What is a copy?

Copy of a variable

• an independent variable that contains the exact value of the original variable

• Two variables are independent of each other if their values can change independently of each other

See Copy1.java
How NOT to copy an array

Often, in a program, you need to duplicate an array or a part of an array. In such cases you could attempt to use the assignment statement (=), as follows:

\[ \text{list2} = \text{list1}; \]
Output of Copy1a.java

<table>
<thead>
<tr>
<th>a array:</th>
<th>Values after updating the copy b[0]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>a array:</td>
</tr>
<tr>
<td>2.2</td>
<td>11.1</td>
</tr>
<tr>
<td>3.3</td>
<td>2.2</td>
</tr>
<tr>
<td>b array:</td>
<td>3.3</td>
</tr>
<tr>
<td>1.1</td>
<td>b array:</td>
</tr>
<tr>
<td>2.2</td>
<td>11.1</td>
</tr>
<tr>
<td>3.3</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>3.3</td>
</tr>
</tbody>
</table>
What happened and how do we avoid it?

The value a[0] in the original array a is changed !!!
In fact: the value a[0] is the same as the value b[0]

Conclusions:
The array element identified by the name a[0] is the same array element that is identified by the name b[0]
The statement b = a; did not make a (true) copy of an array

When creating the copy, we must make space for a brand new array.
Copying arrays

Using a loop:

```java
int[] sourceArray = {2, 3, 1, 5, 10};

int[] targetArray = new int[sourceArray.length];

for (int i = 0; i < sourceArrays.length; i++)
    targetArray[i] = sourceArray[i];
```
The copyarray Utility

arraycopy(sourceArray, src_pos, 
           targetArray, tar_pos, length);

Example:

System.arraycopy(sourceArray, 0, 
                 targetArray, 0, sourceArray.length);
Declaring and Allocating space for a 2 dimensional array

(same as a one-dimensional array)

Step 1: Define an array object reference variable referring to a 2-dimensional array

```java
double[][] a;
// double[][] means:
// a reference (location) of a 2 dim. array
```

Step 2: Allocate space the (2-dimensional) array and store the location of the first element of the array in the (array) object reference variable

```java
a = new double[3][4];
// new double[3][4] creates a 3x4 array:
// an array with 3 rows and 4 columns
```
Initializing a 2D Array

Is very similar to the syntax used to define an initialized one-dimensional array.
The initial value are separated by nested { ... }

See TwoDimArray1.java
Using elements of a 2D Array

In general, using an n-dimensional array proceeds as follows:

• Each array element of a n-dimensional array is a ordinary variable (This is true for a one-dimensional array and it is equally true for a two-dimensional array).

• An array element of a n-dimensional array consists of:
  – the name of the array reference variable
  – n array indices

• Therefore, an element of a two-dimensional array is specified as (used by typing):

  ArrayRefVariable [ index1 ] [ index2 ]