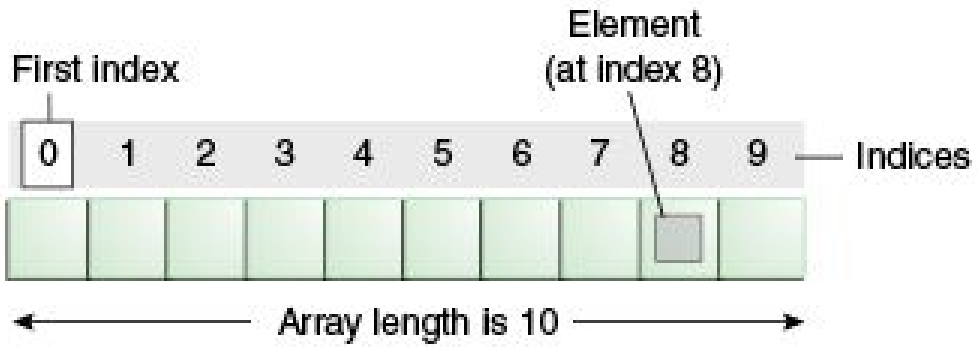
Πίνακες, Τελεστής this, Στατικές Μεταβλητές, autoboxing & auto-unboxing

Γιώργος Θάνος
gthanos@uth.gr
Γραφείο: Γ5/8, 3ος όροφος
Γκλαβάνη 37

Πίνακες

Πίνακες

```
int [] anArray; anArray = new int[10];
ή
int [] anArray = new int[10];
ή
int [] anArray = {2,3,4,5,6,7,8,9,10,11};
```




```
class ArrayDemo {
  public static void main(String[] args) {
    int[] anArray = new int[3];
    anArray[0] = 100; anArray[1] = 200;
    anArray[2] = 300;
    for(int i=0; i < anArray.length; i++) {
      System.out.println("Element at index " + i +
      "/" + anArray.length+ ": " + anArray[i]);
    }
    int j=0;
    for(int element: anArray) {
      System.out.println("Element at index " +
    (j++) + "/" + anArray.length+ ": " + element);
    }
  }
}
```

Παράδειγμα πινάκων αναφορικού τύπου

```
public class CreateObjectDemo {

    public static void main(String[] args) {

        Rectangle [] rectangles = new Rectangle[10];

        Point originOne = new Point(23, 94);

        Rectangle rectOne = new Rectangle(100, 200, originOne);

        Rectangle rectTwo = new Rectangle(50, 100);


        rectangles[0] = rectOne;

        rectangles[1] = rectTwo;

        rectangles[3] = new Rectangle(20, 30, new Point(-5, 7));

        }

}
```

Πίνακες δύο διαστάσεων (1/2)

```
class MultiDimArrayDemo {
  public static void main(String[] args) {
    String [][] names = new String[3][5];
  }
}
```

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	**[2][3]**	[2][4]

Στον διπλανό πρόγραμμα αποτυπώνεται η δημιουργία πίνακα δύο διαστάσεων (3 γραμμών και 5 στηλών).

Στο σχήμα κάτω αριστερά αποτυπώνεται η "αρίθμηση" των στοιχείων του πίνακα. Η αρίθμηση ξεκινά πάντα από το μηδέν (0) για τις στήλες και τις γραμμές.

Παραδείγματα:

1. Το στοιχείο της 2ης γραμμής και της 3ης στήλης αντιστοιχεί στο `names[1][2]`
2. Το στοιχείο της 3ης γραμμής και της 4ης στήλης αντιστοιχεί στο `names[2][3]`

Πίνακες δύο διαστάσεων (2/2)

```
class MultiDimArrayDemo {
  public static void main(String[] args) {
    String [][] names;
    names = new String[3][];
    names[0] = new String[3]; names[1] = new String[4]; names[2] = new String[4];
    names[0][0]="Mr. "; names[0][1]="Mrs. "; names[0][2]="Ms. ";
    names[1][0]="Susan "; names[1][1]="Peter "; names[1][2]="Edgar "; names[1][3]="Mary ";
    names[2][0]="Smith"; names[2][1]="Jones"; names[2][2]="Williams"; names[2][3]="Taylor";
    System.out.println(names[0][0] + names[1][1] + names[2][3] + " enters the room.");
    System.out.println(names[0][1] + names[1][3] + names[2][2] + " leaves the room.");
  }
}
```

Παρατηρήστε ότι μπορούμε να έχουμε διαφορετικό αριθμό στηλών σε κάθε γραμμή του πίνακα.

Ο τελεστής this

Ο τελεστής **this** υποδεικνύει το τρέχον αντικείμενο.

Πρόσβαση σε πεδία και μεθόδους της κλάσης μέσω του τελεστή this


```
public class Rectangle {

   private int width;
   private int height;
   private Point origin;

   public Rectangle(int width, int height,
 Point origin) {
      width = width;
      height = height;
      origin = origin;
   }
   ...
 }
```

Στον παραπάνω κώδικα οι τοπικές παράμετροι του κατασκευαστή συμπίπτουν με τα ονόματα των μεταβλητών της κλάσης. Σε αυτή την περίπτωση η τοπική μεταβλητή/παράμετρος "κρύβει" (*shadows*) την μεταβλητή του αντικειμένου της κλάσης.

Προκειμένου να έχουμε πρόσβαση στην μεταβλητή του αντικειμένου της κλάσης θα πρέπει να χρησιμοποιήσουμε τον τελεστή **this**, όπως παρακάτω

```
public class Rectangle {

   private int width;
   private int height;
   private Point origin;

   // the Rectangle class has one constructor
   public Rectangle(int width, int height,
 Point origin) {
      this.width = width;
      this.height = height;
      this.origin = origin;
   }
   ...
 }
```

Πρόσβαση στους κατασκευαστές της κλάσης μέσω του τελεστή **this**

```
public class Rectangle {

   private int width;
   private int height;
   private Point origin;

   public Rectangle(int initWidth, int
 initHeight, Point initOrigin) {
      width = initWidth;
      height = initHeight;
      origin = initOrigin;
   }

   public Rectangle(int initWidth, int
 initHeight, int xPos, int yPos) {
      width = initWidth;
      height = initHeight;
      origin = new Point(xPos,yPos);
   }
   ....
 }
```

```
public class Rectangle {

   private int width;
   private int height;
   private Point origin;

   public Rectangle(int initWidth, int initHeight,
 Point initOrigin) {
      width = initWidth;
      height = initHeight;
      origin = initOrigin;
   }


   public Rectangle(int initWidth, int initHeight,
 int xPos, int yPos) {

  this(initWidth, initHeight, new Point(xPos,yPos));
   }
   ....
}
```

ισοδύναμα

Η κλήση ενός κατασκευαστή μέσα από άλλο κατασκευαστή θα πρέπει να γίνει στην αρχή του σώματος του κατασκευαστή.

Εάν προηγηθεί μία άλλη εντολή πριν την κλήση του κατασκευαστή θα λάβετε ένα μήνυμα λάθους από τον compiler.

```
public class Rectangle {

   ….
   public Rectangle(int width, int height,
Point origin) {
      this.origin = origin;
      /* η κλήση ενός κατασκευαστή μέσα από
άλλο
       * κατασκευαστή, χωρίς να είναι 1η
εντολή
       * παράγει μήνυμα λάθους από τον
compiler.
       */
      this(width, height);

   }
   ....
}
```

Συμπερασματικά

Ο τελεστής **this** δείχνει στο τρέχον αντικείμενο της εκάστοτε κλάσης. Μπορείτε να τον χρησιμοποιείτε:

1. για να διακρίνεται τα πεδία του τρέχοντος αντικειμένου σε σχέση με τοπικές μεταβλητές που έχουν το ίδιο όνομα με τα πεδία αυτά.
2. για να καλέσετε τον κατασκευαστή της κλάσης για το τρέχον αντικείμενο.

Στατικές μεταβλητές και μέθοδοι της κλάσης

Μέχρι τώρα είδαμε ότι όλα τα πεδία και οι μέθοδοι της κλάσης ανήκουν στα επιμέρους αντικείμενα της κλάσης.

Είναι δυνατόν να έχουμε πεδία και μεθόδους που δεν ανήκουν στα αντικείμενα της κλάσης, αλλά στην κλάση.

Με την έννοια αυτή, οι συγκεκριμένες μέθοδοι και πεδία είναι στατικά για τα αντικείμενα της κλάσης.

Στατικές μεταβλητές της κλάσης

```
public class Rectangle {

   private int width;
   private int height;
   private Point origin;

   // the Rectangle class has one constructor
   public Rectangle(int width, int height, Point origin) {
     this.width = width;
     this.height = height;
     this.origin = origin;
   }

   // add an instance variable for the object ID
   private int id;

   // add a class variable for the
   // number of Rectangle objects instantiated
   private static int numberOfRectangles = 0;

   /* οι υπόλοιπες εντολές της κλάσης είναι όμοιες
    * με προηγούμενα παραδείγματα και παραλείπονται
    * για λόγους οικονομίας χώρου.
    */
}
```

Δημιουργούμε τη μεταβλητή **numberOfRectangles** που έχει ενιαία τιμή για όλα τα αντικείμενα της κλάσης.

Η μεταβλητή αυτή δεν ανήκει σε κάποιο από τα αντικείμενα, αλλά ΜΟΝΟ στην κλάση.

Για τα αντικείμενα της κλάσης Rectangle, η συγκεκριμένη μεταβλητή μπορεί να θεωρηθεί στατική (*για αυτό και η χρήση του keyword static*).

Προσπέλαση στατικών πεδίων

1. Με χρήση του ονόματος της κλάσης
   `Rectangle.numberOfRectangles++;`
2. Με χρήση ενός αντικειμένου ως εξής
   `Rectangle myRectangle = new Rectangle();`
   `myRectangle.numberOfRectangles++;`

Στατικές μέθοδοι της κλάσης

Σε αναλογία με τις μεταβλητές της κλάσης μπορούμε να έχουμε και μεθόδους αποκλειστικά σε επίπεδο κλάσης. Οι μέθοδοι της κλάσης χρησιμοποιούνται για να προσπελάσουν στατικά πεδία της κλάσης.

Επίσης, μέσα σε μία στατική μέθοδο μπορούν να δημιουργηθούν αντικείμενα οποιασδήποτε κλάσης (π.χ. η στατική μέθοδος **main** από την οποία εκκινούν όλα τα προγράμματα).

```
public static int getNumberOfRectangles() {
     return numberOfRectangles;

}
```

Για να καλέσουμε στατικές μεθόδους σε αναλογία με τα πεδία έχουμε τις εξής επιλογές:

1. Με χρήση του ονόματος της κλάσης
   `Rectangle.getNumberOfRectangles();`
2. Με χρήση ενός αντικειμένου ως εξής
   ~~`Rectangle myRectangle = new Rectangle();`~~
   ~~`myRectangle.getNumberOfRectangles();`~~

Χρήση στατικών μεθόδων

Μέσα σε μία στατική μέθοδο μπορούν να δημιουργηθούν αντικείμενα οποιασδήποτε κλάσης.

```
public class CreateObjectDemo {

public static void main(String[] args) {

 Point originOne = new Point(23, 94);

 Rectangle rectTwo = new Rectangle(50, 100);

 }

...

}
```

Οι στατικές μέθοδοι μπορούν να χρησιμοποιηθούν για να εκτελεστούν εργασίες που δεν συνδέονται με τα πεδία της κλάσης.

```
public class Sort {
   public static void bubbleSort(String a[],
                                  int left, int right) {
     String temp;
     for(int i=left; i<right; i++) {
       for(int j=right; j>i; j--) {
         if( a[j].compareTo(a[j-1])<0 ) {
           temp = a[j-1];
           a[j-1] = a[j];
           a[j] = temp;
         }
       }
     }
   }
```

Πρόσβαση στατικών/μη στατικών πεδίων από στατικές/μη στατικές μεθόδους

Μη στατικές μέθοδοι ΜΠΟΡΟΥΝ να προσπελάσουν μη στατικά πεδία.

Μη στατικές μέθοδοι ΜΠΟΡΟΥΝ να προσπελάσουν στατικά πεδία.

Στατικές μέθοδοι ΜΠΟΡΟΥΝ να προσπελάσουν στατικά πεδία.

Στατικές μέθοδοι ΔΕΝ ΜΠΟΡΟΥΝ να προσπελάσουν μη στατικά πεδία.

Σταθερές - Constants

Η δημιουργία σταθερών επιτυγχάνεται με χρήση του συνδυασμού των keywords **static** και **final**.

Εάν θέλουμε να γράψουμε την σταθερά π=3.141592653589793

```
static final double PI = 3.141592653589793;
```

Η μέθοδος toString

Για την κλάση Point που ορίσαμε σε προηγούμενη ενότητα. Θα θέλαμε να μπορούμε να γράψουμε:

```
Point aPoint = new Point(4, 5);

System.out.println("Point value is: " + aPoint);

ισοδύναμα System.out.println(aPoint.toString());
```

και να εκτυπωθεί (x=4, y=5). Αν όμως εκτελέσουμε το παραπάνω κομμάτι κώδικα, τότε θα εκτυπωθεί το αρκετά διαφορετικό: **`Point@5ffdfb42`**, το οποίο αποτελείται από το όνομα της κλάσης στην οποία ανήκει το αντικείμενό μας και ένα κωδικό (hash code του αντικειμένου) .

Αυτό που δε φαίνεται στον κώδικα είναι ότι όταν ζητάμε να αντιμετωπιστεί ένα αντικείμενο ως κείμενο (πχ. για να εκτυπωθεί στην οθόνη) τότε καλείται έμμεσα μια συνάρτηση με όνομα toString η οποία είναι προκαθορισμένη για κάθε κλάση και επιστρέφει ένα String όπως το **`Point@5ffdfb42`**.

Ορίζουμε λοιπόν στην κλάση **`Point`** την μέθοδο **`public String toString()`** ως εξής:

```
public String toString () {
     return "(x=" + x + ", y=" + y + ")";
 }
```

ώστε να εκτυπωθεί το ζητούμενο (4, 5) .

Βασικοί τύποι δεδομένων και ισοδύναμοι αναφορικοί τύποι

Για όλους τους βασικούς τύπους που συναντήσαμε η Java ορίζει τους παρακάτω ισοδύναμούς αναφορικούς τύπους δεδομένων.

Βασικός τύπος	Αναφορικός τύπος
boolean	Boolean
byte	Byte
char	Character
int	Integer
long	Long
short	Short
float	Float
double	Double

Κάθε αναφορικός τύπος μπορεί να δημιουργηθεί από ένα βασικό τύπο ή από ένα αλφαριθμητικό (String).

Για παράδειγμα η κλάση Integer έχει τους παρακάτω δύο κατασκευαστές:

```
Integer(int value);
Integer(String s);
```

Οι παραπάνω κλάσεις περιέχουν μεθόδους για την εξαγωγή ενός βασικού τύπου δεδομένων από τον αναφορικό τύπο.

```
byte byteValue();
short shortValue();
int intValue();
long longValue();
float floatValue();
```

Auto-boxing και Auto-unboxing

Ο compiler της Java μας δίνει την δυνατότητα να χρησιμοποιήσουμε βασικούς τύπους σε σημεία του κώδικα που απαιτείται η ισοδύναμη αναφορική μορφή ή αναφορικούς τύπους σε σημεία του κώδικα που ζητείται η βασική μορφή.

Ο compiler έχει την ιδιότητα να αναγνωρίζει σημεία του κώδικα που απαιτούν τις παραπάνω μετατροπές τύπου και κάνει αυτόματα τις μετατροπές αυτές.

Η ιδιότητα αυτή του Java compiler απαντάται στην βιβλιογραφεία ως **Auto-boxing** και **Auto-unboxing**.

Παράδειγμα Auto-boxing

```
public class AutoboxExample {
   public static void main(String []args) {
     int a=5;
     System.out.println("a: "+a);
   }
}
```

Ο κώδικας εκτυπώνει την τιμή της μεταβλητής a που τυγχάνει να είναι βασικού τύπου.

Για να το επιτύχει αυτό καλείται να εκτυπώσει ένα αλφαριθμητικό (String) που αποτελείται από την ένωση δύο αλφαριθμητικών όπως αυτά ορίζονται πριν και μετά τον τελεστή +.

Αριστερά του τελεστή + έχουμε το αλφαριθμητικό “a: ” ενώ δεξιά του τελεστή έχουμε μία μεταβλητή τύπου int.

Προκειμένου να μπορεί να εκτελεστεί το πρόγραμμα, ο compiler θα πρέπει να μετατρέψει τον βασικό τύπο σε αναφορικό και στη συνέχει να καλέσει την συνάρτηση toString() του αναφορικού τύπου που επιστρέφει την τιμή της μεταβλητής ως String.

Η μετατροπή που κάνει ο compiler, μπορεί να γραφεί ισοδύναμα ως εξής

```
public class AutoboxExample {
   public static void main(String []args) {
     int a=5;
     System.out.println("a: "+
     (new Integer(a)).toString() );
   }
}
```

Παράδειγμα Auto-unboxing

```
public class UnboxExample {
   public static int sum(int a, int b) {
      return a+b;
   }

   public static void main(String []args) {
     Integer a = new Integer(5);
     Integer b = new Integer(10);
     int result = sum(a,b);
     System.out.println("sum: "+ result );
   }
}
```

Είναι προφανές ότι η μέθοδος *sum* λαμβάνει ως ορίσματα δύο ακεραίους βασικού τύπου, ενώ τα ορίσματα με τα οποία καλείται η συνάρτηση είναι ακέραιοι αναφορικού τύπου ***Integer***.

Ο *compiler* αντιλαμβάνεται ότι προκειμένου να χρησιμοποιηθεί η συνάρτηση απαιτείται η μετατροπή των δύο μεταβλητών αναφορικού τύπου σε βασικού τύπου.

Ο ισοδύναμος κώδικας που υλοποιεί ο *compiler* είναι ο παρακάτω

```
public class UnboxExample {
   public static int sum(int a, int b) {
      return a+b;
   }

   public static void main(String []args) {
     Integer a = new Integer(5);
     Integer b = new Integer(10);
     int result = sum(a.intValue(),b.intValue());
     System.out.println("sum: "+ result );
   }
}
```

Κανόνες μετατροπής Auto-boxing/unboxing

Ο compiler έχει την δυνατότητα να μετατρέπει τους βασικούς τύπους σε ισοδύναμους αναφορικούς και αντίστροφα, αλλά δεν επιτρέπει την μετατροπή σε μη ισοδύναμους τύπους.

Για παράδειγμα, στο διπλανό πρόγραμμα, εάν ο ένας από τους δύο αριθμούς είναι τύπου **Long** αντί για **Integer** τότε δεν καλείται η μέθοδος **a.intValue()** που είναι διαθέσιμη και στην κλάση **Long**.

```
public class UnboxExample {
   public static int sum(int a, int b) {
      return a+b;
   }

   public static void main(String []args) {
     Long a = new Long(5);
     Integer b = new Integer(10);
     int result = sum(a,b);
     System.out.println("sum: "+ result );
   }
}
```

```
$ javac UnboxExample.java
UnboxExample.java:9: error: method sum in class UnboxExample cannot be applied
to given types;
    int result = sum(a,b);
                 ^
  required: int,int
  found: Long,Integer
  reason: actual argument Long cannot be converted to int by method invocation
conversion
1 error
```