Διάλεξη 4: Προγραμματισμός σε JAVA II

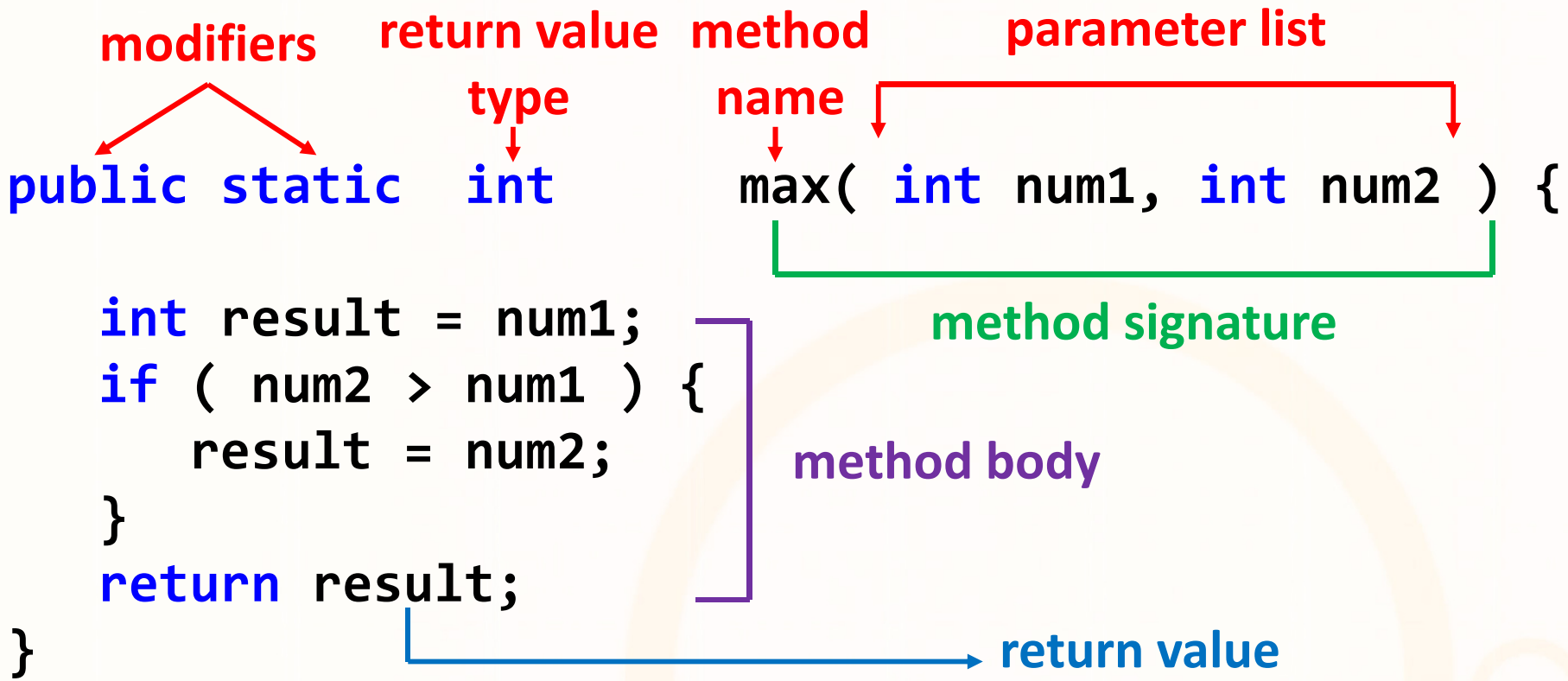
Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:
Εισαγωγή στις έννοιες:

- Μέθοδοι
- Πίνακες, Πολυδιάστατοι Πίνακες
- Boxing/Unboxing
- Χρήσιμες βιβλιοθήκες

Διδάσκων: Παναγιώτης Ανδρέου

Μέθοδοι (methods)

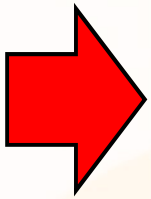
- **Μέθοδος:** μία συλλογή από ομαδοποιημένες δηλώσεις οι οποίες εκτελούν κάποια (ες) λειτουργία (ες).
- Η **υπογραφή μίας μεθόδου** αποτελείται από το **όνομά της** και τη **λίστα με της παραμέτρους** που δέχεται.
- Σύνταξη:



Μέθοδοι (methods) (συν.)

- Μία μέθοδος που δεν επιστρέφει κάτι πρέπει να δηλωθεί με **void** για return type. Παράδειγμα: **public static void main(...)**
- Όταν μία μέθοδος επιστρέφει, πρέπει να υπάρχει δήλωση **return**. **ΠΡΟΣΟΧΗ**: όλα τα μονοπάτια πρέπει να καταλήγουν σε return αλλιώς θα υπάρχει compile error. Παράδειγμα:

```
int max(int num1, int num2){
   int result;
   if ( num2 > num1 ) {
      result = num2;
   }
   else if ( num2 < num1 ) {
      result = num1;
   }
   return result;
   // Αν num1 == num2 ???
}
```



```
int max(int num1, int num2){

   int result;
   if ( num2 > num1 ) {
      result = num2;
   }
   else {
      result = num1;
   }
   return result;
 //return ( num2 > num1 ) ?
    num2 : num;
}
```

Πέρασμα Παραμέτρων

- **Πέρασμα διά τιμής:** οι τιμές των μεταβλητών αντιγράφονται στις παραμέτρους της μεθόδου ==> οι αρχικές μεταβλητές δεν αλλάζουν
- **Πέρασμα διά αναφοράς: δεν υπάρχει!**
- Παράδειγμα

```
public class Increment {
   public static void main( String[] args ) {
      int x = 1;
      System.out.println(
               "before the call, x is " + x);
      increment(x);
      System.out.println(
               "after the call, x is " + x);
   }
   public static void increment( int n ) {
      n++;
      System.out.println(
               "n inside the method is " + n);
   }
}
```

1. x=1
2. x=1
3. x=1 copy to n
7. x=1
4. n=1
5. n=2
6. n=2

Μεταβλητός αριθμός παραμέτρων

- Μπορούμε να δηλώσουμε **συναρτήσεις** και **constructors** με μεταβλητό αριθμό παραμέτρων.
- Μόνο η τελευταία παράμετρος μπορεί να είναι **vararg**.
- Παράδειγμα: `int sum ( int ... nums )`
- Το "..." σημαίνει μηδέν ή περισσότερες παράμετρους
- Παραδείγματα καλέσματος:
- `sum ( 10, 20 );`
- `sum ( 10, 20, 30, 40 );`
- Για να έχουμε πρόσβαση στην κάθε παράμετρο χρησιμοποιούμε σύνταξη πίνακα ή το for-each

```
int sum ( int ... nums ) {

  int total = 0;
  for (int i=0;
       i<nums.length; i++)
    total+= nums[i];

  for (int i : nums)
     total+= i;

   return total;
}
```

Πίνακες

- **Πίνακας:** μία δομή δεδομένων που αντιπροσωπεύει μία συλλογή με στοιχεία του ίδιου τύπου.
- Οι πίνακες στην JAVA έχουν σχεδιαστεί κατά τρόπον ώστε να ξεπερνιούνται οι δυσκολίες του προγραμματισμού πινάκων της C/C++.
- Στη JAVA είναι εξασφαλισμένο ότι ένας πίνακας θα αρχικοποιηθεί και ότι δεν θα επισυμβεί πρόσβαση εκτός των ορίων του.
- Τα χαρακτηριστικά αυτά υλοποιούνται με κάποιο σχετικό κόστος μνήμης και χρόνου εκτέλεσης (κατά την εκτέλεση γίνεται έλεγχος κατά πόσο δεν γίνεται υπέρβαση των ορίων του πίνακα).
- Κατά τη δημιουργία ενός πίνακα, κατ' ουσίαν δημιουργείται ένας πίνακας χειριστηρίων (Handles), τα οποία αρχικοποιούνται σε null.
- Είναι ευθύνη του προγραμματιστή να αρχικοποιήσει σωστά τα χειριστήρια, ώστε να παραπέμπουν σε αντικείμενα.

Πίνακες (συν.)

Σύνταξη: **<τύπος δεδομένων>[ ] <όνομα πίνακα>**, π.χ., double[] d;

Δήλωση Πινάκων

- `int[] x;` `//Δήλωση πίνακα με ακέραιους`
- `char[] a;` `/Δήλωση πίνακα με χαρακτήρες`

Αρχικοποίηση Πινάκων

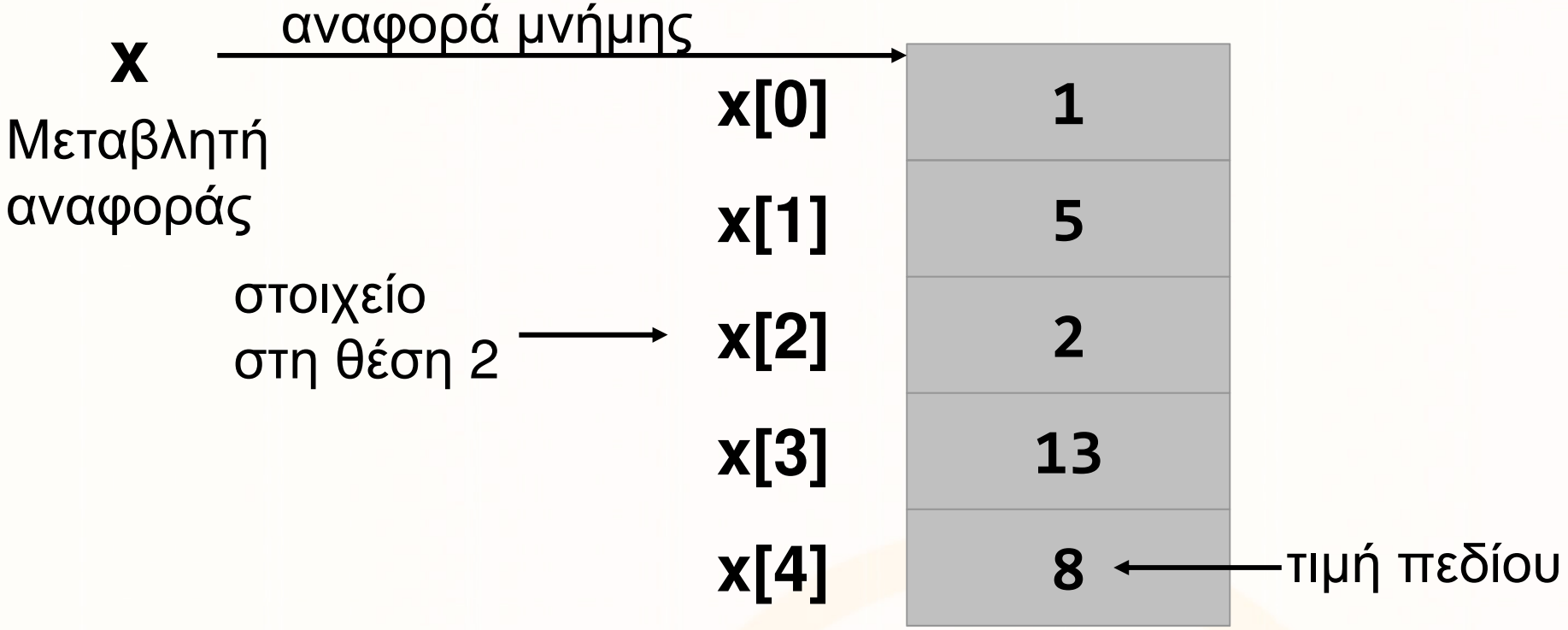
- Η αρχικοποίηση μπορεί να γίνει με τη δήλωση `new`
  `x = new int[10]; // Δημιουργία 10 θέσεων`
- ή μπορεί να γίνει με την αυτόματη ανάθεση στοιχείων
  `x = {1, 2, 3, 4, 5};` **μόνο κατά την δήλωση!**

Δήλωση και Αρχικοποίηση σε ένα Βήμα

- `int[] x= new int[10];`

Πίνακες (συν.)

- Παράδειγμα αναπαράστασης πίνακα: `int[] x = new int[5];`



- Πρόσβαση σε κάθε στοιχείο του πίνακα με το ευρετήριο του. π.χ., το στοιχείο στη θέση 2 = x[2].
- Ιδιότητα (property) <**length**>: Επιστρέφει το μέγεθος του πίνακα. π.χ., x.length = 5

Παραδείγματα χρήσης πίνακα

- Τύπωμα των στοιχείων ενός πίνακα

```
int[] x = new int[5];
...
for(int i=0; i<x.length; i++) {
   System.out.println( x[i] );
}
```

- Πρόσθεση των στοιχείων ενός πίνακα

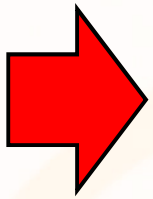
```
int[] x = new int[5];
int sum=0;
...
for(int i=0; i<x.length; i++) {
   sum = sum + x[i];
}
```

Πίνακες και for-each

Η Java1.5 εισήγαγε μία καινούρια δήλωση βρόγχου το οποίο επιτρέπει τη διάσχιση ενός πίνακα χωρίς τη χρήση δείκτη.

- Σύνταξη: **for(<τύπος δεδομένων> <μεταβλητή> : <πίνακας) { ... }**
- **Παράδειγμα:**

```
int[] x = new int[5];
...
for(int i=0;
    i<x.length; i++) {
   System.out.println(
           x[i] );
}
```

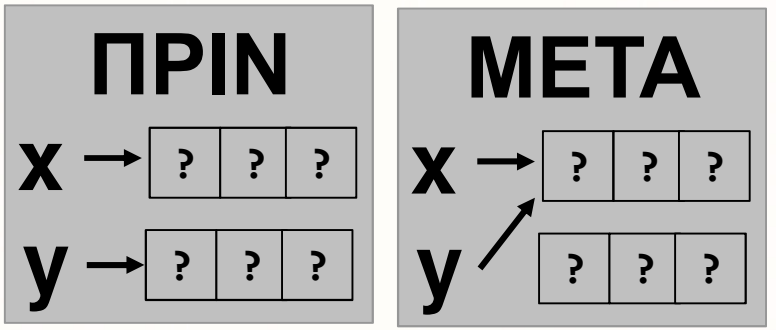


```
int[] x = new int[5];
...
for(int i: x) {
   System.out.println( i );
}
```

Αντιγραφή Πινάκων

Υπάρχουν διάφοροι τρόποι αντιγραφής ενός πίνακα:

- Αντιγραφή αναφοράς: η μεταβλητή που αντιστοιχεί στον δεύτερο πίνακα δείχνει στην διεύθυνση του πρώτου.
  Παράδειγμα:
```
int[] x = new int[3];
int[] y = new int[3];
y = x;
```



  - **Ερώτηση:** Τι γίνεται με τα στοιχεία του δεύτερου πίνακα;
  - **Απάντηση:** Σκουπίδια!
- Αντιγραφή με την μέθοδο **System.arraycopy(<source table>, <source start index>, <dest table>, <dest start index>, <dest end index>)**
  Παράδειγμα: System.arraycopy(x, 0, y, 0, 3);
- Αντιγραφή με την μέθοδο **clone() (ισχύει μόνο για primitive types)**
  Παράδειγμα: `int[] y = x.clone();`

Πίνακες: Χρήσιμες Πληροφορίες

- Πέρασμα πινάκων σαν παραμέτρους σε μεθόδους
  JAVA: Πέρασμα διά τιμής!
  Αντιγράφεται η διεύθυνση του πίνακα στην τοπική μεταβλητή.
  ΠΡΟΣΟΧΗ: Τα στοιχεία του πίνακα μπορούν να αλλαχθούν!

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

- **Ταξινόμηση** πίνακα: **java.util.Arrays.sort**(x);
- **Δυαδική Αναζήτηση** σε πίνακα: **java.util.Arrays.binarySearch**(x, 5);
- **Τύπωμα στοιχείων** πίνακα: **java.util.Arrays.toString(** x);

Πολυδιάστατοι πίνακες

Σύνταξη: **<τύπος δεδομένων>[ ][ ]... <όνομα πίνακα>**, π.χ., int[][] x

Δήλωση Πινάκων

- `int[][]    x;    //Δήλωση πίνακα 2D με ακέραιους`
- `char[][][] a;    //Δήλωση πίνακα 3D με χαρακτήρες`

Αρχικοποίηση Πινάκων

- Η αρχικοποίηση μπορεί να γίνει με τη δήλωση `new`
  `x = new int[2][3]; //Δημιουργία πίνακα με 2 γραμμές και 3 στήλες`
- ή μπορεί να γίνει με την αυτόματη ανάθεση στοιχείων
  `x = { {1, 2, 3}, {4, 5, 6} };` **μόνο κατά την δήλωση!**

Δήλωση και Αρχικοποίηση σε ένα Βήμα

- `int[][]  x= new int[2][3];`

Πολυδιάστατοι πίνακες (συν.)

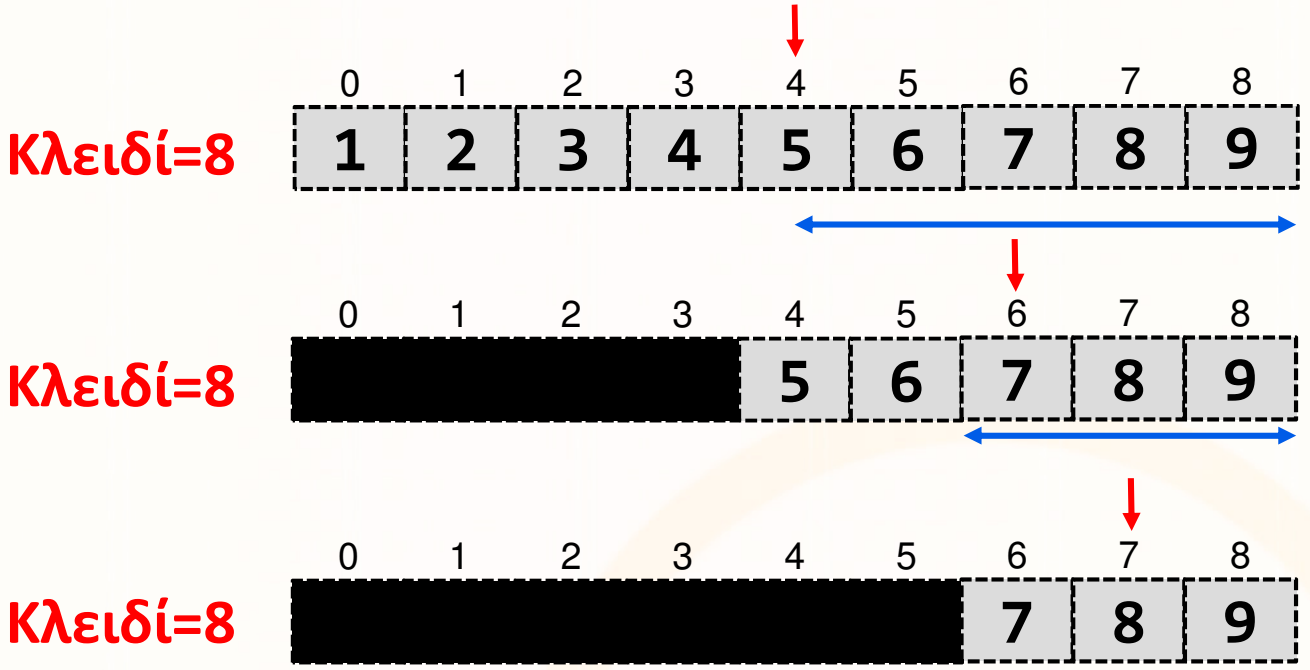
- **Ερώτηση:** Τι επιστρέφει η length σε ένα πολυδιάστατο πίνακα;
- **Απάντηση:** Τον αριθμό των γραμμών
  Παράδειγμα: `int[][] x= new int[2][3]; x.length=2`
- Με την length, μπορούμε να πάρουμε τον αριθμό τον στοιχείων της κάθε γραμμής. Παράδειγμα `x[0].length=3, x[1].length=3`

- **Ερώτηση:** Μπορούμε να έχουμε πίνακες με γραμμές διαφόρων μεγεθών;
- **Απάντηση:** NAI! Ragged Arrays
  Παράδειγμα:

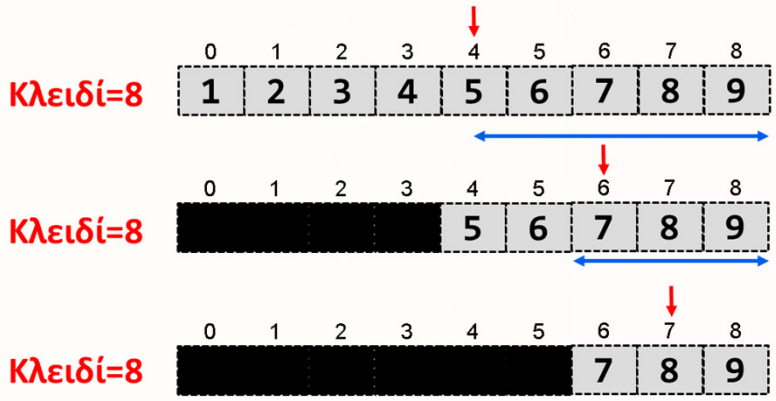
```
int[][] x = { {1}, {2, 3}, {4, 5, 6} };
System.out.println( x.length );      = 3
System.out.println( x[0].length );   = 1
System.out.println( x[1].length );   = 2
System.out.println( x[2].length );   = 3
```

Παραδείγματα: BinarySearch

- BinarySearch: ο αλγόριθμος αυτός βρίσκει ένα στοιχείο σε μία ταξινομημένη λίστα ψάχνωντας κάθε φορά στη μέση του πίνακα. Αν το στοιχείο είναι μεγαλύτερο τότε ο πίνακας μοιράζεται στη μέση και χρησιμοποιείται μόνο το δεξί κομμάτι στο επόμενο βήμα.



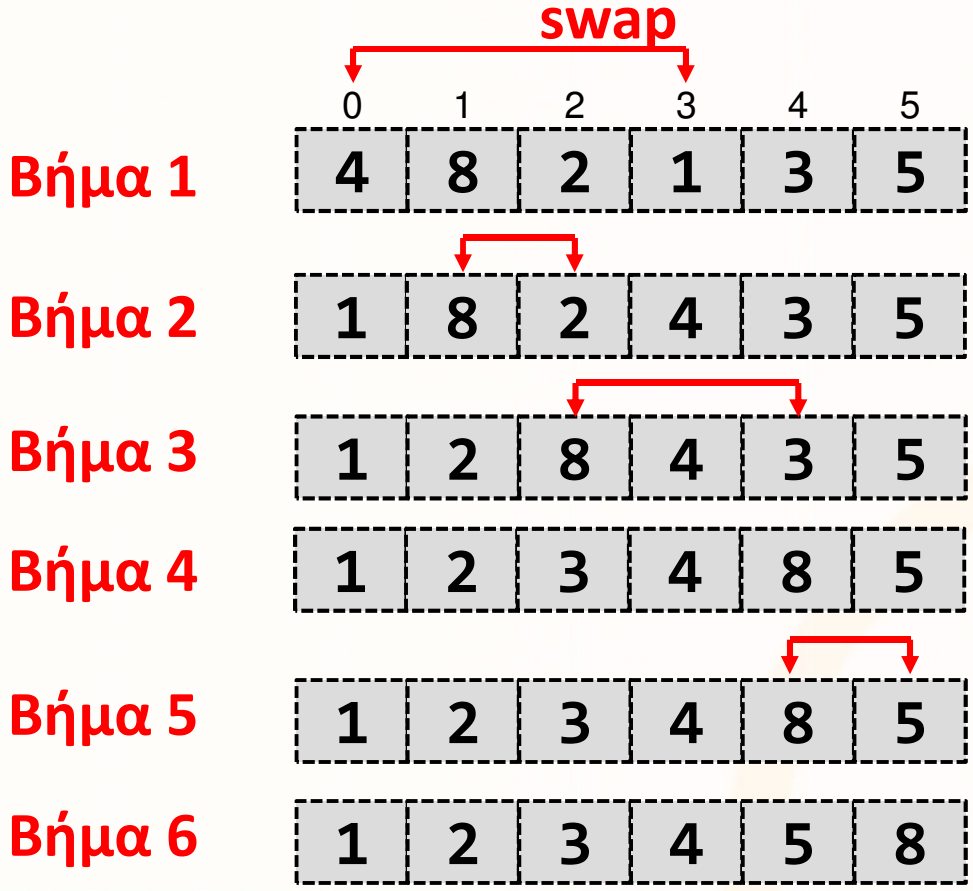
Παραδείγματα: BinarySearch (συν.)




```
int[] x = {1, 2, 3, 4, 5, 6, 7, 8, 9};
int key = 8;
int low=0;
int high = x.length - 1;
int mid;
while( high >= low){
        mid = (low + high) / 2;
        if( key < x[mid])
            high = mid-1;
        else if (key > x[mid])
            low = mid+1;
        else {
            System.out.println(mid);
            break;
        }
}
```

Παραδείγματα: SelectionSort

- SelectionSort: ο αλγόριθμος αυτός ταξινομεί μία λίστα με στοιχεία. Σε κάθε βήμα i βρίσκει το το i-οστό πιο μικρό στοιχείο και το τοποθετεί στην θέση i.
- Παράδειγμα: `int x = {4, 8, 2, 1, 3, 5};`




```
int[] x = {4, 8, 2, 1, 3, 5};
int min_index;
int tmp;
for(int i=0; i<x.length; i++){
  min_index = i;
  for(int j=i+1; j<x.length; j++){
    if(x[j]<x[min_index])
       min_index=j;
  }
tmp = x[i];
x[i] = x[min_index];
x[min_index]=tmp;
}
```

Συλλογές (collections)

- Οι συλλογές (collections) είναι δομές παρόμοιες με πίνακες (arrays) τις εξής διαφορές:
  - Το μέγεθος του μεταβάλλεται δυναμικά
  - Οποιοδήποτε είδος αντικειμένου μπορεί να εισαχθεί σε μία συλλογή
  - ΔΕΝ υποστηρίζουν εισαγωγή αρχέγονων τύπων (π.χ., int).
- Παραδείγματα συλλογών: Vector, BitSet, HashTable, Stack
- **ArrayList**: πολυχρησιμοποιημένη δομή (παρόμοια με Vector)
  - Δεν είναι συγχρονισμένη, αντίθετα με vector
  - Όταν αυξάνει δυναμικά το μέγεθος κατά το μισό του υπάρχον μέγεθος (ο vector αυξάνει κατά το διπλάσιο)
  - Είναι πιο γρήγορη λόγω του ότι δεν είναι συγχρονισμένη

Boxing

- **Οι συλλογές ΔΕΝ δέχονται πρωτόγονους τύπους αλλά μόνο δείκτες σε αντικείμενα (reference types).**
- Ερώτηση: Τι γίνεται όταν χρειαζόμαστε πρωτόγονους τύπους;
- Απάντηση: Boxing

Ορισμοί:

- **Box**
  Ένα στιγμιότυπο μίας κλάσης περιτυλίγματος (wrapper) η οποία αποθηκεύει την τιμή ενός πρωτόγονου τύπου.
- **Boxing**
  Δημιουργία ενός box για μία τιμή πρωτόγονος τύπου
- **Unboxing**
  Επιστροφή της τιμής πρωτόγονου τύπου από το box

Manual/Auto boxing and unboxing

- Οι αρχέγονοι τύποι δεν μπορούν να χρησιμοποιηθούν στις περισσότερες περιπτώσεις -**you need a "wrapper"**
  - myVector.add(new Integer(5));
- Αντίστοιχα δεν επιτρέπεται να χρησιμοποιηθεί ένα αντικείμενο εκεί που χρειάζεται αρχέγονος τύπος. --**you need to "unwrap" it**
  - int n = ((Integer)myVector.lastElement()).intValue();

- Integer iNumber = new Integer(10); → Manual boxing
- Integer iNumber = 10; → Auto-boxing
- iNumber = new Integer(iNumber.intValue()++); → Manual unboxing
- iNumber++; → Auto-unboxing

Χρήσιμες Βιβλιοθήκες (java.lang.Math)

- Σταθερές Κλάσης:
  - PI: το π = 3.1415…
  - E: το e = 2.718…
- Μέθοδοι Κλάσης:
  - Trigonometric Methods: π.χ., sin(double a), cos(double a)
  - Exponent Methods: π.χ., log10(double a), pow(double a, double b), sqrt(double a)
  - Rounding Methods: π.χ., double ceil(double x), double floor(double x)
  - Other methods: min, max, abs, and random

- Παραδείγματα:

```
Math.sin(Math.PI / 2) = 1.0
Math.cos(0) = 1.0

Math.exp(1) = 2.71
Math.log(2.71) = 1.0
Math.pow(2, 3) = 8.0

Math.ceil(2.1) = 3.0
Math.floor(2.1) = 2.0
Math.ceil(-2.1) = -2.0

Math.max(2, 3) = 3
Math.min(2.5, 3.6) = 2.5
Math.abs(-2) = 2
0 <= Math.random() < 1.0
(int)(Math.random() * 10) = [0..9]
```

Χρήσιμες Βιβλιοθήκες (java.io)

- Διαχείριση Αρχείων
  - Create/Open/Delete: π.χ., new File(<path>), delete()
  - Έλεγχοι: π.χ., exists(), canRead(), isDirectory(), isHidden()
  - Ιδιότητες: π.χ., getName(), getAbsolutePath(), getParent(), length()
- Διαχείριση Φακέλων: το ίδιο με Διαχείριση Αρχείων
- Διάβασμα/Εγγραφή σε αρχεία

- Παραδείγματα:

```
//Δημιουργία αρχείου
File f = new File("myfile.txt");
if(!f.exists())
  f.createNewFile();

//Εύρεση του μεγέθους αρχείου
File file = new File("C:/java.txt");
long filesize = file.length();

//Διάβασμα αρχείου
FileInputStream fstream = new
    FileInputStream(" myfile.txt");
DataInputStream in = new
    DataInputStream(fstream);
BufferedReader br = new BufferedReader(new
    InputStreamReader(in));
String strLine;
while ((strLine = br.readLine()) != null) {
   System.out.println (strLine);
}
in.close();
```

Άλλες Χρήσιμες Βιβλιοθήκες

- java.lang.Object: **Ο παππούς** όλων των κλάσεων!
- java.lang.String: Δημιουργία και διαχείριση strings
- java.util.Date: Δημιουργία και διαχείριση πεδίων τύπου ημερομηνία και ώρα
- java.util.ArrayList: Δημιουργία και διαχείριση λιστών διαφόρων τύπων
- java.net: Δημιουργία δικτυακών εφαρμογών
- java.lang.Thread: Δημιουργία πολυνηματικών εφαρμογών
- java.sql: Σύνδεση με βάση δεδομένων και εκτέλεση επερωτήσεων