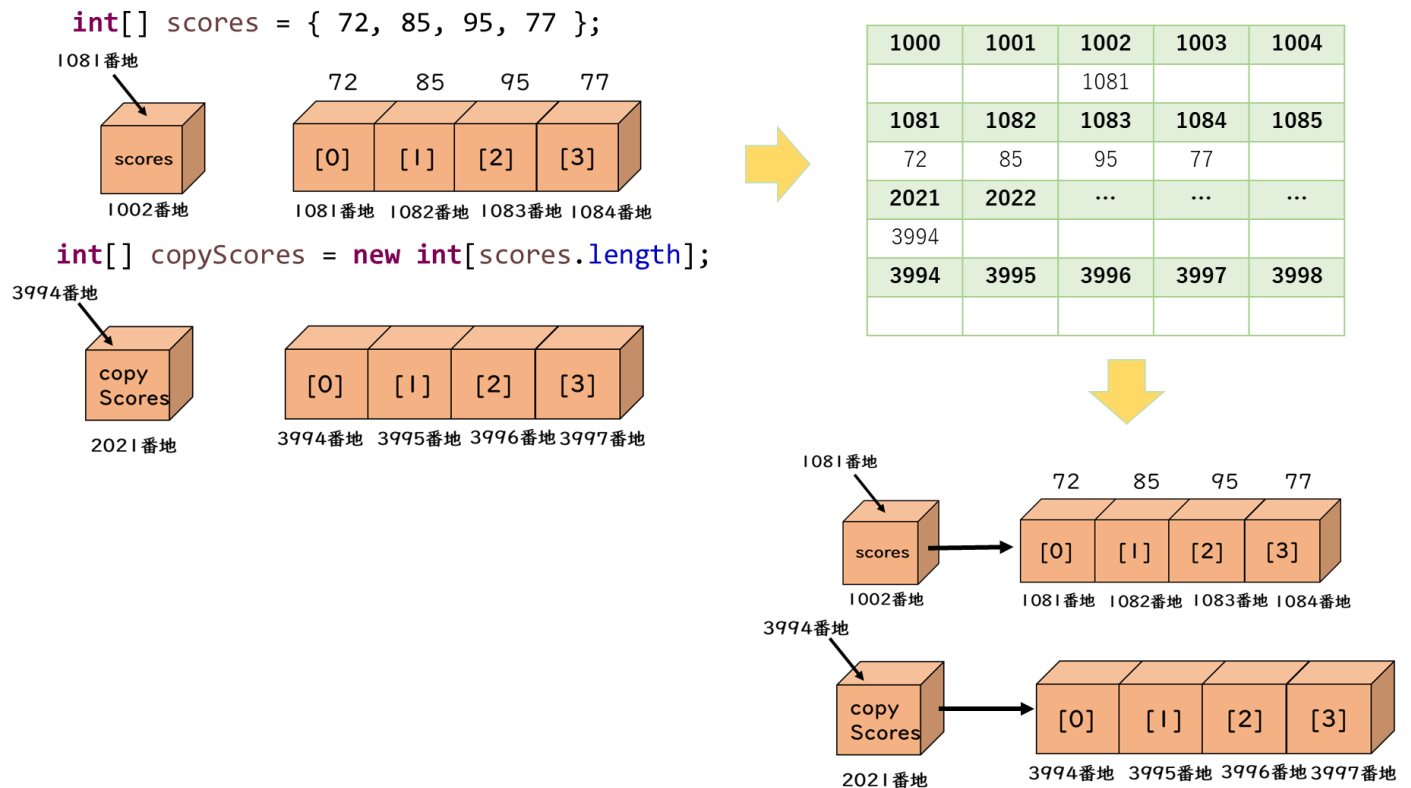
なるほど。だから同じ値が出力されたのですね。正直何か壊してしまったのではないかと焦りました。ですが、新たな疑問何ですが、本当の意味でコピーする。つまり、scores と copyScores が同じ番地を見ないでコピーするにはどうすればよいのでしょうか？

確かにその考え方は大事だね!では、ソースコードともに一緒に見ていこう!

先にどういうソースを書くのかを確認していくよ!下のソースと実行結果を見てみよう!

本当だ!正しくコピーできています。でもどうしてもかわからないです。

確かにこの部分の考えは結構難しいから一緒に一つずつ見ていこう!



なるほど、この「scores.length」の部分は、コピーする要素の数を指定しているんですね。

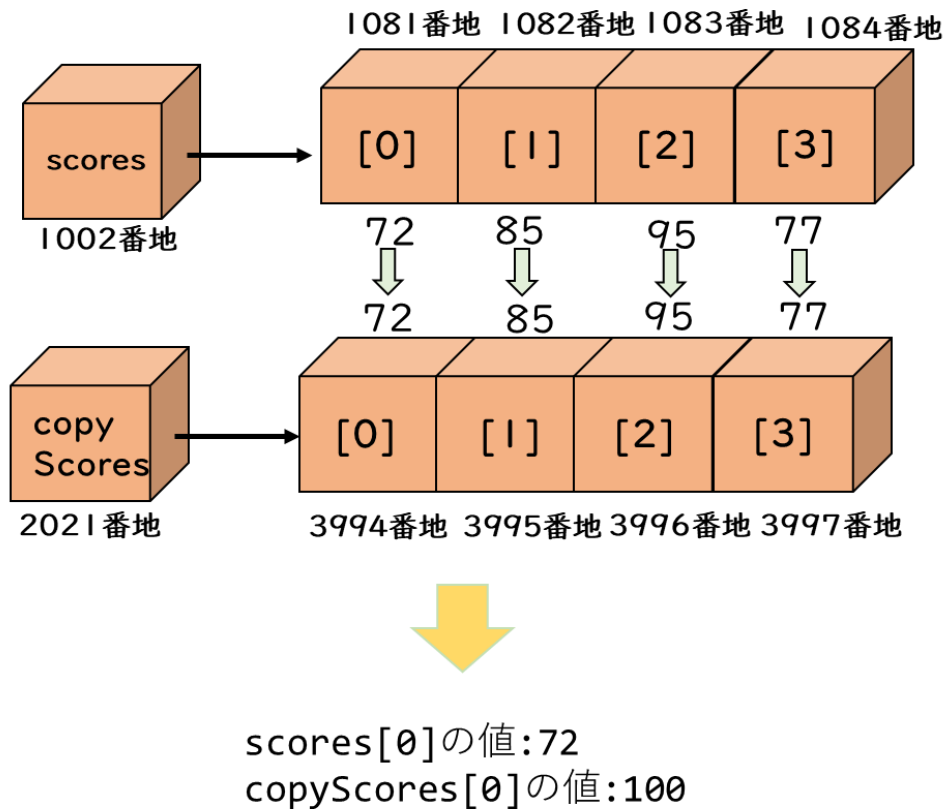
今回、scores の内容をコピーしたいから、scores の要素の数を取得するために、「scores.length」を書くという書き方を初めて知りました。こういう使い方もあるのですね！

ただ、要素の数をコピーする。つまり入れ物は用意できましたけど、このままだと入れ物を用意しただけで、値がコピーされていないと思うのですが、どうやって値をコピーするのですか？

きちんと、理解が進んでいるね。今回きちんと値をコピーするのにつかわれるのが for 文なんだ。

この部分も一緒に見ていこう！

```
for(int i = 0; i<scores.length; i++) {
    copyScores[i] = scores[i];
}
```



なるほど、コピー元 (scores) のインデックス番号を指定して値を取得した後、コピー先の (copyScores) のインデックス番号を指定して値を代入しているのでね！

その通り！

でこのコピーには実は名前があるんだ

最初に紹介したコピー方法の場合、

コピー先がコピー元と同じ番地を参照するようにコピーする方法を「シャローコピー」

コピー先とコピー元が違う番地を参照していてなおかつ値もコピーできている状態を「ディープコピー」

という言い方をするからぜひ知っておこう！

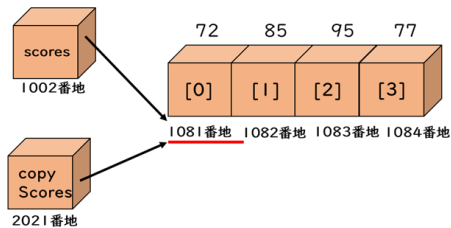
ごちゃごちゃしてしまうので申し訳ないのですが、図でまとめてもらってもいいですか？

もちろん！じゃあ図でまとめて頭の中を整理整頓しよう！

### シャローコピー

コピー元とコピー先が同じデータを参照している状態

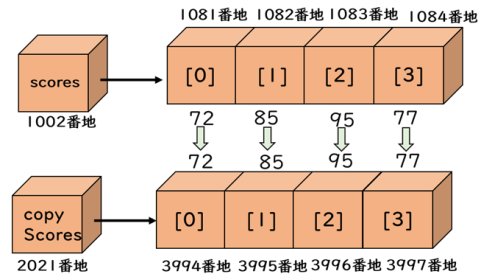
```
int[] scores = { 72, 85, 95, 77 };  
int[] copyScores = scores;
```



### ディープコピー

コピー元とコピー先が異なるデータを参照している状態

```
int[] scores = { 72, 85, 95, 77 };  
int[] copyScores = new int[scores.length];  
for(int i = 0; i < scores.length; i++) {  
    copyScores[i] = scores[i];  
}
```



他に先ほど習った拡張 for 文でもディープコピーをすることができるから参考までだけどこんな書き方もあるんだな  
ということを知っておく良いかな！

```
1 public class Main {  
2  
3     public static void main(String[] args) {  
4         // 大学のテストの点数を格納する配列を作成  
5         int[] scores = { 72, 85, 95, 77 };  
6         // scores配列の内容をコピーして保持するcopyScoresを宣言して要素を作成する。  
7         int[] copyScores = new int[scores.length];  
8         // copyScoresのindex番号（添え字）を設定  
9         int index = 0;  
10        // 拡張for文を使用してコピーする  
11        for (int score : scores) {  
12            copyScores[index] = score;  
13            ++index;  
14        }  
15        // copyScoresのインデックス番号0に100を代入する  
16        copyScores[0] = 100;  
17        // scoresのインデックス番号0の内容を表示  
18        System.out.println("scores[0]の値:" + scores[0]);  
19        // copyScoresのインデックス番号0の内容を表示  
20        System.out.println("copyScores[0]の値:" + copyScores[0]);  
21  
22    }  
23 }
```