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 στιγμιοτύπων
  - Μεταβλητές υπόστασης
- Συναρτήσεις μέθοδοι
  - Παράμετροι
  - Return data type
- ΠΑΡΑΔΕΙΓΜΑ

Παράδειγμα κλάσης

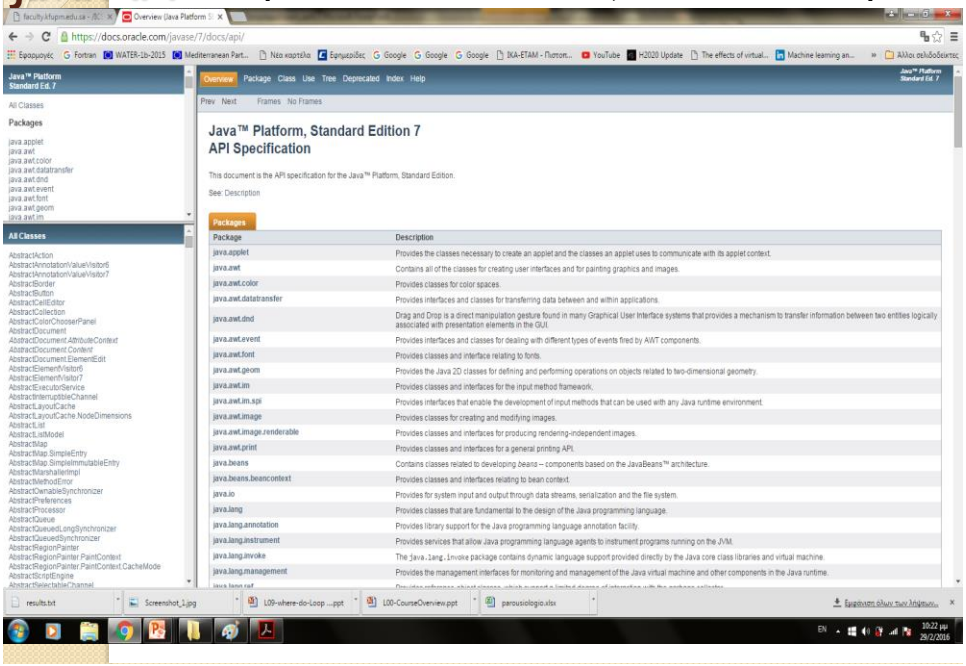
```
class Person {
    //ιδιότητες
    string name;
    int height;
    static int numberofeyes;   // στατικές μεταβλητές
    void peel() {
        system.out.println("I am " + name + "and I peel vegies");
    }
    static void whatlam {  // στατικές μέθοδοι
        system.out.println("I am a person");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

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



ΠΑΚΕΤΑ

Προκαθορισμένες κλάσεις (*classes*) και μέθόδοι (*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
abs(x)	absolute value of x	abs(23.7) is 23.7 abs(0.0) is 0.0 abs(-23.7) is 23.7
ceil(x)	rounds x to the smallest integer not less than x	ceil(9.2) is 10.0 ceil(-9.8) is -9.0
cos(x)	trigonometric cosine of x (x in radians)	cos(0.0) is 1.0
exp(x)	exponential method ex	exp(1.0) is 2.71828 exp(2.0) is 7.38906
floor(x)	rounds x to the largest integer not greater than x	floor(9.2) is 9.0 floor(-9.8) is -10.0
log(x)	natural logarithm of x (base e)	log(Math.E) is 1.0 log(Math.E * Math.E) is 2.0
max(x, y)	larger value of x and y	max(2.3, 12.7) is 12.7 max(-2.3, -12.7) is -2.3
min(x, y)	smaller value of x and y	min(2.3, 12.7) is 2.3 min(-2.3, -12.7) is -12.7
pow(x, y)	x raised to the power y (i.e., xy)	pow(2.0, 7.0) is 128.0 pow(9.0, 0.5) is 3.0
sin(x)	trigonometric sine of x (x in radians)	sin(0.0) is 0.0
sqrt(x)	square root of x	sqrt(900.0) is 30.0
tan(x)	trigonometric tangent of x (x in radians)	tan(0.0) 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 [] agrs) {

                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;

        }

}
```