Java Arrays

Objectives

- Singledimensional arrays
- Multidimensional arrays
- Initializing arrays
- Accessing by index
- Array limitations
- Assigning elements to arrays.
- Iteration through arrays

Understanding Arrays

- An array is an area in the memory (in the heap) with space for a determined number of elements.
- String and StringBuilder are both using arrays. Arrays of char to be exact.
- You can make an array of both primitives and objects.

```
private final char value[];
```

Snippet from the String class

Understanding Arrays

- Created by adding the [ ] (brackets) after the type.

```
int number;    //Normal int
int[] numbers; //Array of ints
```

- Or created by adding the [ ] (brackets) after the name.

```
int numbers[]; //Array of ints
```

Initializing an array

Type of array

Bracket (Array symbol)

```
String[] stringArray = new String[3];
```

null	null	null

Size of array

Will create String array called ”stringArray” that has place for 3 elements.
The created array have all elements set with the **default value** of its type.

Initializing an array

```
int[] numbers = new int[] {34, 4, 12, 65};
```

Here we create an array of size 4.
Instead of specifying size we just set the initial values.

```
char[] word = {'J', 'A', 'V', 'A'};
```

Here is another shorter variance.

Accessing an array

Arrays are accessed with a technique called indexing.
First element in an array is always index 0;
Last element in an array is always index (length - 1)

```
String[] names = new String[3];
System.out.println(names.length); //3

names[0] = "Fredrik"; //Assign "Fredrik" to first element
names[2] = "Jonas";   //Assign "Jonas" to third element

System.out.println(names[0]); //Fredrik
System.out.println(names[1]); //null
System.out.println(names[2]); //Jonas
```

Java arrays are zero indexed!

Indexing example

```
int[] numbers = {5, 2, 3, 4, 7};
```

Index:	0	1	2	3	4
Element:	5	2	3	4	7

```
numbers[1] = 4;
```

Index:	0	1	2	3	4
Element:	5	4	3	4	7

```
numbers[4] = numbers[1];
```

Index:	0	1	2	3	4
Element:	5	4	3	4	4

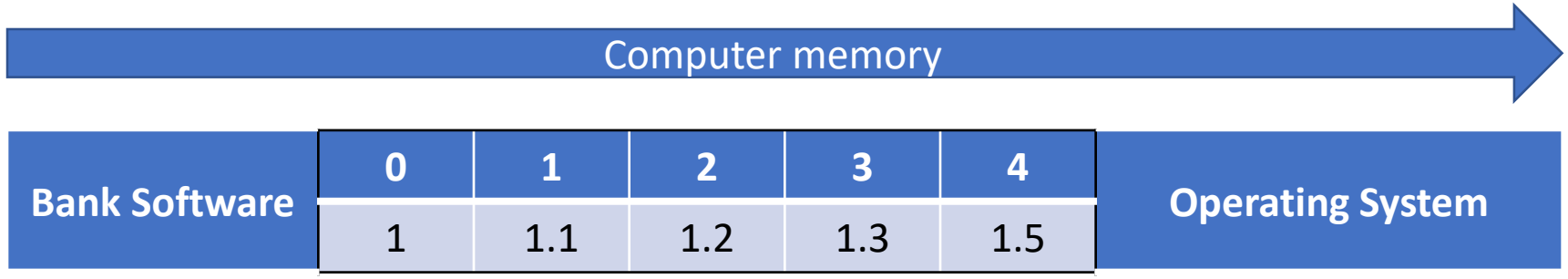
```
numbers[5] = 3;
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
at se.lexicon.examples.App.main(App.java:16)
```

Non Resizable

Java arrays size is always fixed. It is impossible to add 6 elements to an array that only has place for 5.

```
double[] numbers = {1, 1.1, 1.2, 1.3, 1.5};
numbers[5] = 1.6; //Causes ArrayIndexOutOfBoundsException
```



Java dont want to overwrite other systems variables by accident. It could potentially cause permanent damage.

Adding dimensions

- So far we only shown arrays with a single dimension.

Single dimensional array of 5 ints

```
int[] numbers = {5, 2, 3, 4, 7};
```

Index:	0	1	2	3	4
Element:	5	2	3	4	7

Two dimensional array of 5 arrays with 5 ints in each

```
int[][] numbers = new int[5][5];
```

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

That means each element of the array IS an array

Tic Tac Toe indexing

```
String[][] board = new String[3][3];

board[1][1] = "X";
board[0][0] = "O";
board[2][0] = "X";
board[0][2] = "O";
board[0][1] = "X";
board[2][1] = "O";
board[1][2] = "X";
board[1][0] = "O";
board[2][2] = "X";
```

	0	1	2
0	O	X	O
1	O	X	X
2	X	O	X

Iterating through arrays

- Iteration is a process of repeating operations that successively get closer to a desired result.
- Iteration we can iterate through an array in in several ways:
  - Index based for loop.
    - Can iterate forward and backwards or however you want with indexing.
  - Enhanced for loop.
    - Can only iterate from start to finish.

Iteration example with index based loop

```
int[] numbers = new int[10];

/* Iteration 1:  i=0 -> numbers[0] =  0 + 1 -> numbers[0] is 1
 * Iteration 2:  i=1 -> numbers[1] =  1 + 1 -> numbers[1] is 2
 * Iteration 3:  i=2 -> numbers[2] =  2 + 1 -> numbers[2] is 3
 * Iteration 4:  i=3 -> numbers[3] =  3 + 1 -> numbers[3] is 4
 * Iteration 5:  i=4 -> numbers[4] =  4 + 1 -> numbers[4] is 5
 * Iteration 6:  i=5 -> numbers[5] =  5 + 1 -> numbers[5] is 6
 * Iteration 7:  i=6 -> numbers[6] =  6 + 1 -> numbers[6] is 7
 * Iteration 8:  i=7 -> numbers[7] =  7 + 1 -> numbers[7] is 8
 * Iteration 9:  i=8 -> numbers[8] =  8 + 1 -> numbers[8] is 9
 * Iteration 10: i=9 -> numbers[9] =  9 + 1 -> numbers[9] is 10
  */
for(int i=0; i<numbers.length; i++) {
    numbers[i] = i+1;
}
```

Iteration example enhanced for loop

```
public class App {

    private String[] names = {
        "Nisse", "Olle", "Erik", "Simon", "Sofia", "Selma", "Ulf", "Fredrik"
    };

    public String findName(String nameToFind) {
        for(String name : names) {
            if(name.equalsIgnoreCase(nameToFind)) {
                return name;
            }
        }
        return "Error: No match";
    }

    public static void main( String[] args ){
        App app = new App();
        System.out.println(app.findName("Erik"));  //Prints out Erik
        System.out.println(app.findName("Roger")); //Prints out Error: No match
    }
}
```

Interating through a two dimensional array

```
int[][] numbers = new int[5][5];

numbers[0][0] = 1; //Setting the first int in the first array to 1

//Example of printing out a two dimensional int array
for(int i=0; i<numbers.length; i++) {
    for(int j=0; j<numbers[i].length; j++) {
        System.out.print(numbers[i][j]);

        //Blankline when index of subarray is at last element
        if(j == numbers[i].length -1) {
            System.out.println();
        }
    }
}
```

Will print →

```
10000
00000
00000
00000
00000
```

Interating through a two dimensional array

```
int[][] numbers = new int[5][5];

numbers[0][0] = 1;

    for(int[] array : numbers) {
        int index = 0;
        for(int number : array) {
            System.out.print(number);
            if(index == array.length -1) {
                System.out.println();
            }
            index++;
        }
    }
```

Will print →

```
10000
00000
00000
00000
00000
```

Questions?