Arrays

One-Dimensional Arrays

- **One-dimensional array**
- A structured collection of components, all of the same type, that is given a single name; each component is accessed by an index that indicates the component's position within the collection

Array

- composite, structured
- homogeneous
- access by position

Declaration and Instantiation of Array

```
int[] numbers;
```

Type of elements | Brackets | Name: holds address of the array

Declare

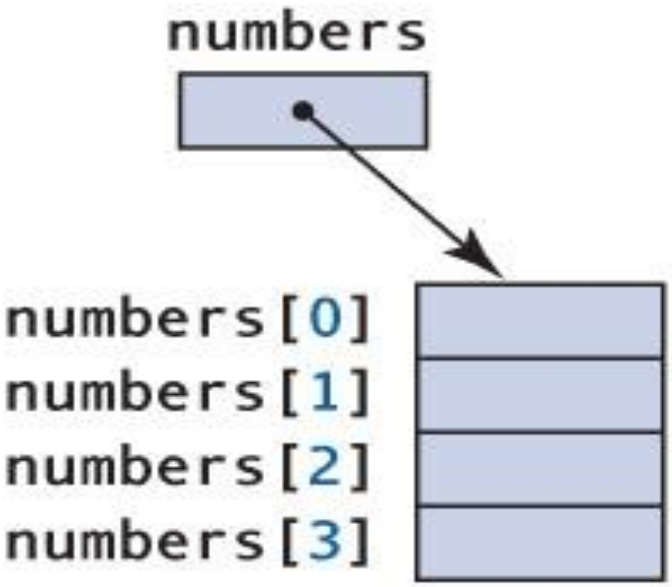
```
numbers = new int[4];
```

Variable name | Instantiation operator | Type of elements | Number of elements

Instantiate

One-Dimensional Arrays

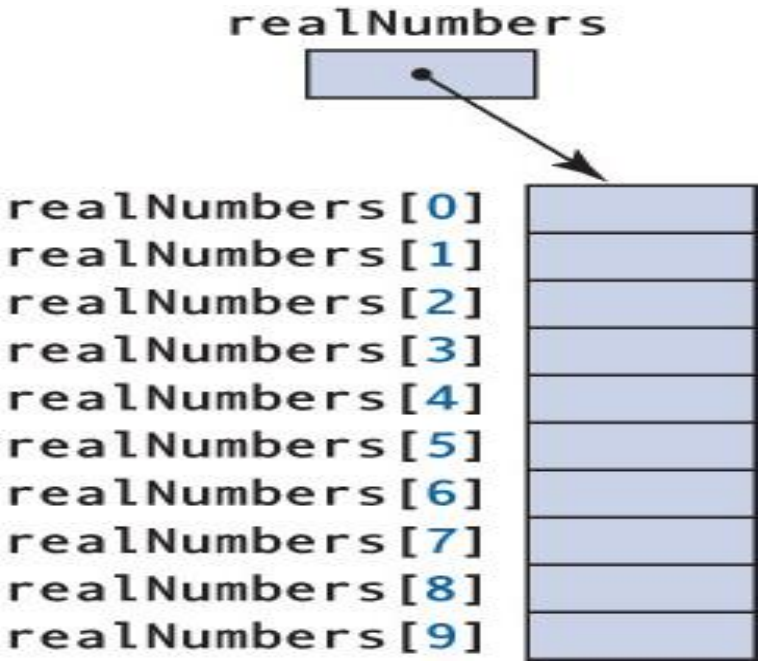
- **int[] numbers = new int[4];**



What type of values can be stored in each cell ?

One-Dimensional Arrays

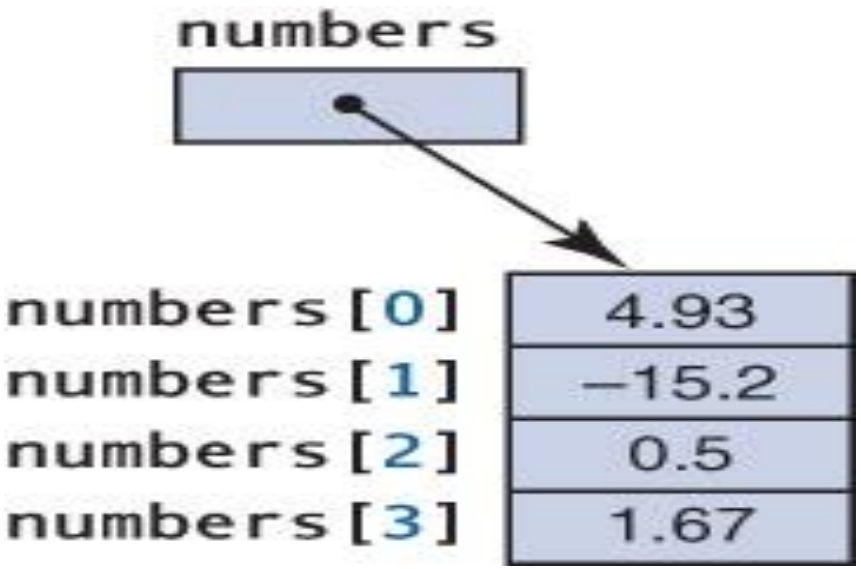
- **float[] realNumbers = new float[10];**



How do you get values into the cells ?

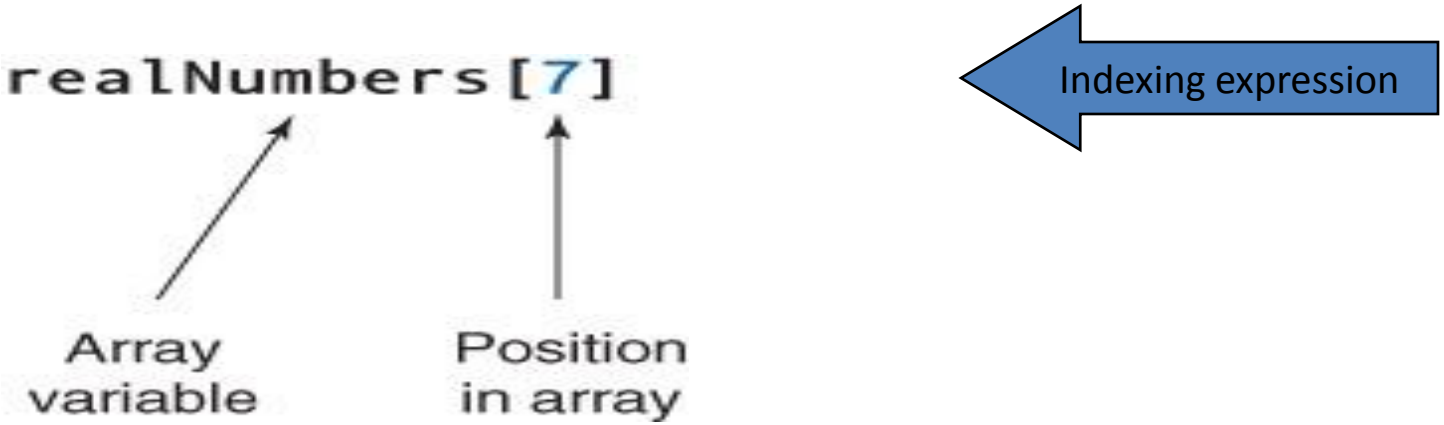
Array Initialization

- Array Initializers
- **int[] numbers = {4.93, -15.2, 0.5, 1.67};**

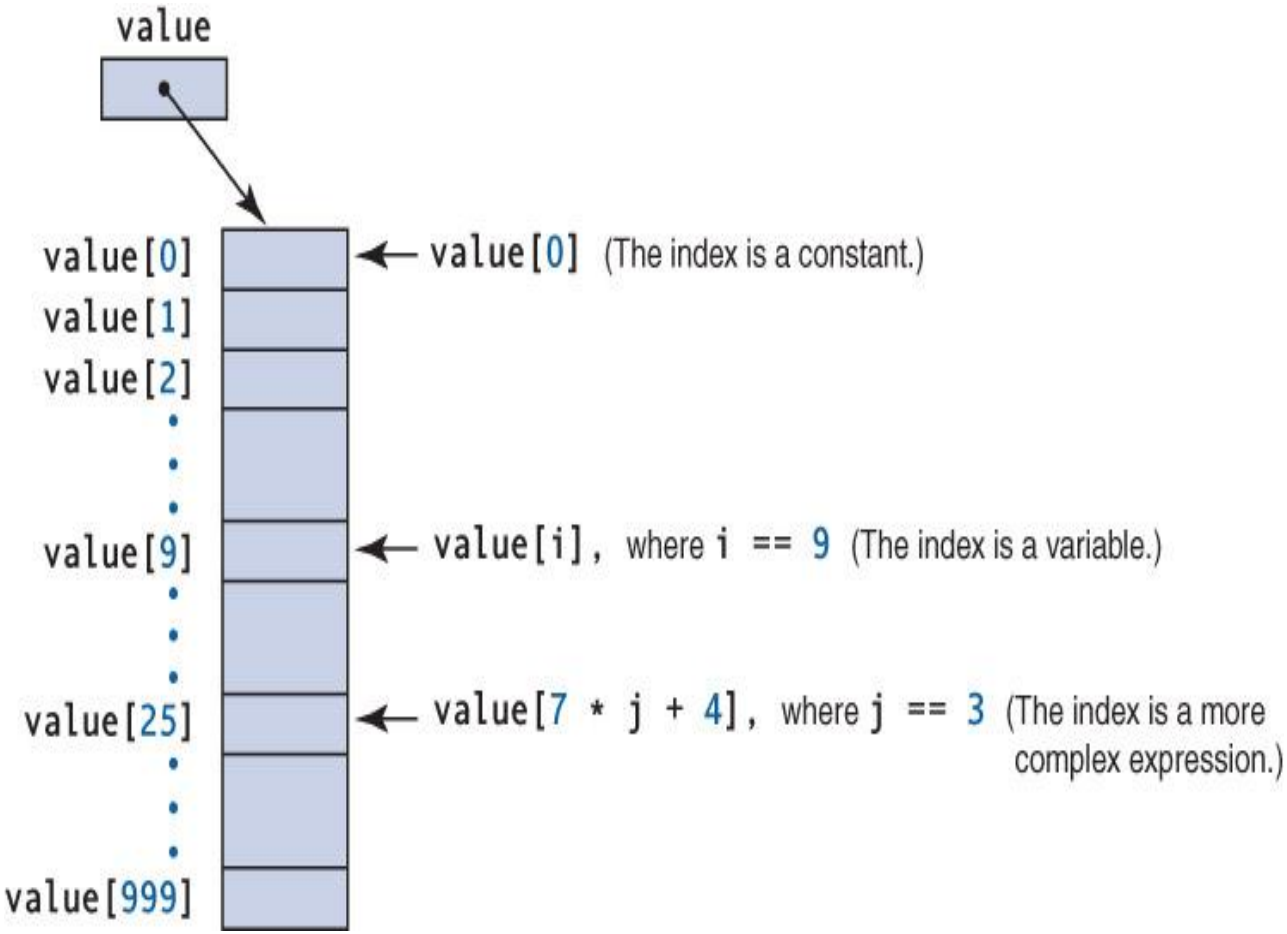


Initializers do the instantiation and storing in with the declaration

- Accessing Individual Components



One Dimensional Array



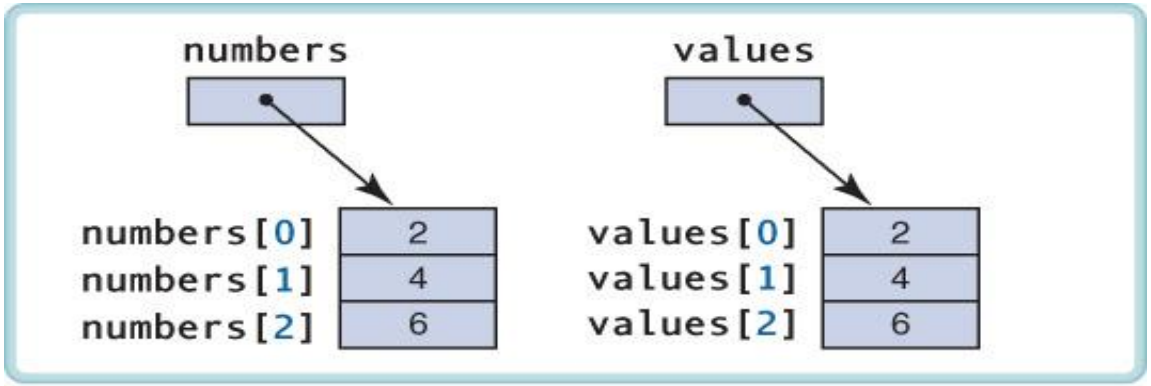
What happens if you try to access value[1000] ?

- **Out-of-bounds array index**
  - An index that is either less than 0 or greater than the array size minus 1, causing an ArrayIndexoutOfBoundsException to be thrown
- **Length**
  - A public instance variable associated with each instantiated array, accessed by

array name.length

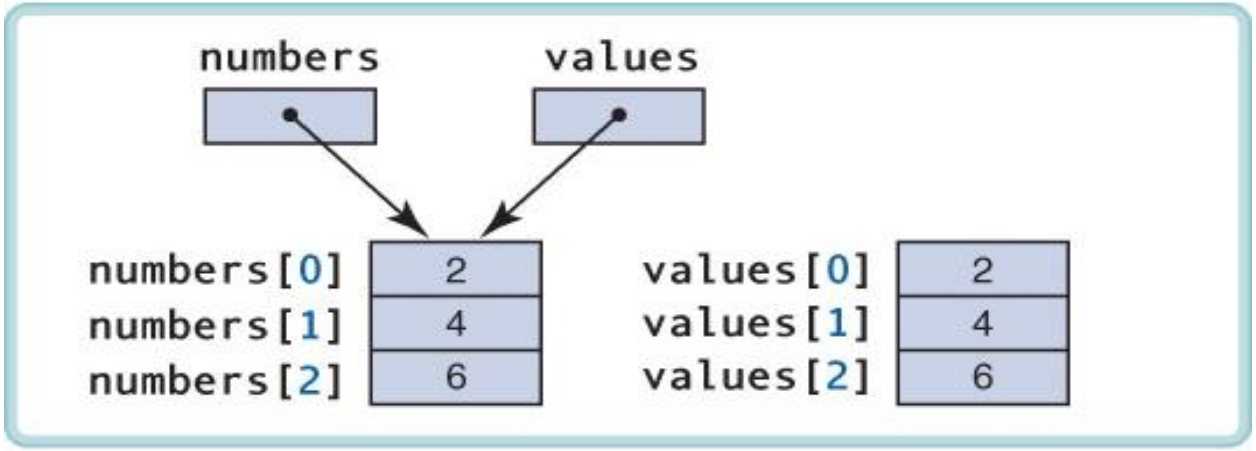
Use length to avoid out-of-bounds indexes

- Aggregate Array Operations



What does the following expression return?

numbers == values



Now, what does the following expression return?

numbers == values

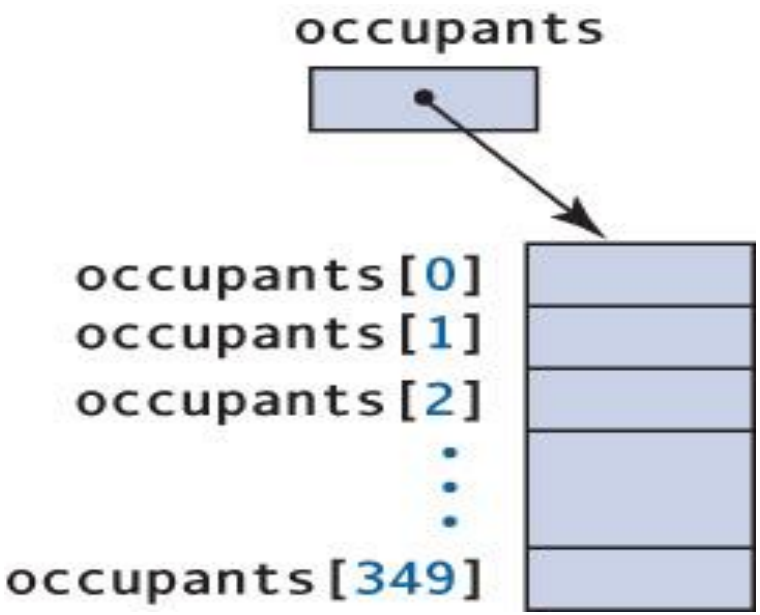
- System provides two useful array methods

```
first = second.clone(); // duplicates second import java.util.Arrays;
Arrays.equals(first, second); // item-by-item check

System.out.println(first == second);
System.out.println(Arrays.equals(first, second);
```

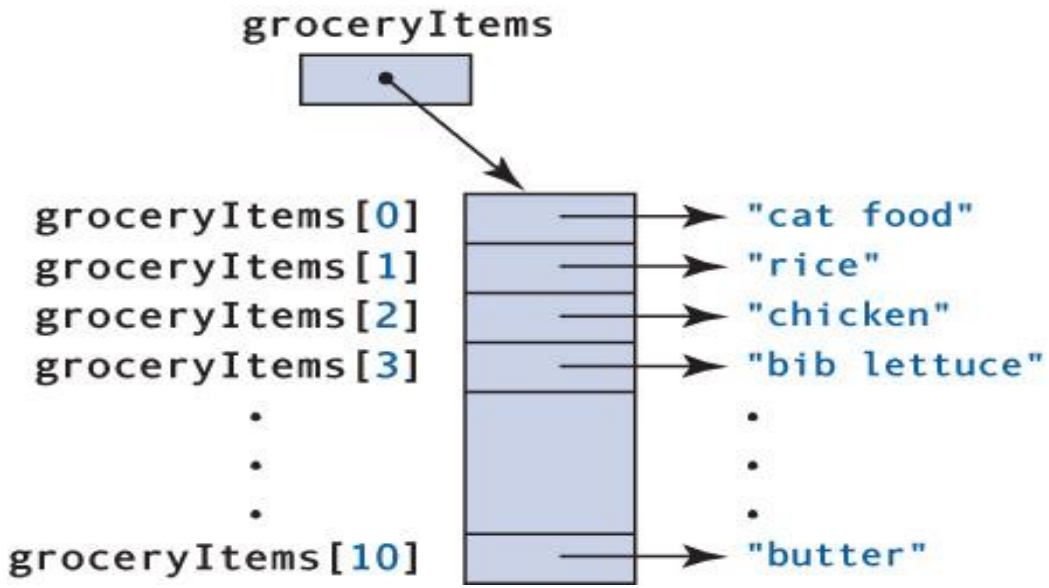
What is printed?

What does this code segment do?




```
totalOccupants = 0;
for (int aptNo = 0; aptNo < occupants.length; aptNo++)
  totalOccupants = totalOccupants +occupants[aptNo];
```

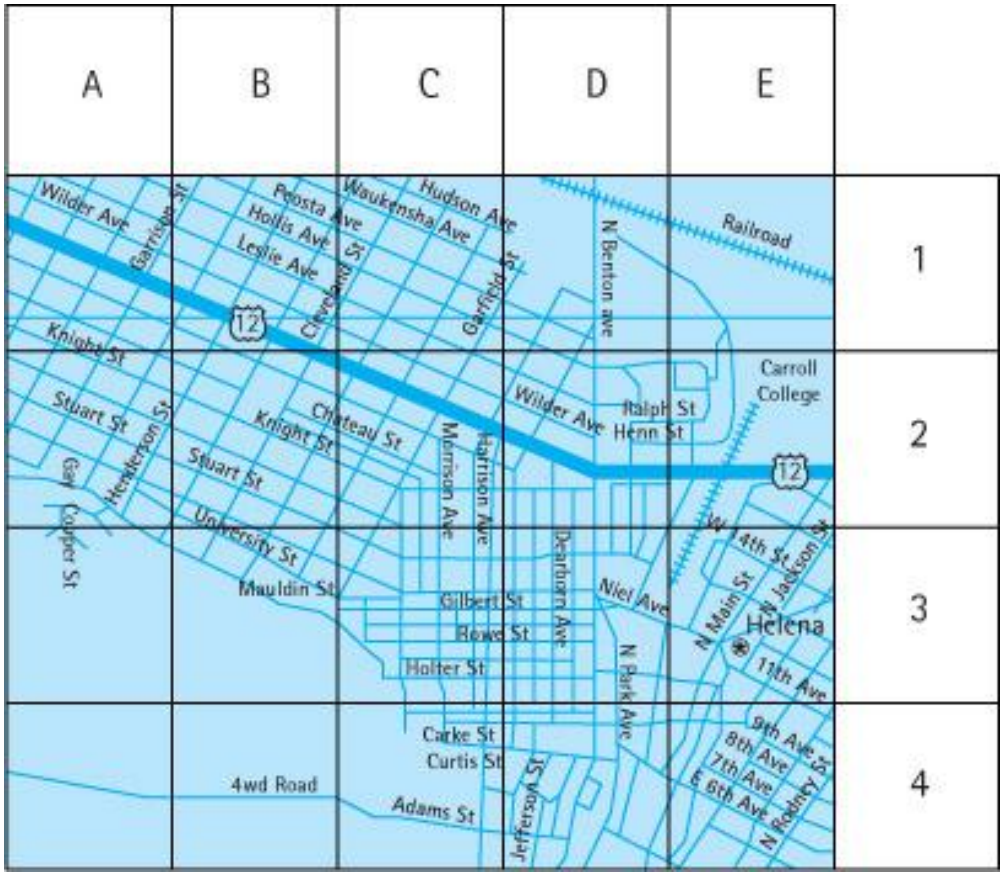
Array of Objects




```
String[] groceryItems = new String[10];
for (index = 0; index < grocerItems.length;  index++)
{
  groceryItems[index] = inFile.nextLine();
}
```

- **length** is the number of slots assigned to the array
- *What if the array doesn't have valid data in each of these slots?*
- Keep a counter of how many slots have valid data and use this counter when processing the array

Two Dimensional Array

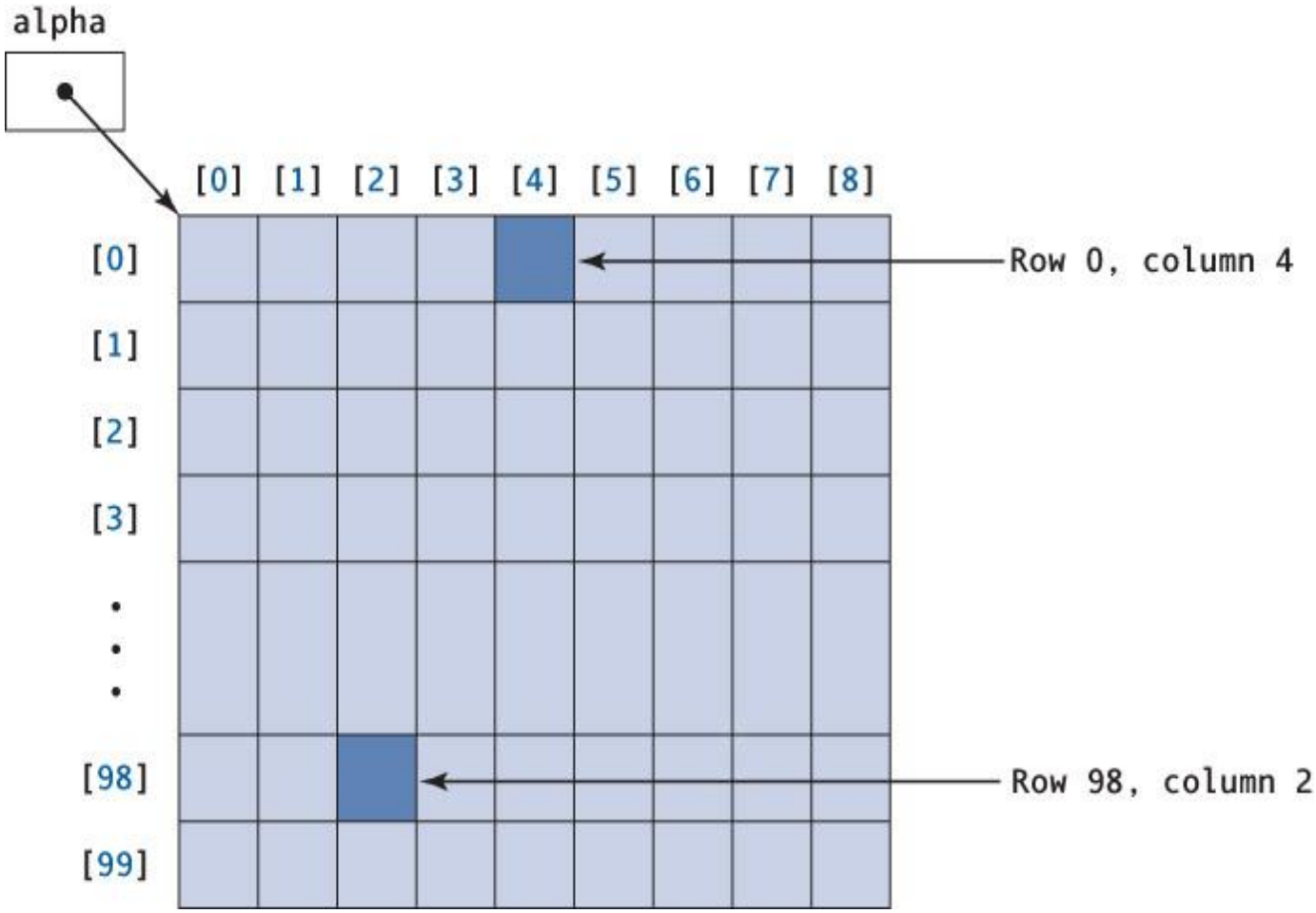


Two-dimensional arrays can be used to represent tables such as this map

Two-dimensional array

- A collection of homogeneous components, structured in two dimensions (referred to as rows and columns); each component is accessed by a pair of indexes representing the component's position within each dimension

Two-dimensional array



Declaration and Instantiation of 2D array

Declaration:

```
double[][] alpha;
```

Type of components

Brackets indicating array

Name of array

Instantiation:

```
alpha = new double[100][9];
```

Name of array

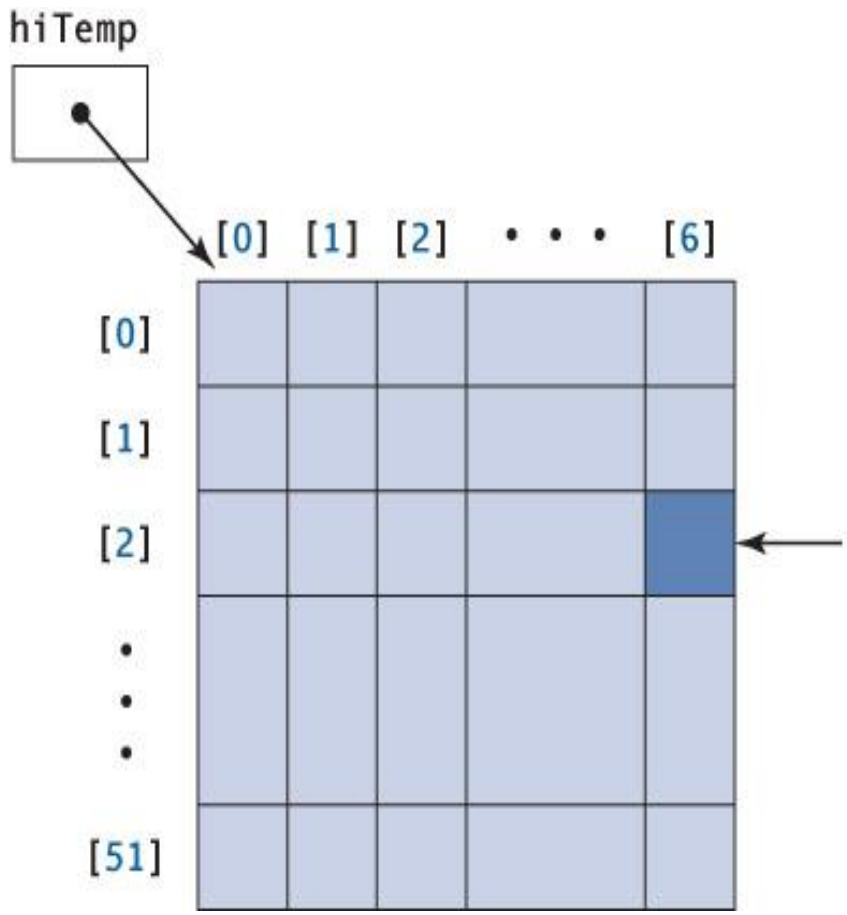
Instantiation operator

Type of components

Brackets with number of components for each dimension

Can you predict how each item is accessed?

Accessing the individual element in 2D array

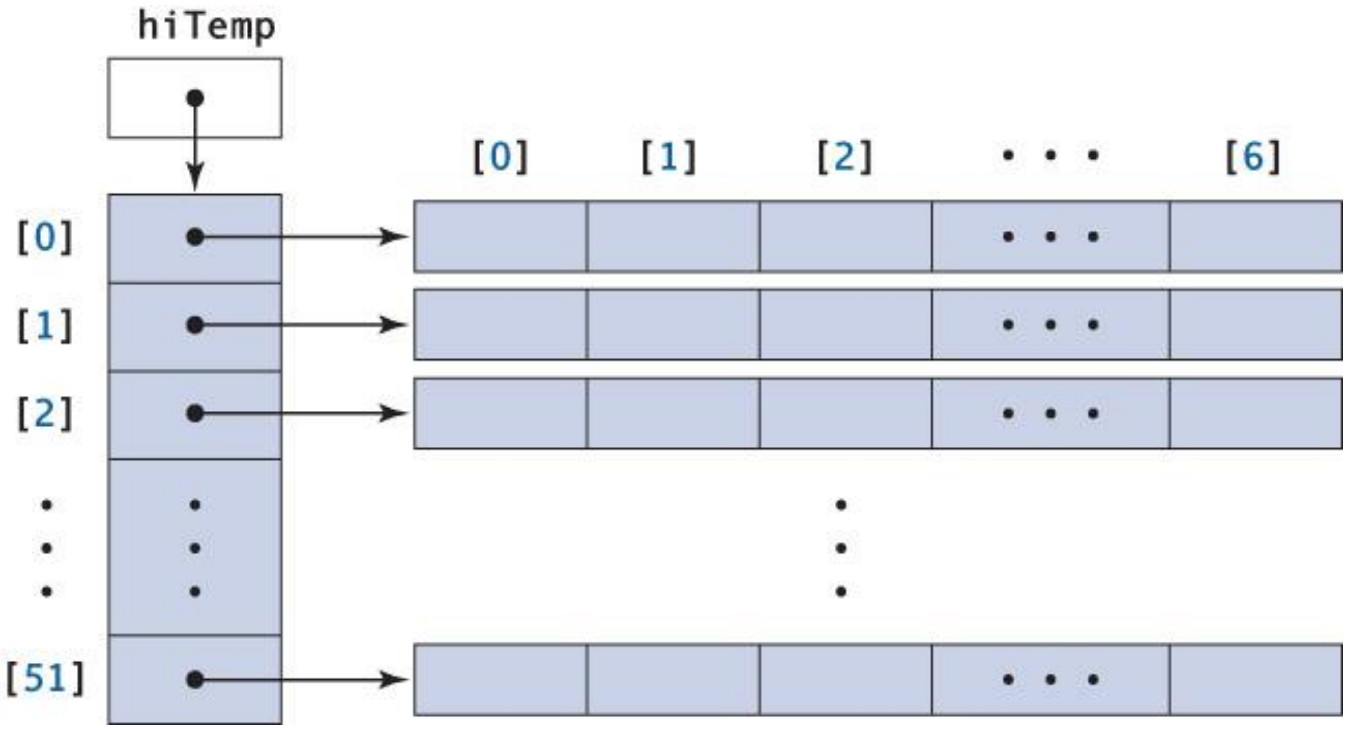


hiTemp[2] [6]

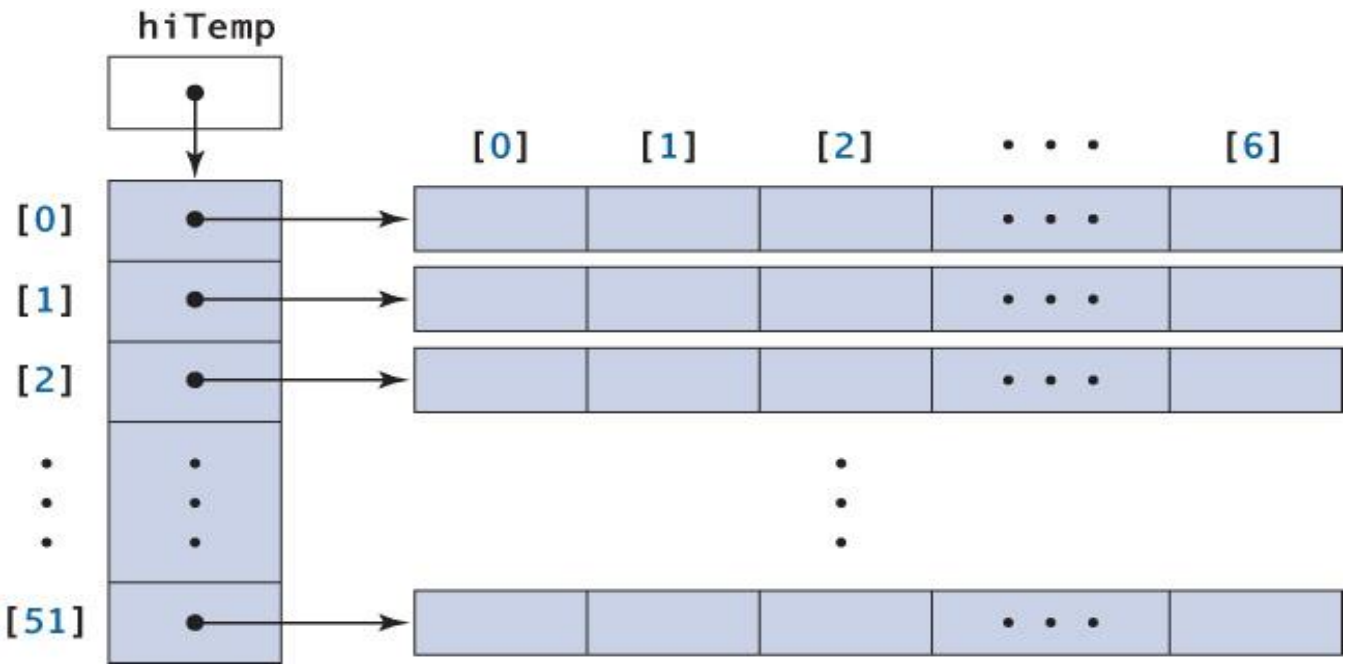
```
//Print temperature values
// for third week (row 2)

for (day = 0; day < 7; day++)
  outFile.printin(hiTemp[2] [day]);
```

Actual JAVA implementation



Actual Java implementation



hiTemp.length is the number of rows

hiTemp[2].length is the number of columns in row two

- *When processing by row,*
- *the outer loop is _____ (row, column)?*
- *the inner loop is _____ (row, column)?*

- *When processing by column,*
- *the outer loop is _____ (row, column)?*
- *the inner loop is _____ (row, column)?*

Variable Length array

- No of columns in each row are not same.
- Creating variable length array:
- **int[ ][ ] a;**
- **a=new int[10][ ];**
- If no of columns are not known then second subscript can be kept blank.
- Then to allocate memory for columns in each row
- **a[0]=new int[5];**

Initializing 2D array

Initializer Lists

```
int[][] hits  =    {{  2,  1,  0,  3,  2 },
                   {  1,  1,  2,  3 },
                   {  1,  0,  0,  0,  0 },
                   {  0,  1,  2,  1,  1 }};
```

- **To display the elements present in the an array :**

```
for(int i=0; i<hits.length; i++) //hits.length gives no of rows
for(int j=0; j<hits[i].length; j++) //hits[i].length gives no of columns in ith row
System.out.println("hits[" + (i+1) + "][" + (j+1) + "]=" +hits[i][j]);
```

Thank you