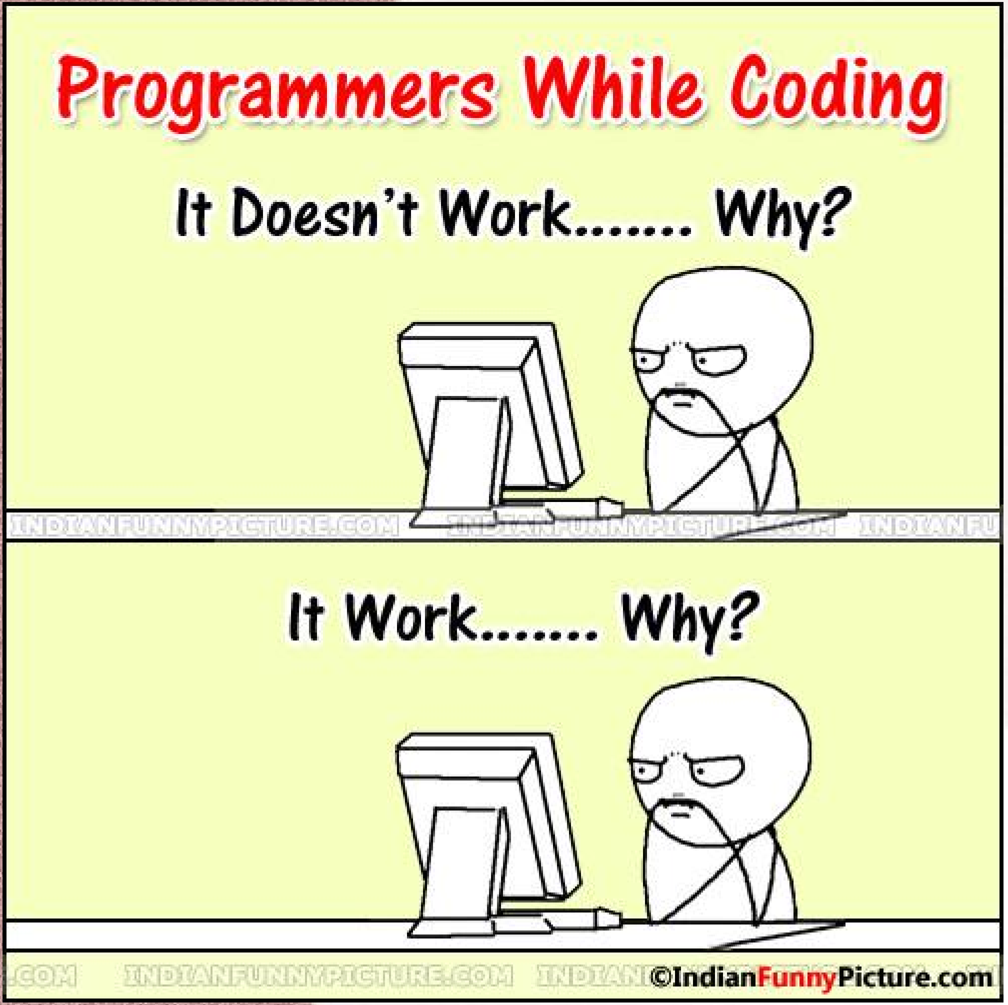
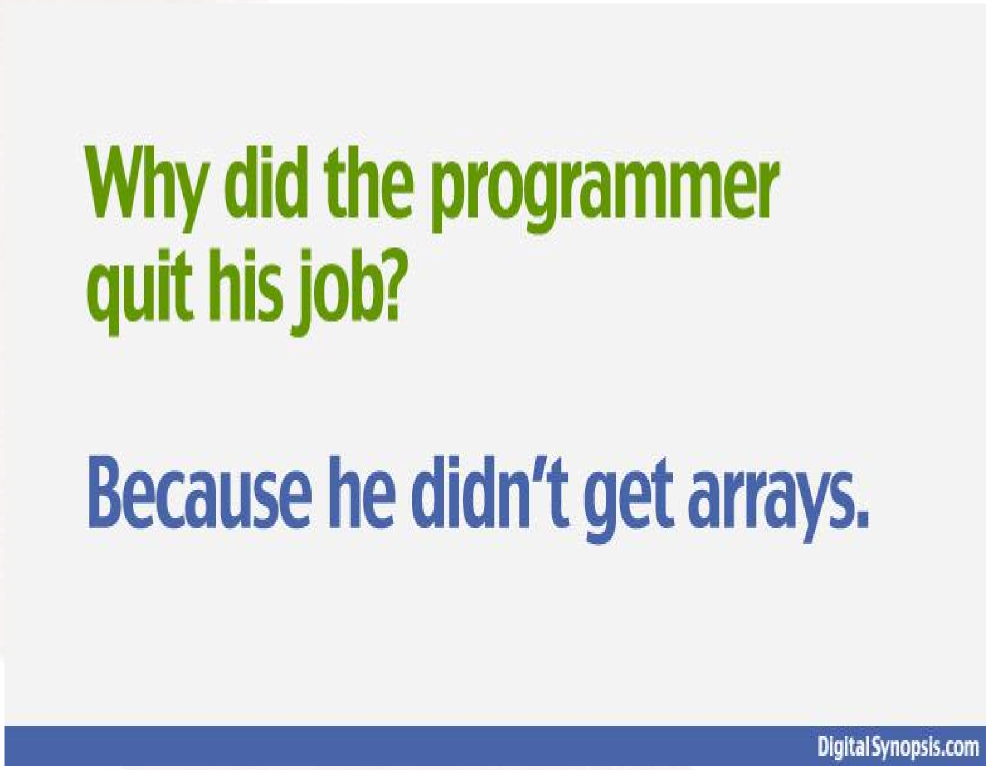
Chapter 5 – Array



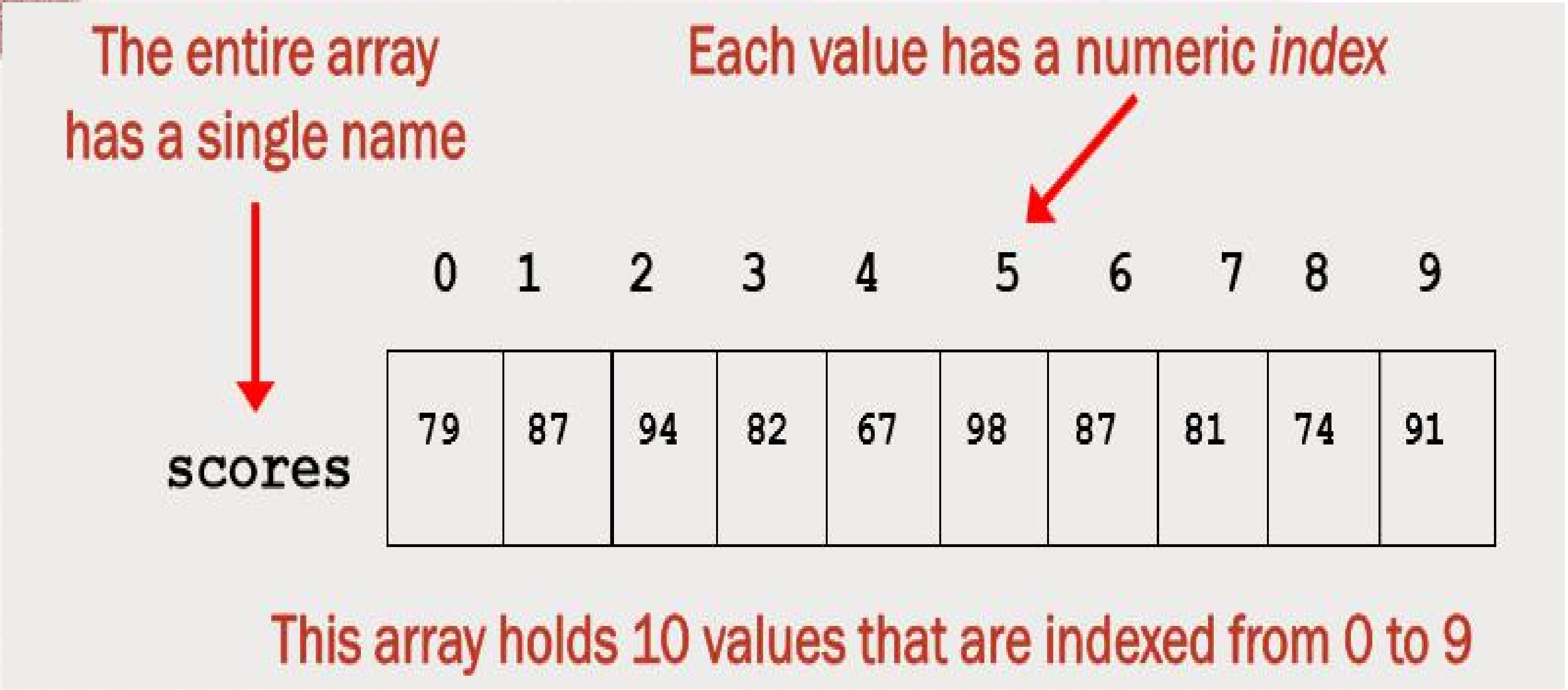


Why did the programmer quit his job?

Because he didn't get arrays.

DigitalSynopsis.com

What is an array?


The entire array has a single name
Each value has a numeric *index*
0 1 2 3 4 5 6 7 8 9
scores
79 87 94 82 67 98 87 81 74 91
This array holds 10 values that are indexed from 0 to 9

Note: we can think of index as an allocation memory or different mailbox

The size of the array goes from 0 to length-1 and in our case "score" has a length of 10

Why do we need to use arrays?

For instance if we wish to store 200 values, are we really going to make 200 different variable names?

Answer: NO! We will declare an array and store the 200 values

Declaring & Creating an array

Declare reference

Syntax:

```
DataType [] nameArray;
```

There are different type of arrays: Primitive type and Objects reference

Create elements:

Syntax:

```
NameArray = new dataType[size];
```

Declaration + Creation:

Size must be an integer

Syntax:

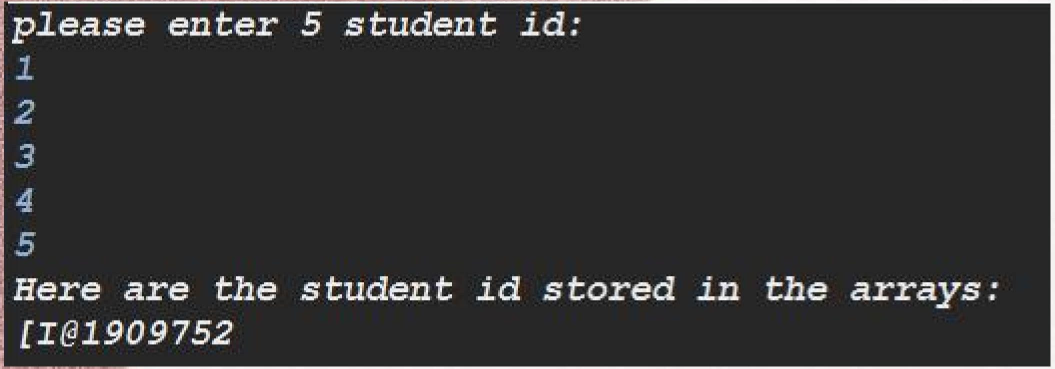
```
DataType [] nameArray = new dataType[size];
```

Example of array

```java
public class arrays {

public static void main(String[] args){
    Scanner kb = new Scanner(System.in);
    int userId;
    double userGrade;

    //declare the reference
    int [] studentId ;
    //creating the elements
    studentId = new int [5];
    //declaration + creation
    double [] grades = new double[5];

    System.out.println("please enter 5 student id: ");
    for(int i =0; i<5; i++)
    {
    userId = kb.nextInt();
    studentId[i] = userId;
    }

    System.out.println("Here are the student id stored in the arrays:");

    System.out.println(studentId);
    }
}
```

What's the output of the following code?

Example of array - answer



```
please enter 5 student id:
1
2
3
4
5
Here are the student id stored in the arrays:
[I@1909752
```

Will only print the reference of the array and not what's the content of it

If we wish to print the contents of the arrays, then we will need to add a for loop to go through each index

```
System.out.println("Here are the student id stored
in the arrays:");

for(int i =0; i <5; i++)
System.out.println(studentId[i]);
```

Initialization of arrays

All arrays are initially initialize to default

- Int,double = 0
- Boolean = false
- reference(object) = null

Initialization of arrays

We can initialize an array manually

For example:

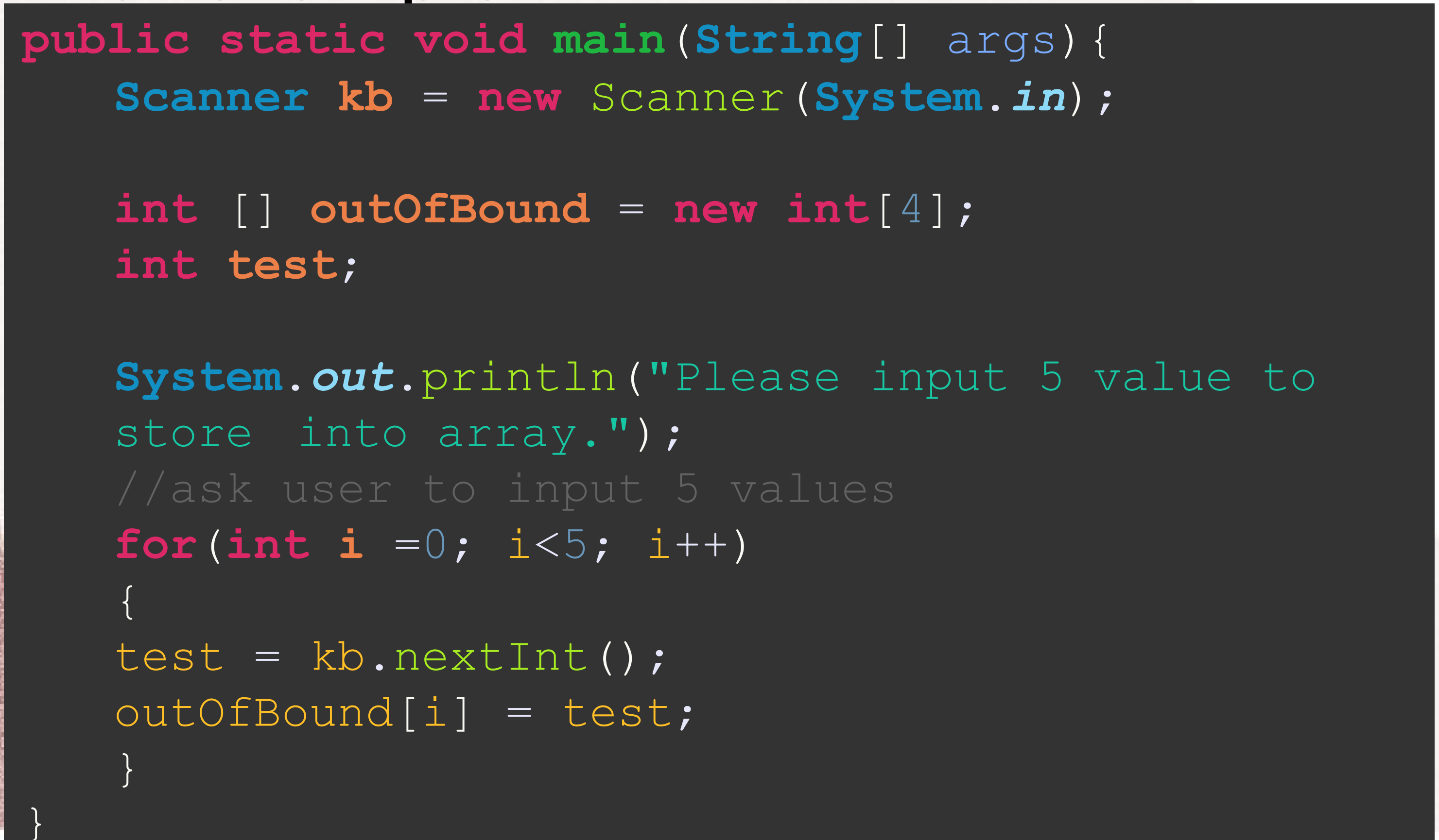
```
Int[] price = { 10, 20, 44, 52, 62 };
String [] letter = {“hi”, “okay”, “um”, “bye”};
```

length of
“price” is 5

Array out of bound

Sometime we need to watch out for the bound of the array, if we go out of bound then we will cause an error

For example:



```
public static void main(String[] args){
    Scanner kb = new Scanner(System.in);

    int [] outOfBound = new int[4];
    int test;

    System.out.println("Please input 5 value to
    store  into array.");
    //ask user to input 5 values
    for(int i =0; i<5; i++)
    {
    test = kb.nextInt();
    outOfBound[i] = test;
    }
}
```

This code segment will cause an out Of Bounds Exception

Multidimensional array

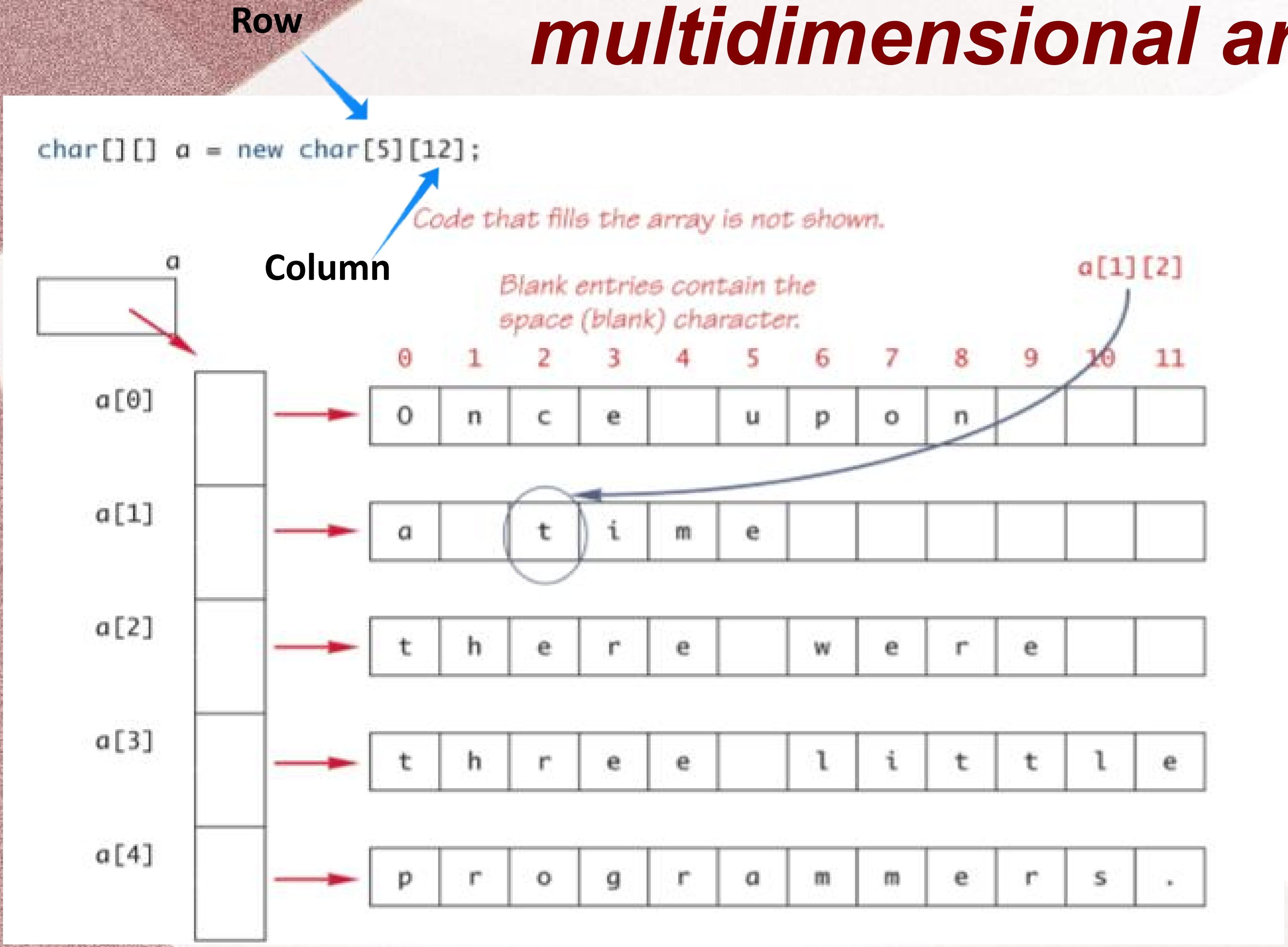
Array of an array, useful if we wish to have more than one index

Declaration and creation is the same as 1D array

Example:

int course = new int[4]; //1D, 4 test for students
int section = new int[4][50]; //50 students per section
Int course = new int[2][4][50]; // 4 section per course

Example of multidimensional array



Length of a multidimensional array

If we have a multidimensional array, the "**.length**" will not give the total number of index that the 1D does.

Here's an example how it work in 2D or more

Given the following code, what's the output?:

```java
int[][] student = new int [2][];

student[0] = new int[12];
student[1] = new int[4];

System.out.println(student.length);
System.out.println(student[0].length);
System.out.println(student[1].length);
```

Ragged arrays

They are both equivalent

```
double [][] raggedArray = new
double[2][10];
```

```
double[][] otherRagged;
otherRagged = new double[2][];
otherRagged[0] = new
double[10];
otherRagged[1] = new double
[10];
```

Note: the second version is longer than the first one, but both code are the same. The “otherRagged” leave an empty [] on second line so we can set it manually

More on ragged array

```
double[][] moreRagged = new double[3][];
moreRagged[0] = new double[15];
moreRagged[1] = new double[6];
moreRagged[2] = new double[8];
```

Note: since the first line of the variable "moreRagged" does not specify the size of a[0], a[1], and a[2]. We can create our own size