Test Automation Lecture 3 – Classes, fields and methods. Arrays.

PRAGMATIC IT Learning & Outsourcing Center

Lector: Milen Strahinski
Skype: strahinski
E-mail: milen.strahinski@pragmatic.bg
Facebook: http://www.facebook.com/LamerMan
LinkedIn: http://www.linkedin.com/pub/milen-strahinski/a/553/615

www.pragmatic.bg

Contents

- Object Oriented Programming (OOP)
- Classes and objects
- Fields
- Manipulating object state
- Using methods
- Introduction to Arrays

Object Oriented Programming

- OOP is concept in programming
- It enable software engineers to write reusable, easy for understanding and maintaining code
- The heart of OOP consist of objects and classes

Objects

- Software objects are used to model the real-world and abstract objects that you find in everyday life
- Real-world objects share two characteristics: They all have state and behavior

Each person has name, age, personal number... (state)

Each person can eat, sleep, walk... (behavior)

Mobile phone – Has memory, has color, is switched on or off. Can ring, can send SMS, can be switched off

Classes

Main idea

- The class acts as the template for building object
- The class defines the properties of the object and its behavior

Person example

Every human:

- Has name
- Has age
- Has personal number
- Has sex
- Has weight

Person example

Ivan

- 25 years old
- p.n. 8612025281
- is male
- 80.5 kg

Maria

- 21 years old
- p.n. 8203301201
- is female
- 55.0 kg

Writing simple classes

- Each starts with *class <name of the class>*
- The properties are called fields. They hold the state of each object
- The fields has type and name

```
public class Person {
        String name;
        int age;
        long personalNumber;
        boolean isWoman;
        double weight;
}
```

Class name

Fields

Objects in Java

- Objects are the presentation of a class
- Each class can have more than one object instances
- Objects of same classes have the same properties, but they may differ by the values of these properties
- Objects exists in heap memory
- Objects can be created and their state can be changed

Creating objects of class Person

- A variable of type Person should be declared
- Objects are created via constructors (we'll talk more about them in the next lesson)
- Using keyword *new*

```
public class PersonTest {

    public static void main(String[] args) {
        Person ivan = new Person();
        Person maria = new Person();
    }
}
```

Differences between classes and objects

- Object is the concrete representation of a class.
- Class is the „model“ for creating an object
- Each object has the properties that its class owns
- Objects have the same properties, but they may differ by the values of these properties
- One class can have more than one objects, but an object can't be instance of more than one classes

More on classes

- Each class begins with a capital letter and use CamelCase convention
- Each class has the same name as the file it is declared in
- The programmer creates the classes in a file .java, Java compiles .java-files and creates .classes
- .java is human-readable, .class is machine-readable

Accessing fields and modifying the state of the object

- <object>.<fieldname> is used to access fields

```
public static void main(String[] args) {

      Person ivan = new Person();
      ivan.name = "Ivan";
      ivan.age = 25;
      ivan.isWoman = false;
      ivan.personalNumber = 861202528;
      ivan.weight = 80.5;

      System.out.print("Ivan is " + ivan.age + "
years old ");
      System.out.print("and his weight is " +
ivan.weight);
}
```

Accessing field with .

Car Example

Let's write class which represents Car

Each car has:

- Max speed
- Current speed
- Color
- Current gear

Car Example

1. Write the class Car

2. Create class CarDemo with main method

3. Create 2 instances of class car and set values to their fields

4. Change the gear and current speed of one of the cars

Car driver/owner

- We want every car to have owner.
- The owner is a person

1. Make some changes to class Car to assign owner to every car

2. In CarDemo print to the console the name of the owner for every car n.

Add friend to class Person

- Each person has a friend, who is a person as well.
- Friend is a field of type Person in class Person.
- *There is no problem for a class to have an instance of itself*

Methods

- Methods are features of the object
- Can manipulate the data of a specified object
- Can perform any other task
- Have name
- Have body, enclosed between braces { } – code
- Have parameters
- Have return type (for now we'll use only void)

```
<return type> <method name> (<parameters>) {
        <body>
}
```

Methods in class Person

Each human eat food, can walk, can drink water and increase his age every year.

- eat ()
- walk()
- growUp() - modify the field age
- drinkWater(double liters)

Methods in class Person

```
public class Person {
        String name;
        int age;
        long personalNumber;
        boolean isWoman;
        double weight;

        void eat() {
          System.out.println("Eating...");
        }
        void walk() {
          System.out.println(name + " is walking");
        }
        void growUp() {
          age++;
        }
        void drinkWater(double liters) {
          if(liters > 1) {
            System.out.println("This is too much water!!!");
          } else {
            System.out.println(name + " is drinking " + liters + " water.");
          }
        }
}
```

Method name

Return type

Parameter

Calling methods

- (non static) methods are called by instance of the class using .
- *<instance>.<method name>(<parameters list>);*

```java
public static void main(String[] args) {
      Person ivan = new Person();
      ivan.name = "Ivan";
      ivan.age = 25;
      ivan.isWoman = false;
      ivan.personalNumber = 861202528;
      ivan.weight = 80.5;

      ivan.walk();
      double literWater = 0.3;
      ivan.drinkWater(literWater);
}
```

Exercise

- Add methods in class Car:

```
void accelerate()
void changeGearUp()
void changeGearDown()
void changeGear(int nextGear)
void changeColor(String newColor)
```

- Write logic in methods which change gear (validate the gear before changing - min is 1, max is 5)
- Invoke them in CarDemo class

Methods in class Car

```
void changeGearUp() {
        if(gear < 5) {
          gear++;
        }
}
void changeGearDown() {
        if(gear > 0 ) {
          gear--;
        } else {
          System.out.println("You are now on 1st gear!!!);
        }
}
void changeGear(int nextGear) {
        if(nextGear > 0 && nextGear < 6) {
          gear = nextGear;
        }
}
void changeColor(String newColor) {
        color = newColor;
}
```

Calling the methods of class Car

```java
public static void main(String[] args) {
          Car golf = new Car();
          golf.speed = 100;
          golf.color = "Red";
          golf.gear = 5;
          golf.maxSpeed = 320.5;

          Car honda = new Car();
          honda.gear = 5;
          honda.changeGearUp();

          System.out.println("The current speed of the golf is " + golf.speed);
          golf.accelerate();
          System.out.println("The current speed of the golf is " + golf.speed);

          System.out.println("The current gear is " + golf.gear);
          for (int i = 0; i < 10; i++) {
                    golf.changeGearUp();
          }
          System.out.println("The current gear is " + golf.gear);

          System.out.println("The Honda's current gear is " + honda.gear);
          honda.changeGear(1);
          System.out.println("The Honda's current gear is " + honda.gear);

          golf.changeColor("Blue");
          golf.changeColor("Red");
}
```

Problem

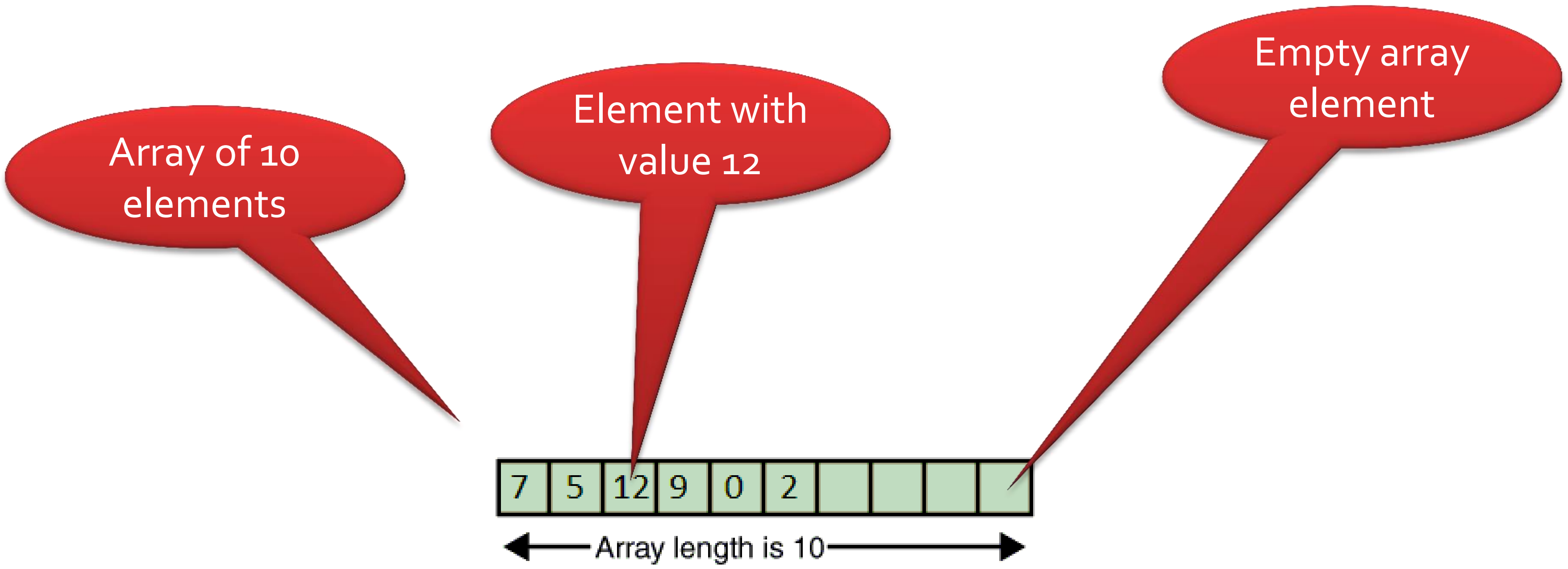
- Define more than one variable for similar purpose
- ***Example***:
  Grades of a student group – define 30 variable for them
- ***Solution***:
  Define 30 variable of type double to hold the information

Is this so rational?

What's an array?

- An array is a sequence of elements
- Arrays keep variables of only one type
- The order of the elements remains the same
- Arrays have a fixed length
- The access to the elements is direct
- The elements are accessed through an index

What's an array?



Declaration and initialization

- Declaration

array name(variable)

```
int[] array;
```

array type

```
int array[];
```

- Initialization

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

size

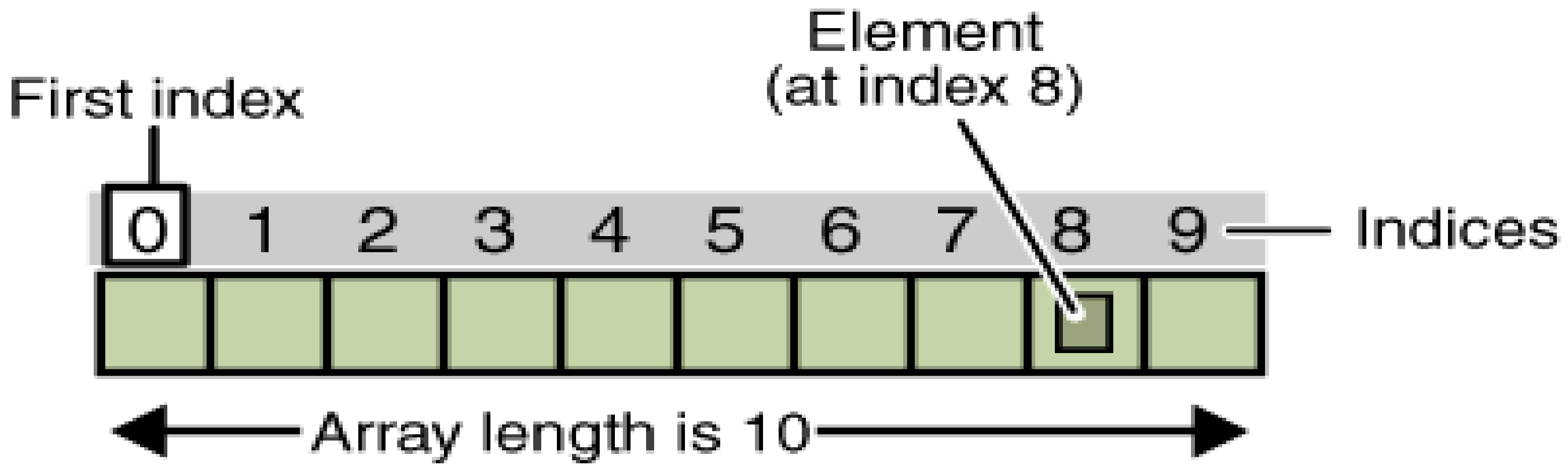
type

- Declaration and initialization

```
int[] array = new int[10];
int[] array = { 5, 7, -2, 12, 0, 4 };
```

Accessing the elements

- Elements are accessed by index
- The index of the first is 0
- The index of the last is equal the length – 1
- The elements can be read and changed



Accessing the elements

- array[i] returns the value of element with index i

```
System.out.println(array[0]);
//prints the value of the first array element

System.out.println(array[1]);
//prints the value of the second array element


array[2] = 100;
//changes the value of the third element
```

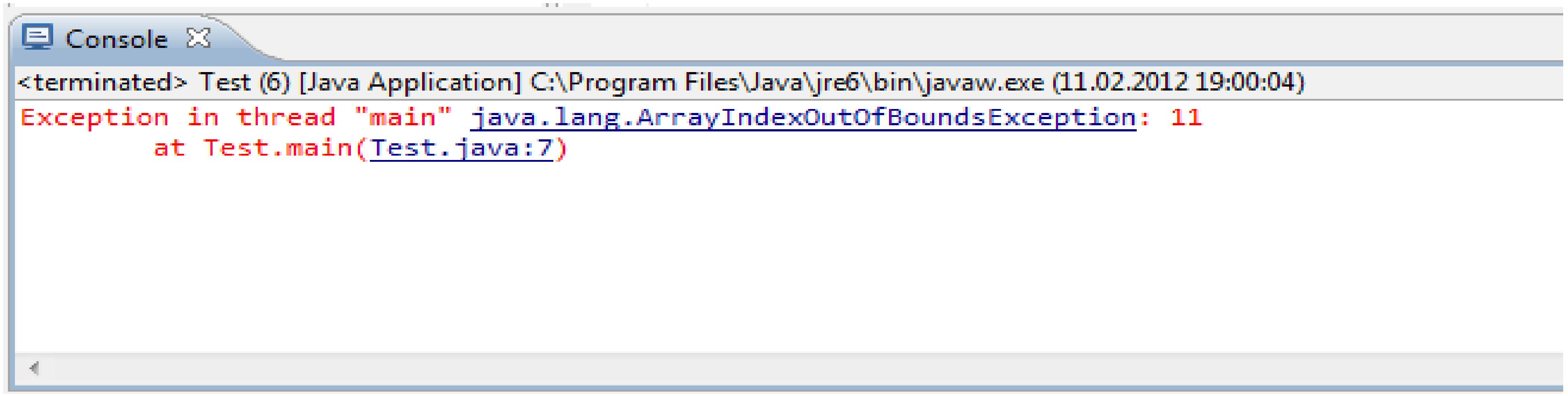
Arrays length

- array.length returns the length

```
System.out.println(array.length);
```

- Getting an element beyond the size will result in a compilation error

```
array[11] = 20;
```


Console

```
<terminated> Test (6) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (11.02.2012 19:00:04)
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 11
	at Test.main(Test.java:7)
```

Iterating the array with for loop

- Normally we’re using loops to iterate over an array
- The most common case is using a **for loop**

```
public static void main(String[] args) {
     int[] array = new int[10];
     for (int i = 0; i < array.length; i++) {
          array[i] = 7;
     }
}
```

Iterating the array with while loop

- You can iterate array with **while loop and any other**

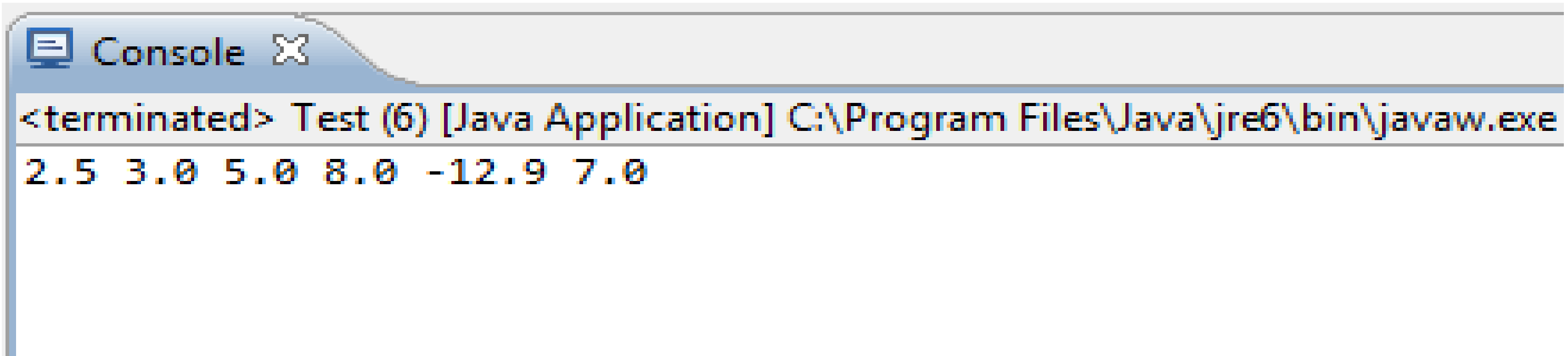
```
public static void main(String[] args) {
      int[] array = new int[10];
      int i = 0;
      while (i < array.length) {
            array[i] = 7;
            i++;
      }
}
```

Printing to console

- The array is iterated
- The value of the current element is printed using System.out.print()

```java
double[] array = { 2.5 ,3, 5, 8, -12.9, 7.0 };

for (int i = 0; i < array.length; i++) {
     System.out.print(array[i] + " ");
}
```



Console

```
<terminated> Test (6) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe
2.5 3.0 5.0 8.0 -12.9 7.0
```

Reading from console

- The array is iterated
- Use scanner to read the value from the console
- Assign the read value to the current element

```
public static void main(String[] args) {
      //declaration and initialization
      int[] array = new int[10];

      //create Scanner
      Scanner sc = new Scanner(System.in);

      //Iterate with for loop and read value for each
      //element from console
      for (int i = 0; i < array.length; i++) {
            System.out.println("Enter value:");
            array[i] = sc.nextInt();
      }
} //ArrayReadingFromConsole.java in code examples
```

Comparing arrays

- Arrays are referred types and can't be compared using **== operator**
- To compare two arrays, you have to iterate them and compare their elements respectively.

- Let’s give it a try!

```
double[] array = { 2.5 , 3, 5.8 };
double[] array2 = new double[3];
array2[0] = 2.5;
array2[1] = 3;
array2[2] = 5.8;
...
```

- Lets take a look in ArrayCompare.java in code examples

Copying arrays

- `int[] newArray = oldArray;`
- The line above is not really what you want
- What would be the result of this code?

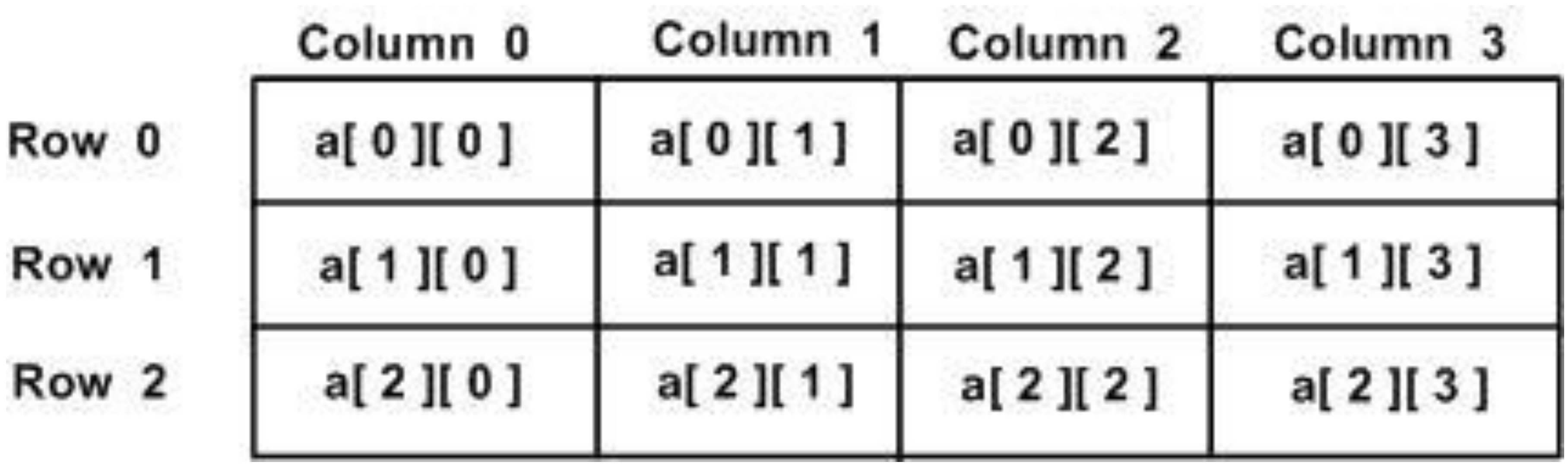
```
public static void main(String[] args) {
      int[] oldArray = { 1, 2, 3};
      int[] newArray = oldArray;

      oldArray[0] = -10;
      System.out.println(newArray[0]);
}
```

- Lets demonstrate System.arraycopy() method ArrayCopyDemo.java in code examples

Multidimentional Arrays

- Have more than one dimension (2, 3, 4, …)
- The 2-dimensional arrays are called matrices
- A matrix is an array in which each element is an array



	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Creating and iterating matrix

- The multidimensional arrays use the same concept as an ordinary arrays

```
public static void main(String[] args) {
      int[][] matrix = new int[3][4];

      for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                  matrix[i][j] = 10;
            }
      }
      matrix[0][0] = 1;
      matrix[2][3] = 100;
}
```

Creating the array

Setting value for top left element

Setting value for bottom right element

Summary

- What is a class?
- What is an object?
- What's the differences between classes and object
- How to declare property of a class
- Use objects as fields
- How to create an object
- How to declare and call methods
- What is an array and how to use it?