

Αντικειμενοστρεφής Προγραμματισμός Διάλεξη – 4 : **CLASSES**

Κων. Κόκκινος

Αντικειμενοστραφής Προγραμματισμός

- Η ιδέα του αντικειμενοστραφούς προγραμματισμού
- Αυτόνομες οντότητες
- Στιγμιότυπα οντοτήτων
- Παράδειγμα
- **Person**
 - Ιδιότητες
 - Όνομα
 - Ύψος
 - κλπ
 - Ενέργειες
 - Cook
 - Peel
- **Vegetable**
 - Ιδιότητες
 - Χρώμα
 - Μέγεθος
 - όνομα
 - Ενέργειες
 - τίποτα
- **Μεταβλητές**
 - Μεταβλητές στιγμιότυπων
 - Μεταβλητές υπόστασης
- **Συναρτήσεις μέθοδοι**
 - Παράμετροι
 - Return data type
- **ΠΑΡΑΔΕΙΓΜΑ**

Παράδειγμα κλάσης

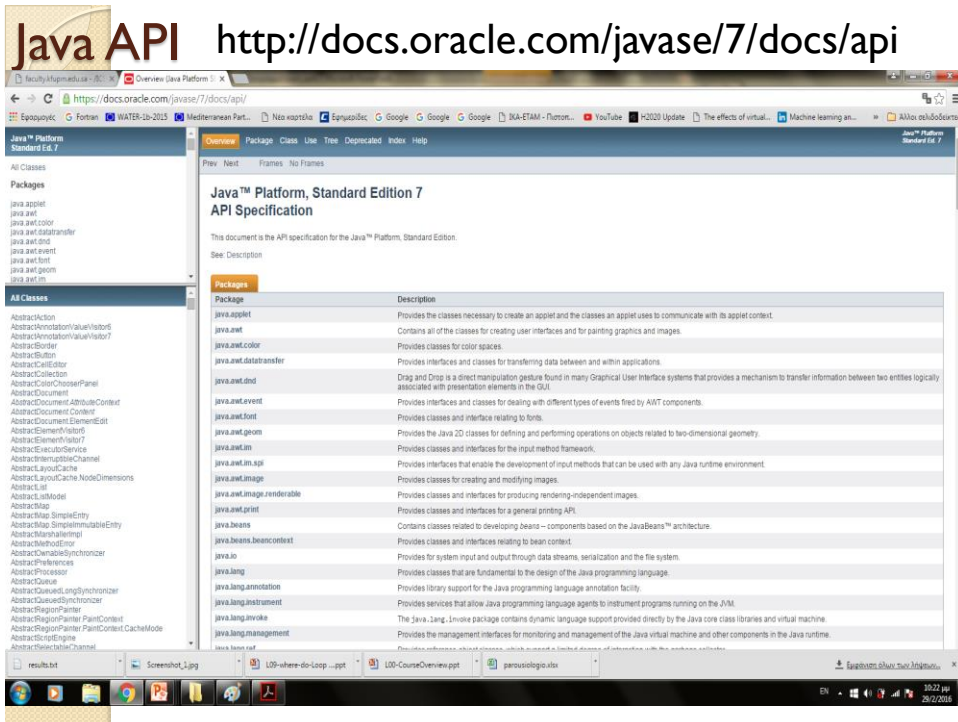
```
class Person {
    //ιδιότητες
    string name;
    int height;
    static int numberofeyes; // στατικές μεταβλητές
    void peel() {
        system.out.println("I am " + name + "and I peel vegies");
    }
    static void whatlam { // στατικές μέθοδοι
        system.out.println("I am a person");
    }
}
```

Παράδειγμα κλάσης

```
class vegetable {
    //ιδιότητες
    string name;
    string color;
}

class driver {
    public static void main(String args[]){
        Person p;
        p = new Person();
        p.name="George";
        p.height=182;
        p.peel();
        vegetable v;
        v.name="tomatoe";
        v.color="red";
        Person.numberofeyes=2;
        Person.whatlam();
    }
}
```

Java API <http://docs.oracle.com/javase/7/docs/api>



The screenshot shows the Java API documentation for Java Platform, Standard Edition 7. The page title is "Java™ Platform, Standard Edition 7 API Specification". Below the title, there is a table listing various packages and their descriptions. The table has two columns: "Package" and "Description".

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans – components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides services that allow Java programming language agents to instrument programs running on the JVM.
java.lang.invoke	The Java .Lang.Invoke package contains dynamic language support provided directly by the Java core class libraries and virtual machine.
java.lang.management	Provides the management interfaces for monitoring and management of the Java virtual machine and other components in the Java runtime.
java.lang.ref	Provides references, which are objects that provide a limited degree of interaction with the garbage collector.

ΠΑΚΕΤΑ

Προκαθορισμένες κλάσεις (*classes*) και μέθοδοι (*methods*), ομαδοποιούνται σε πακέτα (*packages*) όπως π.χ.:

- **java.lang**: βασικές κλάσεις και μέθοδοι, απαραίτητες σε κάθε πρόγραμμα
- **java.math**: κλάσεις και μέθοδοι για μαθηματικές πράξεις
- **java.io**: κλάσεις και μέθοδοι για είσοδο/έξοδο δεδομένων/αποτελεσμάτων
- **java.awt**: κλάσεις για τη δημιουργία γραφικών (*graphics*) και περιβαλλόντων διασύνδεσης (*user interfaces*)
- **javax.swing**: κλάσεις για τη δημιουργία γραφικών και περιβαλλόντων διασύνδεσης χρησιμοποιώντας μόνο Java ("*lightweight*", *all-Java language*)
- **javax.sql**: κλάσεις και μέθοδοι για πρόσβαση σε δομές δεδομένων (*database*)
- **java.applet**: κλάσεις και μέθοδοι για *Java applets*

Στατικές μέθοδοι της τάξης (κλάσης) *Math*

Method	Description	Example
<code>abs(x)</code>	absolute value of x	<code>abs(23.7)</code> is 23.7 <code>abs(0.0)</code> is 0.0 <code>abs(-23.7)</code> is 23.7
<code>ceil(x)</code>	rounds x to the smallest integer not less than x	<code>ceil(9.2)</code> is 10.0 <code>ceil(-9.8)</code> is -9.0
<code>cos(x)</code>	trigonometric cosine of x (x in radians)	<code>cos(0.0)</code> is 1.0
<code>exp(x)</code>	exponential method e^x	<code>exp(1.0)</code> is 2.71828 <code>exp(2.0)</code> is 7.38906
<code>floor(x)</code>	rounds x to the largest integer not greater than x	<code>Floor(9.2)</code> is 9.0 <code>floor(-9.8)</code> is -10.0
<code>log(x)</code>	natural logarithm of x (base e)	<code>log(Math.E)</code> is 1.0 <code>log(Math.E * Math.E)</code> is 2.0
<code>max(x, y)</code>	larger value of x and y	<code>max(2.3, 12.7)</code> is 12.7 <code>max(-2.3, -12.7)</code> is -2.3
<code>min(x, y)</code>	smaller value of x and y	<code>min(2.3, 12.7)</code> is 2.3 <code>min(-2.3, -12.7)</code> is -12.7
<code>pow(x, y)</code>	x raised to the power y (i.e., x^y)	<code>pow(2.0, 7.0)</code> is 128.0 <code>pow(9.0, 0.5)</code> is 3.0
<code>sin(x)</code>	trigonometric sine of x (x in radians)	<code>sin(0.0)</code> is 0.0
<code>sqrt(x)</code>	square root of x	<code>sqrt(900.0)</code> is 30.0
<code>tan(x)</code>	trigonometric tangent of x (x in radians)	<code>tan(0.0)</code> is 0.0

Περιβάλλουσες κλάσεις

- Integer `int`
- Float `float`
- Double `double`
- Char `char`

Δημιουργία αντικειμένων τύπου σαν και τα παραπάνω

Υπερφόρτωση Μεθόδων

- Πολλαπλές μέθοδοι μπορούν να καθοριστούν με το ίδιο όνομα εφόσον έχουν διαφορετικές παραμέτρους όσο αφορά τον αριθμό και τύπο τους
- Ο μεταγλωττιστής (*compiler*) αποφασίζει ποια μέθοδο να καλέσει βάσει της ταύτισης των παραμέτρων που χρησιμοποιούνται στην κλήση της μεθόδου και των παραμέτρων κάποιας από τις υπερφορτωμένες μεθόδους
- Διαφορές στον τύπο που επιστρέφεται (*return type*) από τη μέθοδο δεν επηρεάζουν την απόφαση για το ποια μέθοδος να κληθεί αφού δεν είναι μέρος της υπογραφής της μεθόδου (*method signature*)
- Είναι λάθος να έχει την ίδια υπογραφή μιας μέθοδος (*method signature*), ακόμη και αν έχει διαφορετικό τύπο επιστροφής (*return type*)

Παράδειγμα

```

1 // Fig. 6.13: MethodOverload.java
2 // Overloaded method declarations.
3
4 public class MethodOverload
5 {
6     // test overloaded square methods
7     public void testOverloadedMethods()
8     {
9         System.out.printf( "Square of integer 7 is %d\n", square( 7 ) );
10        System.out.printf( "Square of double 7.5 is %f\n", square( 7.5 ) );
11    } // end method testOverloadedMethods
12
13    // square method with int argument
14    public int square( int intValue )
15    {
16        System.out.printf( "\nCalled square with int argument: %d\n",
17            intValue );
18        return intValue * intValue;
19    } // end method square with int argument
20
21    // square method with double argument
22    public double square( double doubleValue )
23    {
24        System.out.printf( "\nCalled square with double argument: %f\n",
25            doubleValue );
26        return doubleValue * doubleValue;
27    } // end method square with double argument
28 } // end class MethodOverload

```

Παράδειγμα

```

1 // Fig. 6.14: MethodOverloadTest.java
2 // Application to test class MethodOverload.
3
4 public class MethodOverloadTest
5 {
6     public static void main( String args[] )
7     {
8         MethodOverload methodOverload = new MethodOverload();
9         methodOverload.testOverloadedMethods();
10    } // end main
11 } // end class MethodOverloadTest

```

```

Called square with int argument: 7
Square of integer 7 is 49

```

```

Called square with double argument: 7.500000
Square of double 7.5 is 56.250000

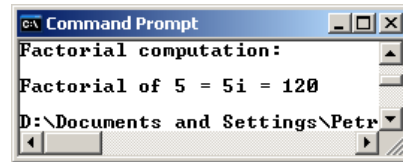
```

Αναδρομικές μέθοδοι (recursive methods)

- Οι αναδρομικές μέθοδοι μπορούν να χρησιμοποιηθούν για να επιλυθεί κάποιο πρόβλημα με νέα κλήση μέσα από την ίδια μέθοδο του εαυτού της για επίλυσης πιο απλοποιημένου προβλήματος, εφαρμόζοντας κατά κάποιο τρόπο το διαίρει και βασίλευε (*divide & conquer*)
- Η *Java* μας δίνει αυτή τη δυνατότητα αφού επιτρέπεται να κληθεί μία μέθοδος μέσα από εκτέλεση της ίδιας μεθόδου
- Πολλοί αλγόριθμοι μπορούν εναλλακτικά να υλοποιηθούν πιο κομψά με χρήση *recursive methods/functions* παρά με χρήση επαναλήψεων (*iteratively*)
- Κάποιος αλγόριθμος που υλοποιείται αναδρομικά (*recursively*) μπορεί να υλοποιηθεί και επαναληπτικά (*iteratively*)
- Πρέπει να υπάρχουν κάποιες συνθήκες τερματισμού της αναδρομής πιο κοντά στις οποίες πρέπει να οδηγούμαστε με κάθε νέα αναδρομική κλήση

ΠΑΡΑΔΕΙΓΜΑ Υπολογισμός παραγοντικού

```
public class TestMyFactorial {  
  
    public static void main(String [] args) {  
  
        System.out.println("Factorial computation:\n");  
  
        int n=5;  
  
        System.out.printf("Factorial of %d = %di = %d\n", n, n,  
            recursiveFactorial(n));  
  
    }  
  
    static int recursiveFactorial(int n) {  
  
        if (n>1)  
  
            return n*recursiveFactorial(n-1);  
  
        else  
  
            return 1;  
  
    }  
}
```



```
c:\ Command Prompt  
Factorial computation:  
Factorial of 5 = 5! = 120  
D:\Documents and Settings\Petr
```