ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 2: Η ΓΛΩΣΣΑ JAVA

Γενικά Χαρακτηριστικά

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΓΕΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ JAVA

ΙΣΤΟΡΙΑ ΤΗΣ JAVA (1)

- Έργο δημιουργίας δικτύου οικιακών ηλεκτρονικών συσκευών (Sun Microsystems, 1990)
- Αποτέλεσμα (1992)
  - ✓ Εμπορική αποτυχία
  - ✓ Τεχνική επιτυχία → Γλώσσα Oak
- Γλώσσα Java (1995)
- Δημιουργός: James Gosling

ΙΣΤΟΡΙΑ ΤΗΣ JAVA (2)

- Browser HotJava (1995)
  - Υποστήριξη java applets
  - Δύσχρηστος
- Netscape Navigator 2.0 (1995)
  - Υποστήριξη java applets
  - Αποδοχή java
- Γλώσσα Java 1.0 (1996)
- JDK 1.1
- Java SE 1.7.03 (JDK 7, JRE 7)

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ JAVA

- Αντικειμενοστρεφής ή Οντοκεντρική (Object-Oriented)
- Ανεξάρτητη πλατφόρμας (Platform independent)
- Γλώσσα (δια)δικτύου (Internet language)
- Πολυνηματική (Multi-threaded)

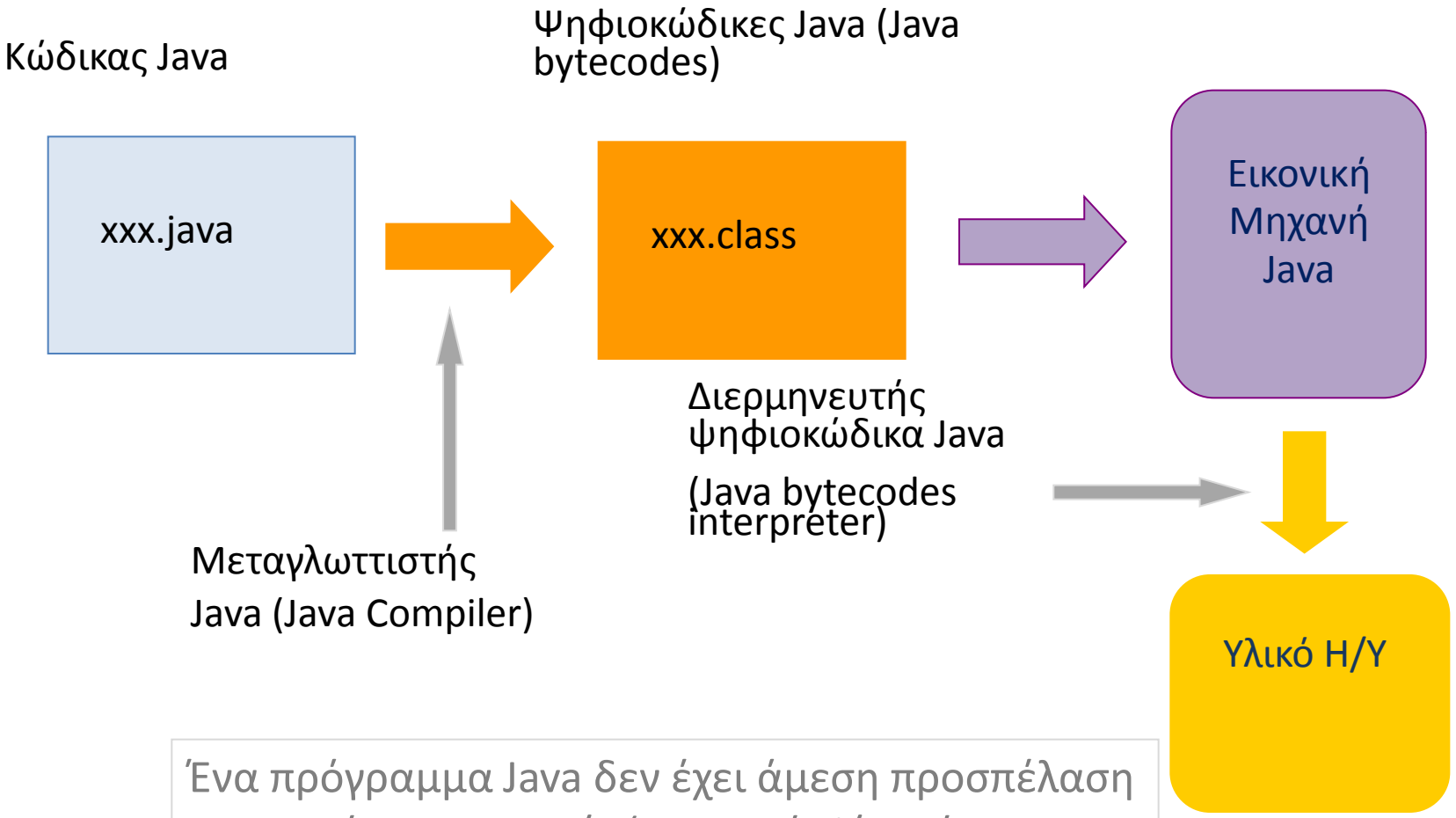
ΕΙΚΟΝΙΚΗ ΜΗΧΑΝΗ JAVA
(JAVA VIRTUAL MACHINE)

- Τα προγράμματα java εκτελούνται στην «εικονική μηχανή java» όχι στον πραγματικό Η/Υ
- Τα προγράμματα java γράφονται για την «εικονική μηχανή java»
- Η «εικονική μηχανή java» λειτουργεί σαν ένα firewall μεταξύ Η/Υ και προγραμμάτων java

ΠΛΕΟΝΕΚΤΗΜΑΤΑ Ε.Μ.J.

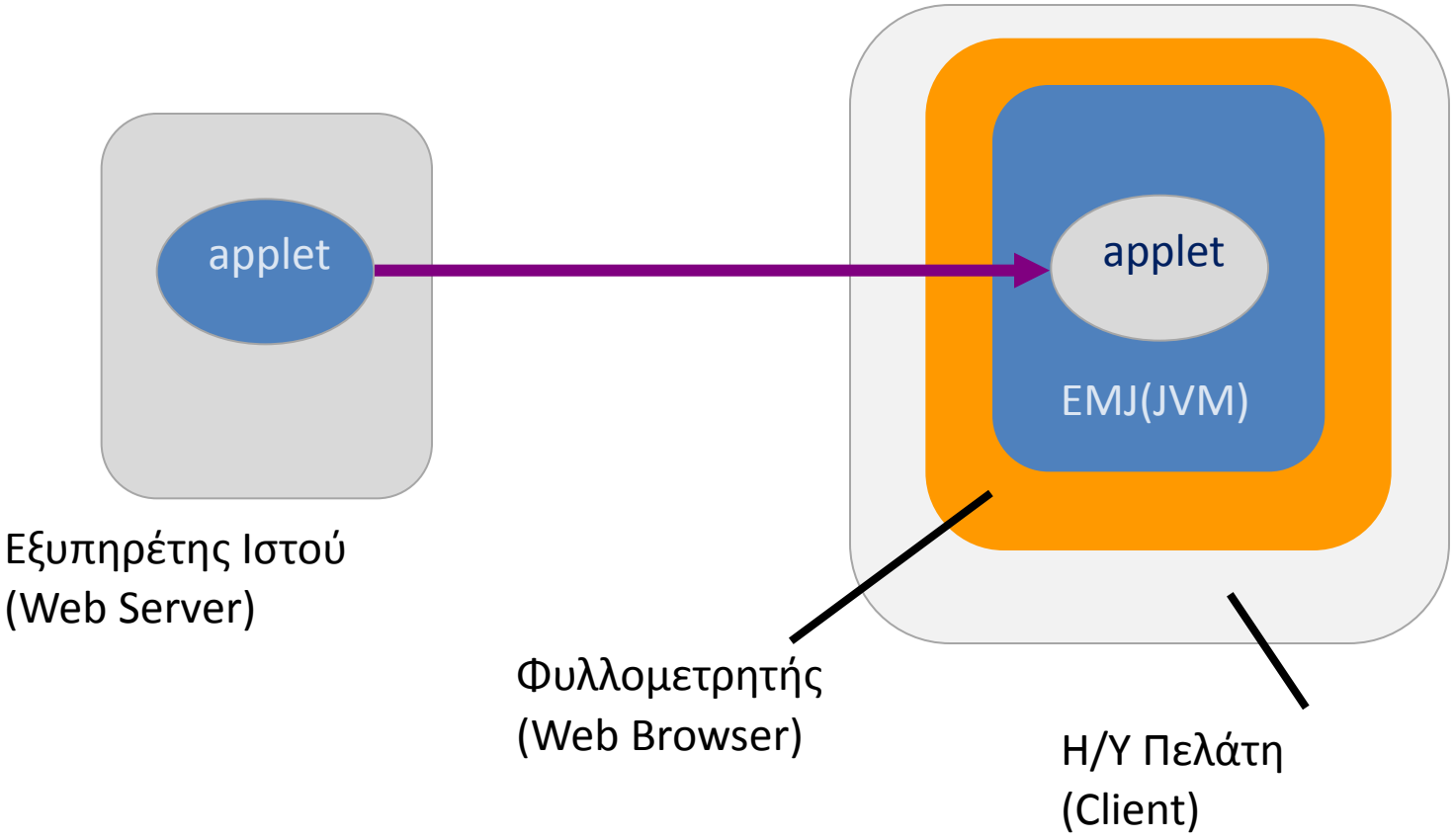
- Ανεξαρτησία από τη μηχανή (προγράμματα ανεξάρτητα πλατφόρμας)
- Ασφάλεια εκτέλεσης προγραμμάτων (π.χ. applets στο διαδίκτυο)

ΜΕΤΑΓΛΩΤΤΙΣΗ-ΕΚΤΕΛΕΣΗ JAVA



Ένα πρόγραμμα Java δεν έχει άμεση προσπέλαση στους πόρους του Η/Υ (συσκευές i/o, σύστημα αρχείων). Μόνο η EMJ έχει.

ΕΚΤΕΛΕΣΗ APPLETS



ΠΡΟΓΡΑΜΜΑΤΑ JAVA

- Αυτόνομες εφαρμογές (stand-alone programs)

1. Συγγραφή πηγαίου κώδικα → Πρόγραμμα java (xxx.java)
2. Μεταγλώττιση προγ/τος (javac xxx.java) → Ψηφιοκώδικας java (xxx.class)
3. Εκτέλεση ψηφιοκώδικα (java xxx) → Αποτελέσματα

- Μικροεφαρμογές (applets)

1. Συγγραφή πηγαίου κώδικα
2. Μεταγλώττιση προγ/τος
3. Δημιουργία αναφοράς σε ιστοσελίδα
4. Φόρτωση ιστοσελίδας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 2: Η ΓΛΩΣΣΑ JAVA

Βασικά Δομικά Στοιχεία

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΔΟΜΙΚΑ ΣΤΟΙΧΕΙΑ

ΔΟΜΙΚΑ ΣΤΟΙΧΕΙΑ JAVA

Βασικά

- Πακέτα (packages)
- Κλάσεις (classes)
- Αντικείμενα/Στιγμιότυπα (objects/instances)
- Μέθοδοι (methods)
- Μεταβλητές (variables)

Μη Βασικά

- Διεπαφές/Διασυνδέσεις (interfaces)
- Εξαιρέσεις (exceptions)
- Νήματα (threads)

ΠΡΟΓΡΑΜΜΑ JAVA

Πρόγραμμα Java = ένα σύνολο ορισμών κλάσεων

Δομή

```
[<Δηλώσεις εισαγωγής κλάσεων
  βιβλιοθήκης>]
[<Δηλώσεις κλάσεων>]
 <Δήλωση βασικής κλάσης>
```

ΠΑΡΑΔΕΙΓΜΑ ΠΡΟΓ/ΤΟΣ

```
import java.lang.*;
```

Δήλωση εισαγωγής κλάσης βιβλιοθήκης

```
class Window {
    protected int size = 1;
    public Window() {
      …..
    }
}
class MWindow extends Window {
    …..
}
```

Ορισμοί κλάσεων

```
public class CheckWindow {
    …..
   public static void main(String args[]){
        …..
   }
}
```

Ορισμός βασικής κλάσης

ΚΛΑΣΕΙΣ

ΒΑΣΙΚΗ ΚΛΑΣΗ

- Κάθε πρόγραμμα java περιέχει μια βασική (ή πρωτεύουσα) κλάση (primary class)
- Είναι η πρώτη κλάση που αναγνωρίζει το περιβάλλον εκτέλεσης της java
- Το αρχείο που περιέχει το πρόγραμμα έχει το ίδιο όνομα
- Περιέχει μια ενσωματωμένη μέθοδο της java, την **main** (που δηλώνεται πάντα pubic static)
- Δηλώνεται πάντα **public**

ΓΕΝΙΚΟΣ ΟΡΙΣΜΟΣ ΚΛΑΣΗΣ

```
<προσδ. κλάσης> class <όνομα κλασης>
{
  <δηλώσεις μεταβλητών>
  <δηλώσεις δημιουργών>
  <δηλώσεις μεθόδων>
}
```

Δήλωση ή Κεφαλίδα κλάσης

Σώμα κλάσης

ΠΡΟΣΔΙΟΡΙΣΤΕΣ ΚΛΑΣΗΣ

- **public:** (μπορεί να χρησιμοποιηθεί από οποιοδήποτε πακέτο)
- **abstract:** (κλάση χωρίς στιγμιότυπα: αφαιρετική ή αφηρημένη κλάση)
- **final:** (κλάση χωρίς υποκλάσεις, μόνο με στιγμιότυπα: τερματική)

ΜΕΤΑΒΛΗΤΕΣ

ΜΕΤΑΒΛΗΤΕΣ–ΕΙΔΗ

- **Μέλους (member)**
  - ✓ **κλάσης** **(π.χ. μετρητής στιγμιοτύπων)**
  - ✓ **στιγμιότυπου**
- **Παράμετροι (parameters)**
- **Τοπικές (local)**

(Η έννοια της καθολικής μεταβλητής δεν υπάρχει με τον ίδιο τρόπο όπως στη C)

ΜΕΤΑΒΛΗΤΕΣ (ΜΕΛΟΥΣ)

- **Δήλωση**

```
<προσδ> <τύπος> <όνομα> [= <τιμή>];
```

- **Προσδιοριστές**

Για μεταβλητές στιγμιοτύπου

- ✓ private (ορατή μόνο από την κλάση της)
- ✓ protected (ορατή από την κλάση της, τις υποκλάσεις της κλάσης της και τις κλάσεις του ίδιου πακέτου)
- ✓ public (ορατή από παντού)

Για μεταβλητές κλάσης

- ✓ static (μεταβλητή κλάσης)
- ✓ final (μεταβλητή με σταθερή τιμή)

ΠΑΡΑΔΕΙΓΜΑΤΑ ΔΗΛΩΣΕΩΝ

προσδιοριστής μεταβλητής

μεταβλητές στιγμιοτύπου

παράμετρος

τοπική μεταβλητή

```
public class Circle {
   private double x, y ;
   private double r ;
   static double biggest_radius;

   public double circumference ( ){
      return 2*3.1416*r ;
   }
   public void increase_radius (double dr) {
      double z;
      z = r + dr;
      this.r = z ;
   }
}
```

ΜΕΘΟΔΟΙ & ΔΗΜΙΟΥΡΓΟΙ

ΓΕΝΙΚΟΣ ΟΡΙΣΜΟΣ ΜΕΘΟΔΟΥ

```
[<προσδ. μεθόδου>] <επιστρ. τύπος>
      <όνομα μεθ.> ([<λίστα παραμ.>])
{
[<δηλώσεις τοπικών μεταβλητών>]

 <προτάσεις java>
}
```

Δήλωση ή κεφαλίδα μεθόδου

Σώμα μεθόδου

Τύποι μεθόδων

- κλάσης
- στιγμιοτύπων

ΠΡΟΣΔΙΟΡΙΣΤΕΣ ΜΕΘΟΔΟΥ

- **public, private, protected, static**
  (όπως για τις μεταβλητές)
- **abstract**
  (σε αφαιρετική κλάση)
- **final**
  (μέθοδος που δεν μπορεί να επικαλυφθεί στις υποκλάσεις της κλάσης-σχετίζεται με την έννοια της κληρονομικότητας)

ΚΛΗΣΗ ΜΕΘΟΔΟΥ– ΠΕΡΑΣΜΑ/ΑΠΟΣΤΟΛΗ ΜΗΝΥΜΑΤΩΝ

- Ο μόνος τρόπος επικοινωνίας αντικειμένων (δηλ. κλήσης μιας μεθόδου για εκτέλεση)
- Σύνταξη

```
<όνομα αντικειμ.>.<όνομα μεθόδου> (<λίστα παραμέτρων>)
```

Π.χ.

```
c.area( ) ;
c.increase_radius(0.5);
Circle.bigger(c1, c2);
```

όπου c στιγμιότυπο και Circle κλάση.

ΠΑΡΑΔΕΙΓΜΑ (1)

Μέθοδος στιγμιοτύπου

```
public class Circle {
    public double x, y, r ;
    public Circle bigger(Circle c){
      if (c.r > r)
         return c;
      else
         return this;
    }
}
```

Αν **c1, c2** δύο στιγμιότυπα της Circle με r 2.0 και 5.0 αντίστοιχα και c μια μεταβλητή τύπου Circle, τότε οι

`c = c1.bigger(c2)` ή `c = c2.bigger(c1)`

δίνουν στη **c** την τιμή **c2** (: η **c** γίνεται αναφορά στο **c2**).

ΠΑΡΑΔΕΙΓΜΑ (2)

Μέθοδος κλάσης

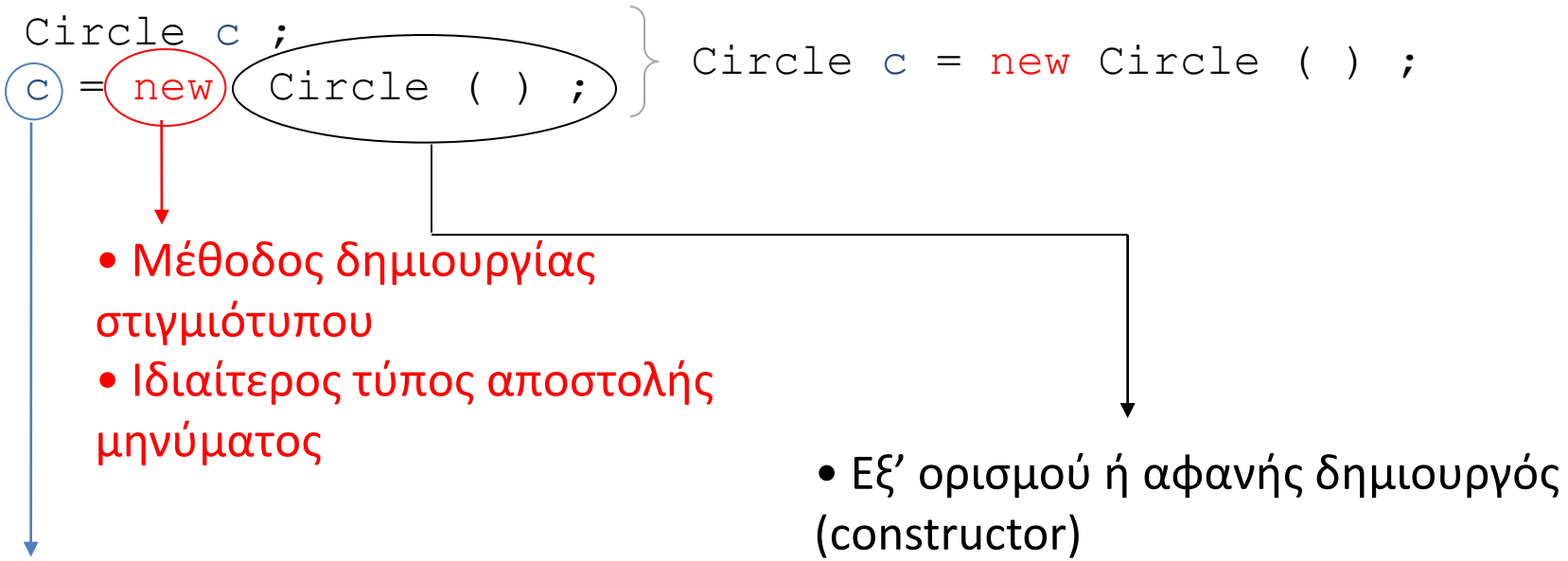
```
public class Circle {
    public double x, y, r ;
    public static Circle bigger(Circle a, Circle b) {
        if (a.r > b.r)
            return a;
        else
            return b;
        }
}
```

Αν **c1, c2** δύο στιγμιότυπα της Circle με r 2.0 και 5.0 αντίστοιχα και c μια μεταβλητή τύπου Circle, τότε η

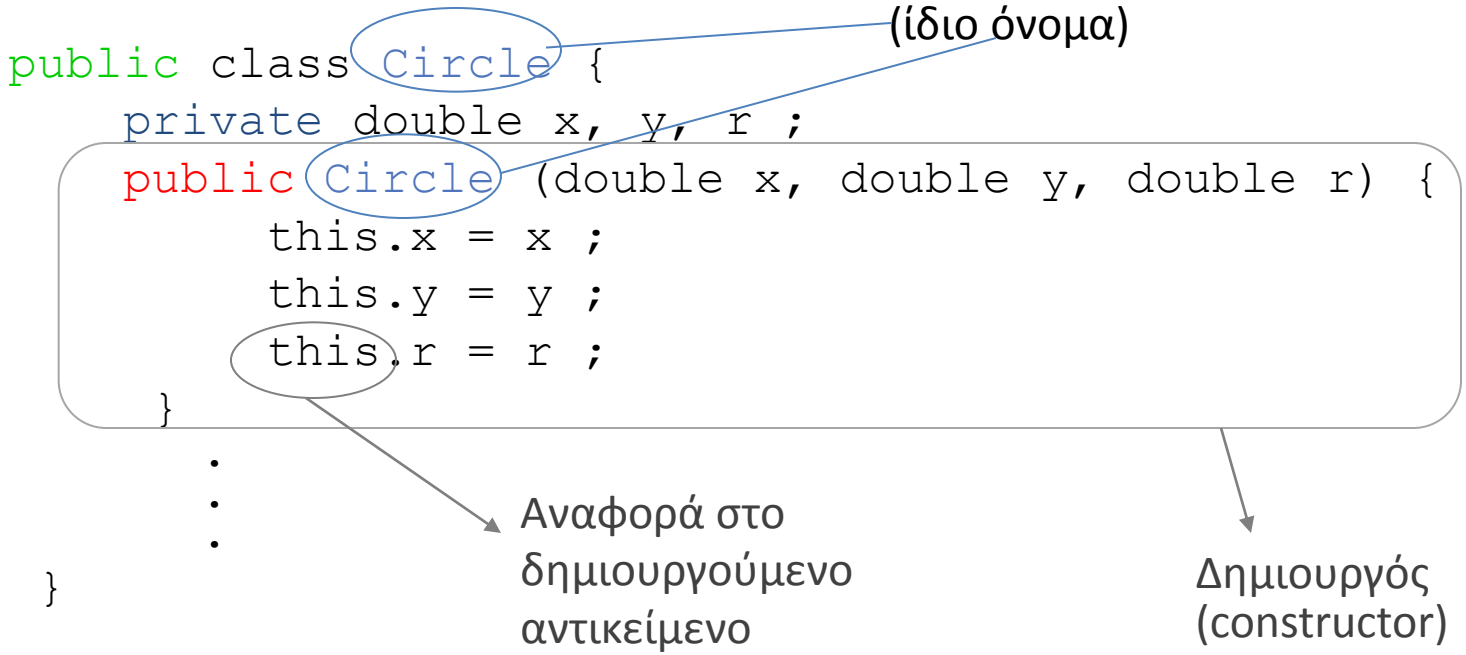
```
c = Circle.bigger(c1, c2)
```

δίνει στη **c** την τιμή **c2** (: η **c** γίνεται αναφορά στο **c2**).

ΔΗΜΙΟΥΡΓΙΑ ΑΝΕΝΕΡΓΟΥ ΣΤΙΓΜΙΟΤΥΠΟΥ



ΔΗΜΙΟΥΡΓΙΑ ΕΝΕΡΓΟΥ ΣΤΙΓΜΙΟΤΥΠΟΥ (ΔΗΜΙΟΥΡΓΟΣ)




```
Circle c = new Circle (10.0, 20.0, 2.0) ;
```

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΔΗΜΙΟΥΡΓΟΥ

- Ίδιο όνομα με την κλάση, προσδιοριστή **public**
- Ειδική κατηγορία συνάρτησης (όχι μέθοδος). Καλείται ανεξάρτητα από την ύπαρξη στιγμιοτύπου της κλάσης
- Δεν ορίζεται επιστρεφόμενη τιμή (όχι void, όχι return)
- Όχι άμεση κλήση σ' ένα πρόγραμμα (μόνο έμμεση, κατά τη δημιουργία αντικειμένου)
- Χρήση της λέξης κλειδί '**this**' (όταν τα ονόματα των ορισμάτων είναι ίδια με αυτά των μεταβλητών της κλάσης)
- Επιστρέφει έμμεσα την τιμή αναφοράς του '**this**'

ΔΗΜΙΟΥΡΓΙΑ ΣΤΙΓΜΙΟΤΥΠΟΥ ΧΩΡΙΣ ΔΗΜΙΟΥΡΓΟ

- Δήλωση μεταβλητών ως public
- Αρχικοποίηση μέσω εντολών της μορφής

  <στιγμιότυπο>.<μεταβλητή>

```
public class Circle {
    public double x, y, r ;
}
```

```
Circle c = new Circle () ;
c.x = 10.0;
c.y = 20.0;
c.r = 2.0;
```

ΠΟΛΛΑΠΛΟΙ ΔΗΜΙΟΥΡΓΟΙ

```
public class Circle {
    private double x, y, r ;
    public Circle (double x, double y, double r) {
        this.x = x ; this.y = y ; this.r = r ;}
    public Circle (double r) {
        x = 0.0 ; y = 0.0 ; this.r = r ;}
    public Circle () {
        x = 0.0 ; y = 0.0 ; r = 1.0 ;}
    public Circle (Circle c) {
        x = c.x ; y = c.y ; r = c.r ;}
}
```

Υπερφόρτωση Μεθόδων (method overloading)

(αναγνώριση μεθόδου όχι μόνο από το όνομα, αλλά και τον τύπο των ορισμάτων)

ΤΕΛΕΣΤΗΣ 'THIS' – ΕΠΑΝΑΧΡΗΣΙΜΟΠΟΙΗΣΗ

```
public class Circle {
    private double x, y, r ;
    public Circle (double x, double y, double r) {
        this.x = x ;
        this.y = y ;
        this.r = r ;
    }
    public Circle (double r) {
        this (0.0 , 0.0 , r );
    }
    public Circle () {
        this (0.0, 0.0, 1.0) ;
    }
    public Circle (Circle c) {
        this (c.x, c.y, c.r) ;
    }
}
```

Ο τελεστής **'this'** στο σώμα μιας μεθόδου μεταφέρει τον έλεγχο στη μέθοδο με το ίδιο όνομα και αντίστοιχα ορίσματα.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 2: Η ΓΛΩΣΣΑ JAVA

Σύγκριση JAVA-C

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΣΥΓΚΡΙΣΗ JAVA - C

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ (1)

Πρωτογενείς τύποι

- Οι μεταβλητές περιέχουν τις τιμές τους
- Ίδιοι με αυτούς της C (char, short, int, long, float, double)
- Διαφορές:
  - ✓ Υποστηρίζει boolean (true, false), byte (-128, 127). Μεταβλητές τύπου boolean δεν μπορούν να θεωρηθούν σαν ακέραιοι.
  - ✓ Δεν υποστηρίζει μη προσημασμένους αριθμούς
  - ✓ Δεν υποστηρίζει τύπο δείκτη

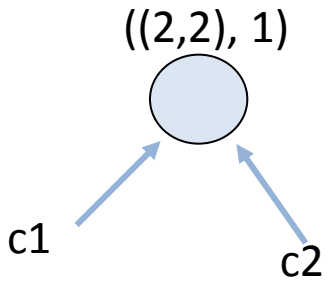
ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ (2)

Τύποι Αναφοράς

- Οι μεταβλητές είναι μια αναφορά στην τιμή τους (περιέχουν τη διεύθυνσή της)
- αντικείμενα, πίνακες
- Κάθε αναφορά που δεν αναφέρεται σε αντικείμενο ή πίνακα, έχει τιμή **null**

```
double d;
Circle c1, c2;
c1 = new Circle (2.0, 2.0, 1.0);
c2 = c1;
c1.r = 4.0;
d = c2.r;
```

```
double x=10;
double y = x;
x = 4.0;
```



ΑΝΤΙΓΡΑΦΗ-ΣΥΓΚΡΙΣΗ ΑΝΤΙΚΕΙΜΕΝΩΝ

Αντιγραφή

- Δεν γίνεται με ανάθεση/καταχώρηση
- Χρήση της μεθόδου clone (Cloneable Interface)

```
Circle c1 = new Circle();
c2 = c1.clone();
```

Σύγκριση

- Δεν γίνεται με έλεγχο ισότητας

```
c1 == c2
```

- Δημιουργία μεθόδου σύγκρισης

ΠΙΝΑΚΕΣ (1)

Ορισμός (αντικείμενα που περιέχουν άλλα αντικείμενα)

```
int x[];   ή  int [] x;
```

→ 1) Δημιουργία αναφοράς

```
x= new int[5];
```

→ 2) Δημιουργία αντικειμένου και

→ 3) Ανάθεση στην αναφορά

```
int x []= new int[5];   ή   int [] x = new int[5];
```

```
x[0] = 2;
x[1] = 3;
```

→ Ανάθεση τιμών στον πίνακα

```
int x [] = {2, 3, 5, -6, 9};
```

(Δημιουργία και αρχικοποίηση)

ΠΙΝΑΚΕΣ (2)

Μέγεθος πίνακα: μεταβλητή length
(η μόνη μεταβλητή της κλάσης πίνακας)

π.χ. `x.length` δίνει το μέγεθος του πίνακα **x**

Η **πρόσβαση** στα στοιχεία ενός πίνακα γίνεται όπως και στη C.

```
int x [] = new int[10];
x[0] = 0;
for (int i=1; i < x.length; i++)
   x[i] = i + x[i-1];
```

ΠΙΝΑΚΕΣ (3)

Πολυδιάστατοι πίνακες

```
boolean mat [] [] = new boolean [3] [4];
int y [] [] [] = new int [3] [4] [5];
```

`mat.length` (πρώτη διάσταση)

`mat[1].length` (δεύτερη διάσταση)

ΑΛΦΑΡΙΘΜΗΤΙΚΑ (1)

C → πίνακες χαρακτήρων

Java → στιγμιότυπα της κλάσης `String` ή `StringBuffer`
(δύο ανεξάρτητες κλάσεις)

Όταν θέλουμε να μπορούμε να τροποποιήσουμε τα αλφαριθμητικά.

Δημιουργία

- Με αυτόματη αναγνώριση-δημιουργία
  π.χ. String s1 = "Hello";

- Αποστολή του μηνύματος new στην κλάση String
  π.χ. String s2 = new String();
  String s3 = new String(s1);

(δημιουργοί)

- Αποστολή του μηνύματος new στην κλάση StringBuffer
  π.χ. StringBuffer s4 = new StringBuffer();
  StringBuffer s5 = new StringBuffer(s1);

(δημιουργοί)

ΑΛΦΑΡΙΘΜΗΤΙΚΑ (2)

Βασικές μέθοδοι

- **`length()`** (επιστρέφει τον αριθμό χαρακτήρων)
  π.χ. `s1.length();` και όχι ~~`s1.length;`~~
- **`charAt(int i)`** (επιστρέφει τον χαρακτήρα στη θέση **i**)
  π.χ. `s1.charAt(1);` → `'e'`

Πρόσθεση αλφαριθμητικών- τελεστής **"+"**

`s1 + "John"` → `"Hello John"`

ΤΕΛΕΣΤΕΣ

- Η Java υποστηρίζει όλους σχεδόν τους τελεστές της C (αριθμητικούς, σύγκρισης, λογικούς) με την ίδια προτεραιότητα.

  - Η Java **δεν υποστηρίζει**
    - ✓ τον τελεστή κόμμα (,) για συνδυασμό εκφράσεων
    - ✓ τους τελεστές δεικτών * , &
    - ✓ τον τελεστή sizeof

  - Η Java **υποστηρίζει επί πλέον**
    - ✓ τον τελεστή + (και +=) για συνένωση αλφαριθμητικών
    - ✓ τον τελεστή **instanceof** (π.χ. `s1 instanceof String`)
    - ✓ τους τελεστές &, | (AND, OR αντίστοιχα σε boolean) (πρβλ. &&, ||)

ΠΡΟΤΑΣΕΙΣ ΕΛΕΓΧΟΥ ΡΟΗΣ

if-else, while, do-while, for

- ίδια σύνταξη
- προσοχή στην <έκφραση-συνθήκη> (τύπου boolean, όχι int ή άλλου τύπου)

```
int i = 10 ;
while (i--) {
  Circle c1 = new Circle ();
  if (c1){
    int j;
    do {
          :
          :
    } while (j);
   }
}
```

```
int i = 10 ;
while (i-- > 0) {
 Circle c1 = new Circle ();
 if (c1!= null)  {
   int j;
   do {
         :
         :
   } while (j != 0);
  }
}
```

ΔΗΛΩΣΕΙΣ ΠΑΚΕΤΟΥ

Ορισμός πακέτου

package <όνομα πακέτου>; (πρώτη πρόταση στο αρχείο)

Π.χ. `package transport ;` (προεραιτικά-ανώνυμο πακέτο)

Εισαγωγή στοιχείων πακέτου

import <περιγραφή>;

Π.χ.
```
import java.awt.Graphics;
import java.awt.*;
import transport.car;
```

Προσδιορισμός στοιχείων πακέτων Java

java.<όνομ-πακέτου>.<όνομ-κλάσης>.<όνομ-μεθόδου>

Π.χ. `Java.lang.String.substring()`

ΕΙΣΟΔΟΣ–ΕΞΟΔΟΣ

Είσοδος ορισμάτων από γραμμή εντολών

java <όνομα-αρχείου> <ορίσματα>

Π.χ. **java PrintArgs Kyprou 15 Patra**

```
class PrintArgs {
  public static void main(String args[]) {
   for (int i = 0; i < args.length; i++)
      System.out.println("Argument " + (i+1) + ": " + args[i]);
   }
  }
}
```

Argument 1: Kyprou
Argument 2: 15
Argument 3: Patra

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 2: Η ΓΛΩΣΣΑ JAVA

Βιβλιοθήκες

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΒΙΒΛΙΟΘΗΚΗ JAVA

ΒΑΣΙΚΗ ΒΙΒΛΙΟΘΗΚΗ JAVA

- Ένα σύνολο κλάσεων και διεπαφών οργανωμένων σε πακέτα
- Βασικά πακέτα
  - ✓ **java.applet**
  - ✓ **java.awt**
  - ✓ **java.io**
  - ✓ **java.lang**
  - ✓ **java.math**
  - ✓ **java.net**
- Για να χρησιμοποιήσουμε κλάσεις ή μεθόδους κλάσεων από τα πακέτα της βιβλιοθήκης (πλην αυτών του java.lang), πρέπει να τις εισάγουμε στο πρόγραμμά μας με την εντολή import

ΚΛΑΣΗ System (1)

```
public final class System extends Object
```

- Η κλάση **System** περιέχει τη βασική μεταβλητή εξόδου **out** (μεταβλητή κλάσης)
- Η out είναι τύπου **PrintStream** (μια άλλη κλάση του πακέτου)

Η γνωστή εντολή εκτύπωσης στην οθόνη

`System.out.println`

ουσιαστικά αποτελεί αποστολή μυνήματος (println) σ' ένα στιγμιότυπο της **PrintStream**, του οποίου αναφορά είναι η μεταβλητή **out** της **System**.Το στιγμιότυπο δημιουργείται αυτόματα από το σύστημα.

ΚΛΑΣΗ System (2)

- Η **println** είναι μέθοδος στιγμιοτύπων της κλάσης PrintStream
- Αν η **println** ήταν μέθοδος κλάσης, δεν θα χρειαζόταν να στείλουμε μήνυμα σε στιγμιότυπο της **PrintStream**

Π.χ. οι μέθοδοι `toString, valueOf` είναι μέθοδοι κλάσης. Οπότε μπορώ να στείλω κατ' ευθείαν μήνυμα στην κλάση τους.

ΚΛΑΣΗ Double (1)

```
public final class Double extends Number
                                implements Comparable
```

- Προσοχή!!! Άλλο η κλάση Double,
  άλλο ο πρωτογενής τύπος double
- Περιέχει ένα απλό πεδίο (μεταβλητή) τύπου double

ΚΛΑΣΗ Double (2)

Μέθοδοι (κλάσης)

`static String toString (double d)`

(δημιουργεί στιγμιότυπο της String που είναι η αλφαριθμητική αναπαράσταση του d, επιστρέφει αναφορά στο στιγμιότυπο)

Π.χ.

```
double d1 = 1821;
String myString;
myString = Double.toString(d1);
System.out.println(myString);
```

"1821"

ΚΛΑΣΗ Double (3)

`static Double valueOf (String s)`

(δημιουργεί στιγμιότυπο της Double με αρχική τιμή την αριθμητική αναπαράσταση του s)

Π.χ.

```
Double myDouble;
String s1 = "1821";
myDouble = Double.valueOf(s1);
System.out.println(myDouble);
```

"1821"

ΔΗΜΙΟΥΡΓΙΑ ΣΤΙΓΜΙΟΤΥΠΟΥ (ΧΩΡΙΣ NEW)

Με τη χρήση μεθόδου που επιστρέφει αναφορά σε στιγμιότυπο του ζητούμενου τύπου.

Π.χ.

```
Double myDouble = Double.valueOf("2000")
```

Προσοχή!!!

```
Double myDouble;
double d1 = myDouble.doubleValueOf();
```

Η doubleValueOf() είναι μέθοδος στιγμιοτύπου και επιστρέφει double.

Δεν αναφέρεται σε συγκεκριμένο στιγμιότυπο

```
Double myDouble;
myDouble = new Double ("2000");
double d1 = myDouble.doubleValueOf();
```

ΚΛΑΣΗ String (1)

```
public final class String extends Object
                        implements Serializable, Comparable
```

(Όλες οι αλφαριθμητικές σταθερές, π.χ. "abc", υλοποιούνται σαν στιγμιότυπά της)

Μέθοδοι

`int length ()`

(επιστρέφει τον αριθμό χαρακτήρων)

`char charAt(int index)`

(επιστρέφει τον χαρακτήρα στη θέση index)

ΚΛΑΣΗ String (2)

`indexOf(char ch)`

(επιστρέφει την πρώτη θέση που βρίσκεται ο ch, αλλιώς -1)

`lastIndexOf(char ch)`

(επιστρέφει την τελευταία θέση που βρίσκεται ο ch, αλλιώς -1)

`equals(String s)`

(επιστρέφει'true', αν το ίδιο αντικείμενο με αυτό που καλεί τη μέθοδο, αλλιώς 'false')

`replace(char oldChar, char newChar)`

(επιστρέφει ένα νέο String, ίδιο με αυτό που καλεί, αλλά με newChar όπου oldChar)

ΚΛΑΣΗ StringBuffer

Μέθοδοι

`char setCharAt(int x, char newChar)`
(αλλάζει τον χαρακτήρα στη θέση x με τον newChar)

`replace(char oldChar, char newChar)`
(αλλάζει με newChar όπου oldChar)

Προσοχή!!! Διαφορά από την αντίστοιχη της String:
Δεν δημιουργεί νέο String,
αλλάζει αυτό που καλεί τη μέθοδο.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 2: Η ΓΛΩΣΣΑ JAVA

Κληρονομικότητα

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ

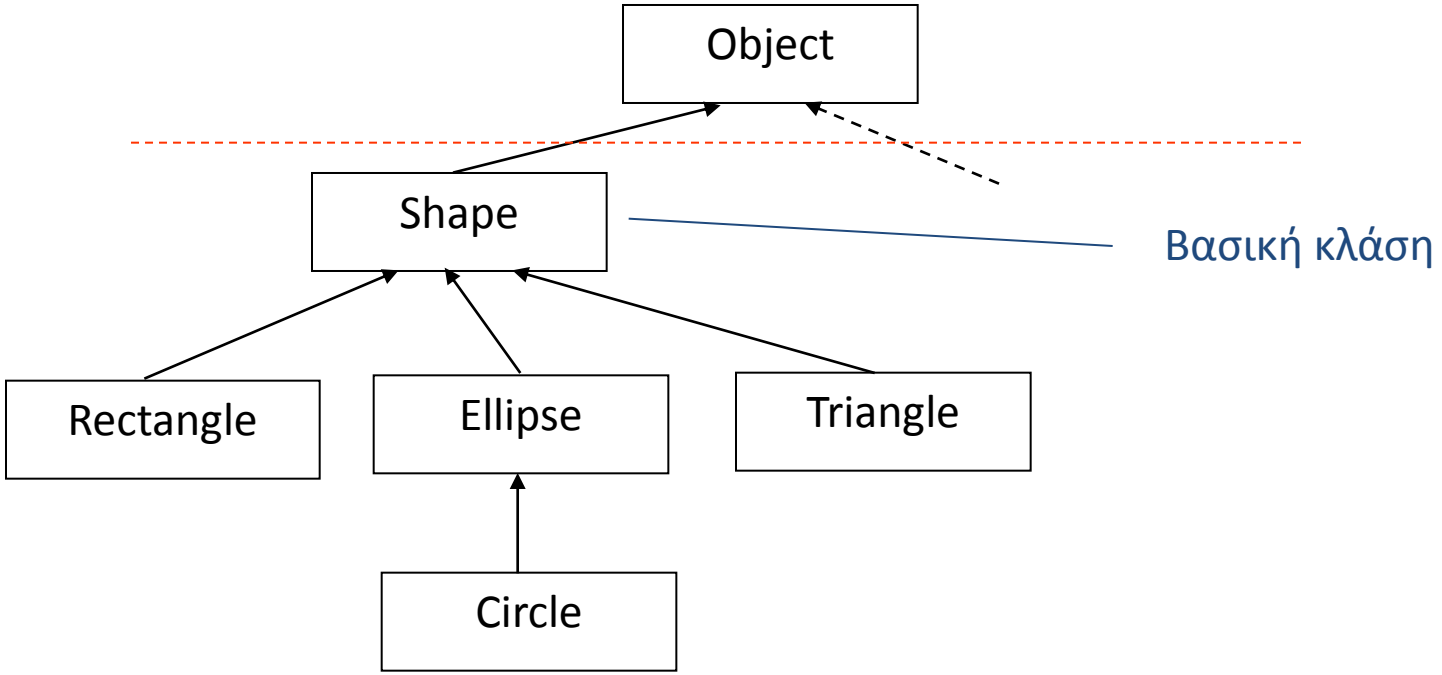
ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ

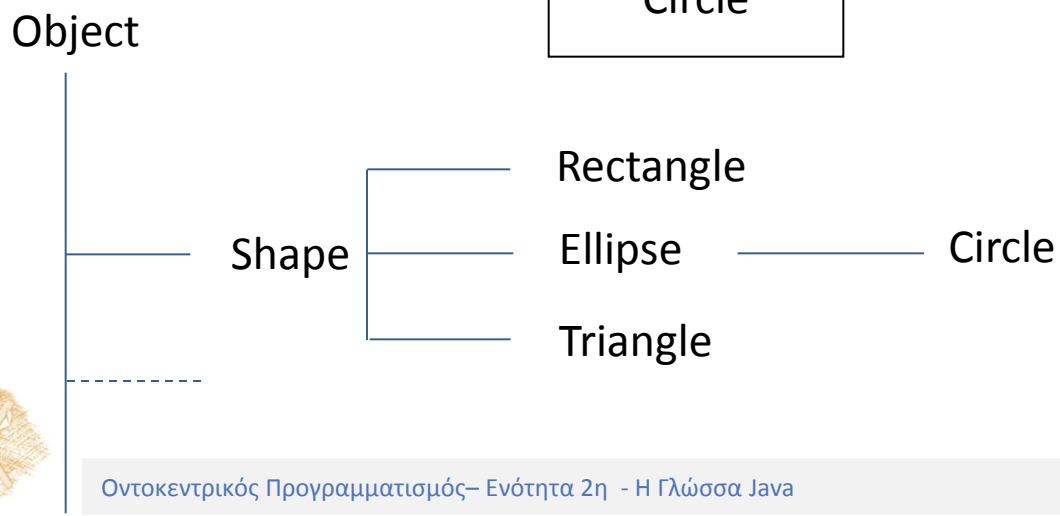
- Μηχανισμός υλοποίησης των σχέσεων γενίκευσης/εξειδίκευσης μεταξύ κλάσεων
- Η σχέση εξειδίκευσης «υποκλάση-της» (subclass-of) είναι γνωστή σαν σχέση «είναι ένα» (isa) ή «είναι ένα είδος» (ako: a kind of)
- Σχετίζεται με τη σχεδίαση του προγράμματος
- Πλεονέκτημα: αύξηση επαναχρησιμοποίησης

ΑΠΛΗ ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ

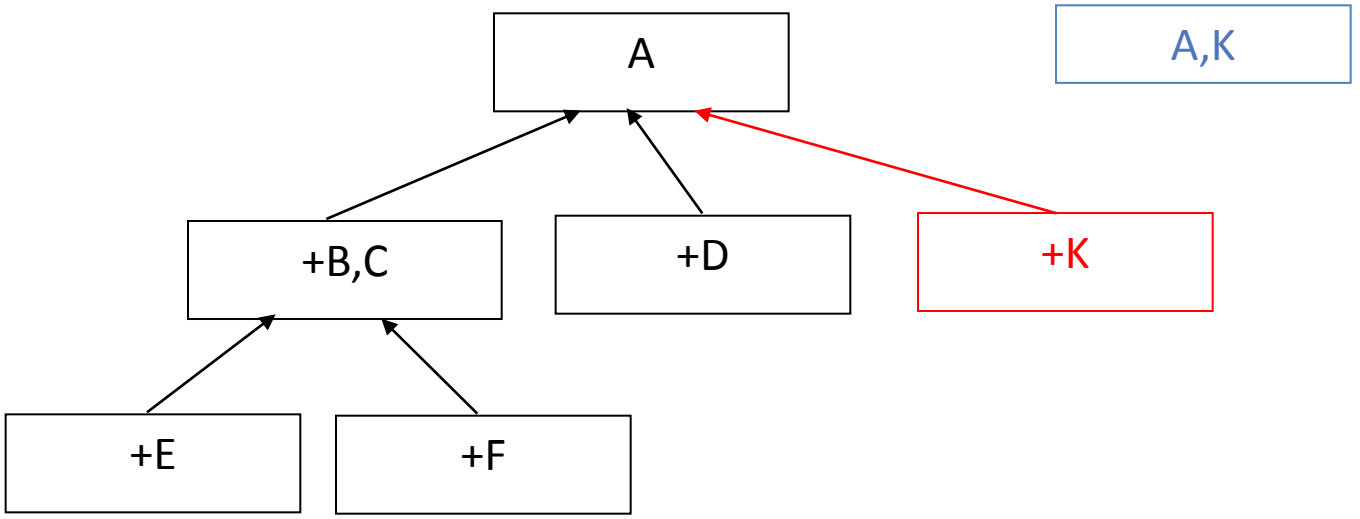
- Μια κλάση είναι **υποκλάση** μιας μόνο κλάσης
- Η κλάση (υποκλάση) κληρονομεί μεταβλητές και μεθόδους από την (άμεση) **υπερκλάση** της και τις (έμμεσες) υπερκλάσεις αυτής
- Η **ιεραρχία**/δέντρο κλάσεων ονομάζεται και ιεραρχία/ δέντρο κληρονομικότητας. Η ρίζα του δέντρου ονομάζεται **βασική κλάση** (base class)
- Σ'ένα πρόγραμμα συνήθως έχουμε περισσότερες από μια βασικές κλάσεις, επομένως και δέντρα κληρονομικότητας
- Όλες οι βασικές κλάσεις είναι υποκλάσεις της κλάσης **Object**

ΠΑΡΑΔΕΙΓΜΑ


Object
Shape
Βασική κλάση
Rectangle
Ellipse
Triangle
Circle

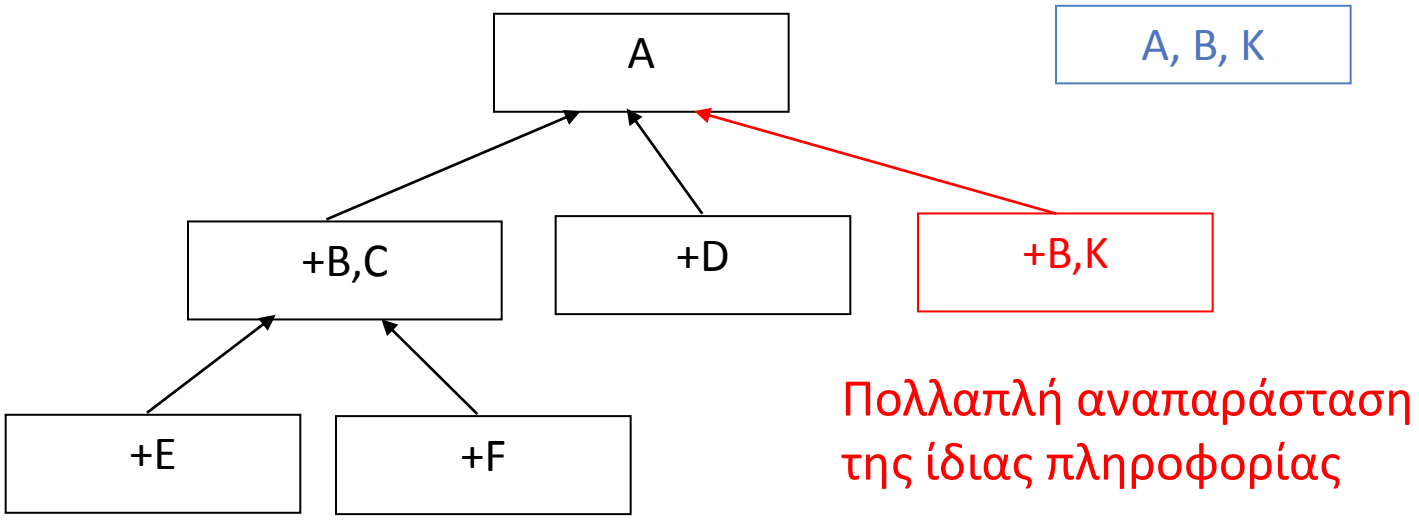

Object
Shape
Rectangle
Ellipse
Triangle
Circle

ΠΡΟΣΘΗΚΗ ΝΕΑΣ ΚΛΑΣΗΣ: ΧΩΡΙΣ ΑΝΑΔΟΜΗΣΗ ΙΕΡΑΡΧΙΑΣ



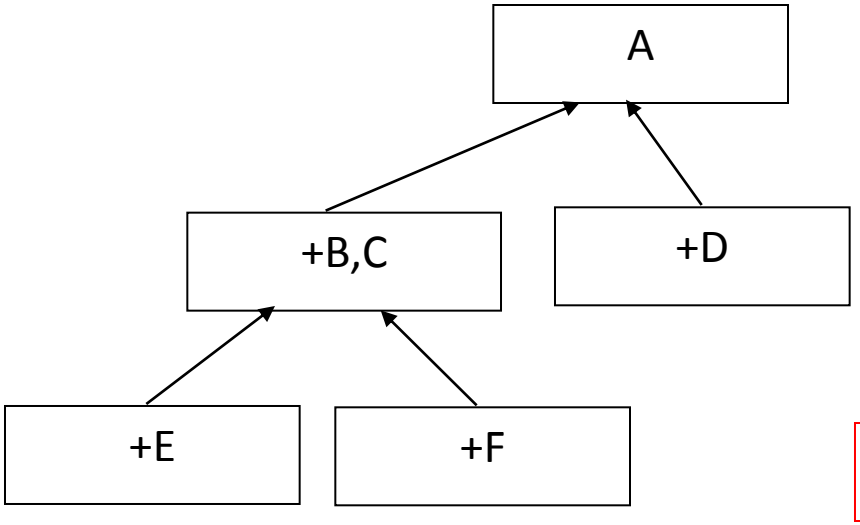
Προσθήκη μιας κλάσης με χαρακτηριστικά A, K

ΠΡΟΣΘΗΚΗ ΝΕΑΣ ΚΛΑΣΗΣ: ΜΕ ΑΝΑΔΟΜΗΣΗ ΙΕΡΑΡΧΙΑΣ (1)

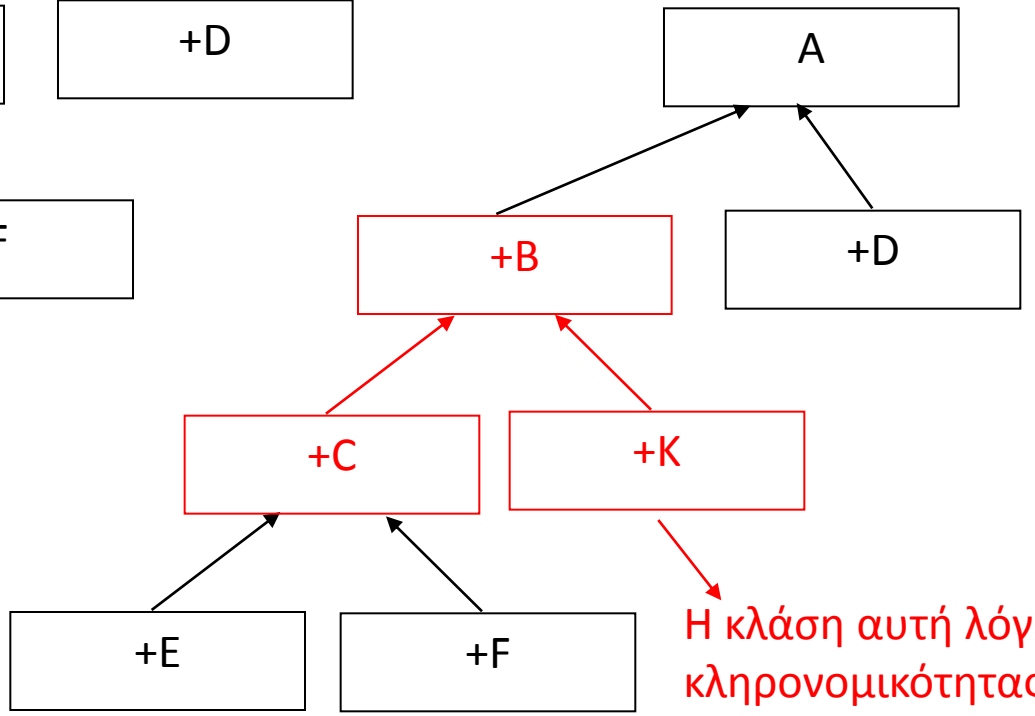


Προσθήκη μιας κλάσης με χαρακτηριστικά A, B, K

ΠΡΟΣΘΗΚΗ ΝΕΑΣ ΚΛΑΣΗΣ: ΜΕ ΑΝΑΔΟΜΗΣΗ ΙΕΡΑΡΧΙΑΣ (2)



A, B, K



Απαιτείται αναδόμηση της ιεραρχίας

Η κλάση αυτή λόγω κληρονομικότητας έχει ως χαρακτηριστικά A, B, K

ΔΗΛΩΣΕΙΣ ΣΧΕΣΕΩΝ ΙΕΡΑΡΧΙΑΣ

```
<προσδ. κλάσης> class <όνομα κλασης>
  [extends <όνομα κλάσης>]
{
 <δηλώσεις μεταβλητών>
 <δηλώσεις δημιουργών>
 <δηλώσεις μεθόδων>
}
```

υπερκλάση

ΠΑΡΑΔΕΙΓΜΑ

```
public class Circle {
    protected double x, y, r ;
    public Circle (double x, double y, double r)
       {this.x=x; this.y=y; this.r = r ;}
    public double area ( )
       {return 3.1416*r*r ;}
}

public class GraphicCircle extends
Circle{
    Color outline, fill ;
    public void draw ( )
       {          …                    }
}
```

ΠΟΛΥΜΟΡΦΙΣΜΟΣ

- Το γεγονός ότι ο αποστολέας ενός μηνύματος δεν χρειάζεται να γνωρίζει την κλάση του παραλήπτη (στιγμιοτύπου)
- Το γεγονός ότι μια λειτουργία μπορεί να υλοποιηθεί με το ίδιο όνομα αλλά με διαφορετικό περιεχόμενο (τρόπο λειτουργίας) σε διαφορετικές κλάσεις.

ΥΠΕΡΦΟΡΤΩΣΗ–ΥΠΕΡΚΑΛΥΨΗ ΜΕΘΟΔΩΝ

- Είναι δυνατή η ύπαρξη μεθόδων με το ίδιο όνομα, αλλά διαφορετικά ορίσματα (είτε ως προς τον τύπο είτε ως προς τον αριθμό) στην ίδια ή διαφορετική κλάση **(υπερφόρτωση μεθόδων-method overloading)**.
- Κάθε μέθοδος (μεταβλητή) χαμηλότερα στην ιεραρχία υπερκαλύπτει (επισκιάζει) κάθε ίδια μέθοδο (μεταβλητή) που βρίσκεται υψηλότερα στην ιεραρχία **(method overriding/variable shadowing).**

ΑΦΗΡΗΜΕΝΕΣ Ή ΑΦΑΙΡΕΤΙΚΕΣ ΚΛΑΣΕΙΣ

- Κλάσεις που χρειάζονται στο σχεδιασμό (κυρίως στα ανώτερα επίπεδα ιεραρχίας), αλλά δεν αναφέρονται σε πραγματικά στιγμιότυπα/οντότητες
- Μια αφαιρετική κλάση περιέχει τουλάχιστον μια αφαιρετική μέθοδο (δηλ. μέθοδο χωρίς σώμα)
- Η απόγονος μιας αφαιρετικής κλάσης δεν είναι αφαιρετική αν ορίζει τα σώματα όλων των αφαιρετικών μεθόδων της προγόνου της

ΠΑΡΑΔΕΙΓΜΑ

```
abstract class Shape {}
class Rectangle extends Shape {}
class Ellipse extends Shape {}
class Triangle extends Shape {}
class Circle extends Ellipse {}

abstract class Shape {
  public abstract double area();
  public abstract double circumference();
}
```

ΠΟΛΛΑΠΛΗ ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ

- Μια κλάση είναι υποκλάση περισσότερων της μιας κλάσης
- Μια κλάση (υποκλάση) κληρονομεί μεταβλητές και μεθόδους ταυτόχρονα από όλες τις υπερκλάσεις της (άμεσες και έμμεσες)
- Η Java **δεν** υποστηρίζει πολλαπλή κληρονομικότητα με άμεσο τρόπο (μόνο έμμεσα, μέσω των διεπαφών)

ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ ΚΑΙ ΔΗΜΙΟΥΡΓΟΙ

```
public class GraphicCircle extends Circle{
    Color outline, fill ;
    public GraphicCircle (double x, double y, double r,
                          Color outline, Color fill) {
      this.x = x; this.y=y; this.r=r;
      this.outline=outline; this.fill=fill;
}
```

```
public class GraphicCircle extends Circle{
    Color outline, fill ;
    public GraphicCircle (double x, double y, double r,
                          Color outline, Color fill) {
      super(x, y, r);
      this.outline=outline;
      this.fill=fill;
}
```

ΛΕΞΗ-ΚΛΕΙΔΙ SUPER (1)

- Σε κάθε δημιουργό εισάγεται από το σύστημα σαν πρώτη πρόταση στο σώμα του η πρόταση " super(); ", εφ' όσον δεν υπάρχει άλλη πρόταση super.
- Η πρόταση " super(); " καλεί τον εξ'ορισμού δημιουργό της (άμεσης) υπερκλάσης της κλάσης του δημιουργού και μετά εκτελείται το (υπόλοιπο) σώμα του δημιουργού.
- Η κλήση του εξ' ορισμού δημιουργού δεν έχει κανένα ουσιαστικό αποτέλεσμα και δεν μας απασχολεί, εκτός αν έχουμε ορίσει εμείς δημιουργό χωρίς ορίσματα στην υπερκλάση, οπότε καλείται αυτός.

ΠΑΡΑΔΕΙΓΜΑ

```
class Parent{
    public Parent () {
    System.out.println ("Hello Parent");}}

class Child extends Parent {
   String message = "No Child";
   public Child (String message) {
      this.message = message;
      System.out.println (message);}}
```

Τι αποτέλεσμα θα έχει η `Child c = new Child("First Child");`

1. Εκτύπωση: Hello Parent (λόγω έμμεσου super();)
2. Ανάθεση: "First Child" στην message του c (λόγω)
3. Εκτύπωση: First Child (λόγω)

ΛΕΞΗ-ΚΛΕΙΔΙ SUPER (2)

- Η χρήση της super δεν αφορά μόνο τους δημιουργούς, αλλά και τις μεθόδους
- Μέσω της super μπορούμε να καλέσουμε απ' ευθείας μια μέθοδο της υπερκλάσης μιας κλάσης:

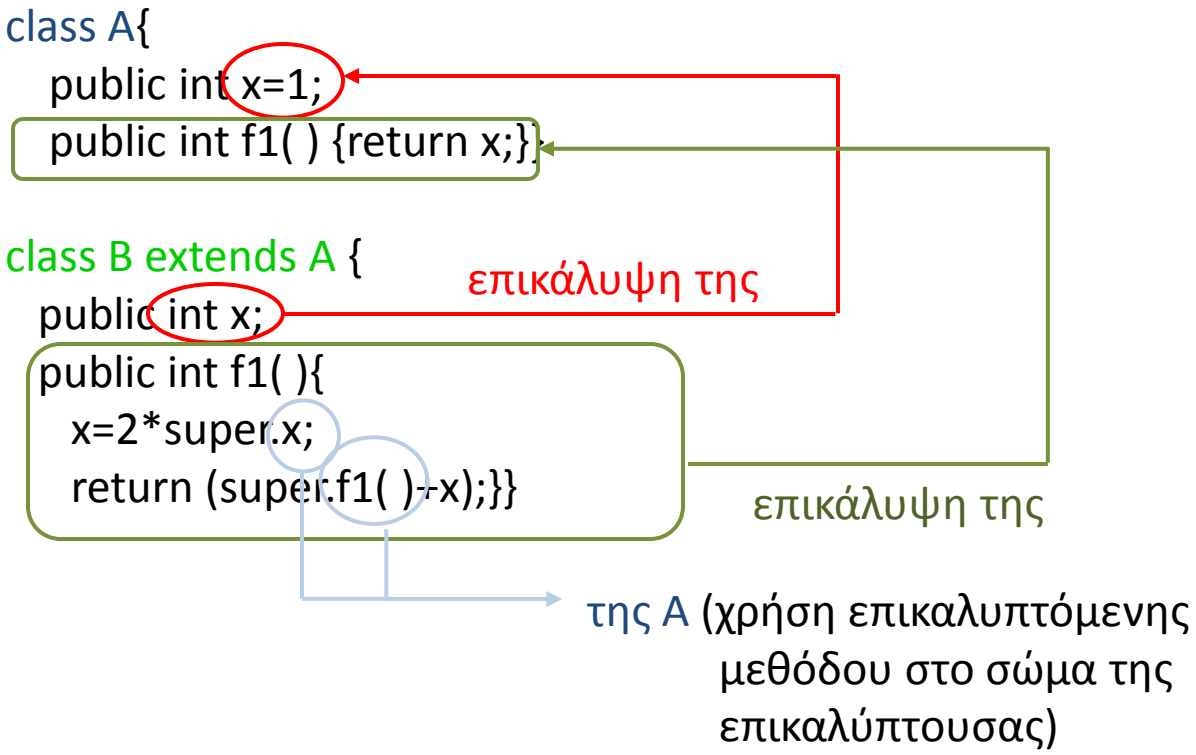
```
super.<όνομα-μεθόδου>(<παράμετροι>);
```

- Επίσης, μπορούμε να καλέσουμε απ' ευθείας μια μεταβλητή της υπερκλάσης:

```
super.<όνομα-μεταβλητής>;
```

- Η δυνατότητα αυτή μπορεί να χρησιμοποιηθεί για να καλέσουμε επικαλυπτόμενες μεθόδους ή μεταβλητές.

ΠΑΡΑΔΕΙΓΜΑ


class A{
public int x=1;
public int f1( ) {return x;}}
class B extends A {
επικάλυψη της
public int x;
public int f1( ){
x=2*super.x;
return (super.f1( )-x);}}
επικάλυψη της
της A (χρήση επικαλυπτόμενης
μεθόδου στο σώμα της
επικαλύπτουσας)

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 3: JAVA: ΕΞΑΙΡΕΣΕΙΣ, ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ, ΝΗΜΑΤΑ

Εξαιρέσεις

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΕΞΑΙΡΕΣΕΙΣ

ΕΞΑΙΡΕΣΕΙΣ

Εξαίρεση (Exception): Ένα συμβάν κατά τον χρόνο εκτέλεσης (π.χ. αδυναμία ανοίγματος αρχείου ή ανάγνωσης πληροφορίας, μη ύπαρξη αντικειμένου στη στοίβα) που απαιτεί ειδικό χειρισμό. Ονομάζεται και σφάλμα (error).

Χειρισμός εξαιρέσεων (Exception handling): Αναγνώριση και αντιμετώπιση του συμβάντος (σφάλματος).

Εξαιρέσεις προκαλούνται → `από το σύστημα`

Εξαιρέσεις προκαλούνται → `από το πρόγραμμα`

ΓΙΑΤΙ ΕΙΔΙΚΟΣ ΧΕΙΡΙΣΜΟΣ ΕΞΑΙΡΕΣΕΩΝ;

Η κλασσική διαχείριση λαθών περιπλέκει τον κώδικα, ώστε να χάνεται η «διαύγειά» του, διότι δεν υπάρχει διάκριση μεταξύ του κώδικα που αφορά στην κυρίως (ή «κανονική») λειτουργία και αυτού που αφορά στον χειρισμό των λαθών-εξαιρέσεων.

Ο ξεχωριστός μηχανισμός χειρισμού εξαιρέσεων επιτρέπει τη συγγραφή καθαρού, εύρωστου και ανεκτικού σε λάθη κώδικα. Διατηρεί μια ισορροπία μεταξύ αξιοπιστίας και διαύγειας.

Γλώσσες που διαθέτουν τέτοιο μηχανισμό: PL/1, Ada, Java

ΜΗΧΑΝΙΣΜΟΣ ΕΞΑΙΡΕΣΕΩΝ ΣΤΗ JAVA

Στόχοι

- Έγερση (ή πρόκληση) εξαιρέσεων
- Σύλληψη και διευθέτηση εξαιρέσεων

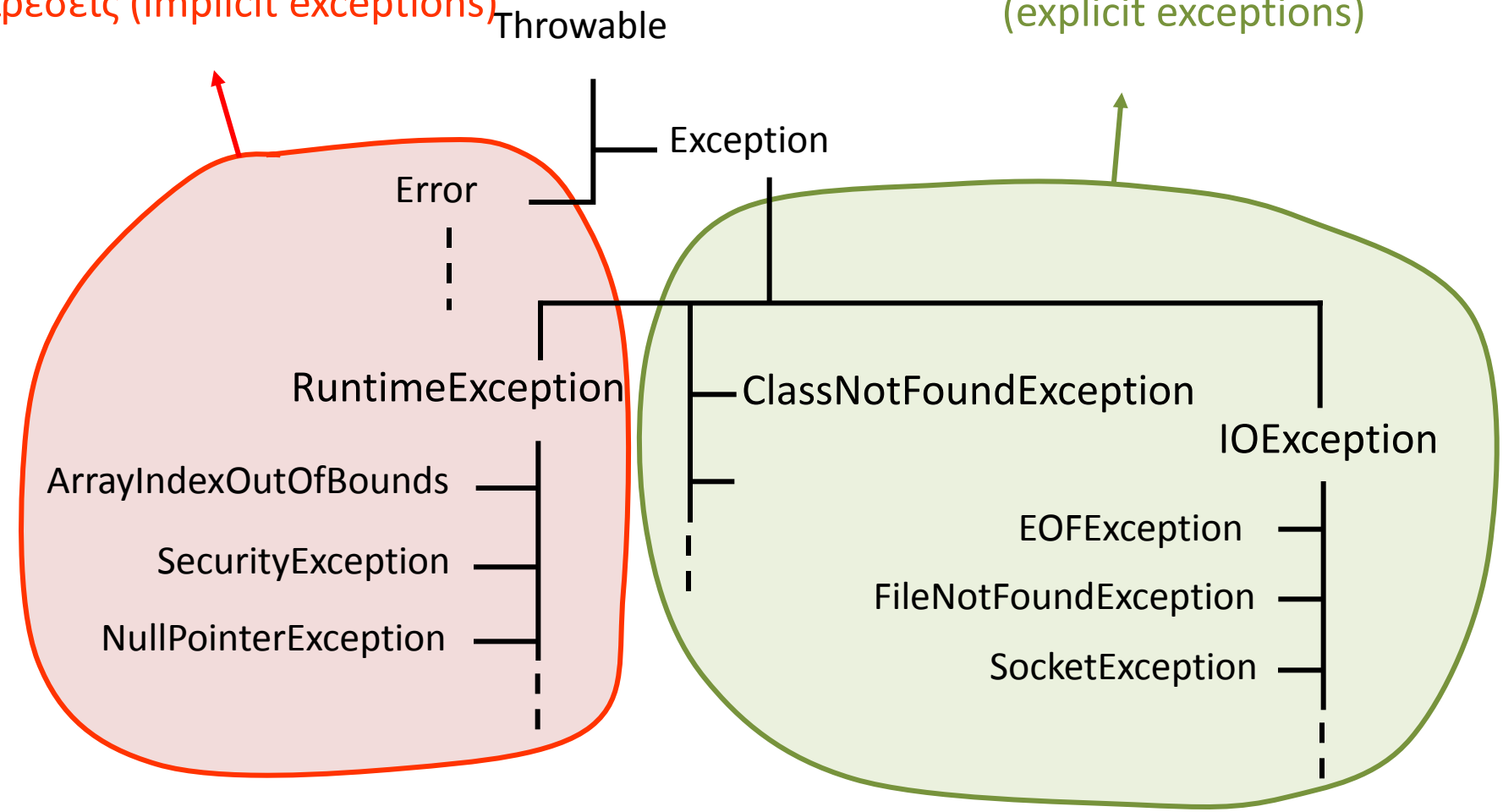
Χαρακτηριστικά

- Οι εξαιρέσεις στη Java είναι αντικείμενα (στιγμιότυπα)
- Υπάρχει ιδιαίτερη ιεραρχία κλάσεων εξαιρέσεων

ΙΕΡΑΡΧΙΑ ΕΞΑΙΡΕΣΕΩΝ

Αφανείς ή εσωτερικές εξαιρέσεις (implicit exceptions)

Εμφανείς (ή σαφείς) εξαιρέσεις (explicit exceptions)



Throwable, Exception, RuntimeException → **Java.lang**
IOException → **Java.io**

ΔΗΛΩΣΗ ΕΞΑΙΡΕΣΕΩΝ (1)

Σύνταξη

```
<προσδ.> <τύπος> <όνομα-μεθ.> (<παραμ.>) throws
<όν. εξαιρ.-1> [, <όν. εξαίρ.-2>, ...]
```

(εμφανείς εξαιρέσεις)

π.χ.

```
public ComplexNumber divide (double d) throws IOException
```

Δηλώνουμε δηλ. ότι η μέθοδος divide έχει κάποιο κώδικα στο σώμα της, που μπορεί να προκαλέσει εξαίρεση του τύπου IOException, αν κάτι πάει στραβά, π.χ. αν ζητηθεί από τη μέθοδο να υπολογίσει κάτι που δεν επιτρέπεται (ή δεν προβλέπεται) από τον τρόπο υλοποίησής της, κάτι δηλ. που προκαλεί σφάλμα/λάθος.

ΔΗΛΩΣΗ ΕΞΑΙΡΕΣΕΩΝ (2)

Αυτό σημαίνει ότι:

(α) η μέθοδος προκαλεί η ίδια, άμεσα, εξαίρεση (χρησιμοποιώντας την εντολή **throw** στο σώμα της).

(άμεση πρόκληση/έγερση)

ή

(β) χρησιμοποιεί στο σώμα της μέθοδο που προκαλεί εξαίρεση, την οποία όμως δεν χειρίζεται η ίδια, αλλά περνά τον χειρισμό της στην καλούσα μέθοδο.

(έμμεση πρόκληση/έγερση)

(Οι κλάσεις εξαιρέσεων συνήθως έχουν δύο δημιουργούς, έναν χωρίς ορίσματα και έναν με ένα όρισμα τύπου String, που αντιπροσωπεύει το μήνυμα στον χρήστη).

ΑΜΕΣΗ ΠΡΟΚΛΗΣΗ/ΕΓΕΡΣΗ ΕΞΑΙΡΕΣΗΣ

Δημιουργία νέου στιγμιοτύπου εξαίρεσης
(χρήση εντολής `throw`)

Π.χ.

```
Exception dwz = new Exception ("Διαίρεση με το μηδέν");
throw dwz;
```

ή

```
throw new Exception ("Διαίρεση με το μηδέν");
```

στο σώμα της μεθόδου

Π.χ.

```
public ComplexNumber divide (double d) throws Exception{
   if (d == 0)
     throw new Exception ("Διαίρεση με το μηδέν");
   return new ComplexNumber (mdReal/d, mdImag/d);
}
```

ΕΜΜΕΣΗ ΠΡΟΚΛΗΣΗ/ΕΓΕΡΣΗ– ΣΥΛΛΗΨΗ ΕΞΑΙΡΕΣΗΣ

Χρήση στο σώμα μιας μεθόδου Α άλλης μεθόδου Β, η οποία προκαλεί/εγείρει εξαίρεση

Π.χ.

```
public void readFile (String filename) throws IOException{
  while (numBytes <= myBuffer.length){
         myInputStream.read (myBuffer);
         myBytes++;
  }
}
```

Προκαλεί εξαίρεση τύπου

Δηλ. η readFile προκαλεί **<u>έμμεσα</u>** εξαίρεση.

Ο χειρισμός της εξαίρεσης περνά στη μέθοδο που καλεί την readFile (έμμεσος χειρισμός/σύλληψη)

ΔΙΑΧΕΙΡΙΣΗ ΕΞΑΙΡΕΣΕΩΝ

Πολλές μέθοδοι (της βιβλιοθήκης) της Java προκαλούν εξαιρέσεις (δηλ. περιέχουν την εντολή *throw* στο σώμα τους).

Αυτές τις εξαιρέσεις, οποιαδήποτε μέθοδος τις χρησιμοποιεί επιβάλλεται (από τον μεταγλωττιστή) να τις χειριστεί (δηλ. να προστατεύσει τον κώδικά της από αυτές τις εξαιρέσεις).

Π.χ. η " `System.in.read( );` " , αν δεν προστατευτεί, θα προκαλέσει το παρακάτω λάθος μεταγλώττισης:

```
"Test.java:23: unreported exception java.io.IOException
  must be caught or declared to be thrown
  System.in.read( );"
                ^
```

ΑΜΕΣΗ ΣΥΛΛΗΨΗ (ΔΙΑΧΕΙΡΙΣΗ) ΕΞΑΙΡΕΣΕΩΝ

Χρήση μπλοκ try-catch

Σύνταξη:

```
try {<κώδικας που μπορεί να εγείρει εξαίρεση>}
                    .....
catch (<κλάση εξαίρ.> <μεταβλ. εξαίρ.>)
      {<κώδικας χειρισμού>}
```

Ερμηνεία: Δοκίμασε (try) αυτό το κομμάτι κώδικα, που μπορεί να προκαλέσει εξαίρεση. Αν εκτελεστεί σωστά, προχώρησε με το υπόλοιπο πρόγραμμα. Αν όχι, τότε κάνε σύλληψη (catch) της εξαίρεσης και χειρίσου την.

Μπορεί να υπάρχουν περισσότερα του ενός catch μπλοκ για το ίδιο try μπλοκ.

ΔΙΑΔΙΚΑΣΙΑ ΑΜΕΣΗΣ ΣΥΛΛΗΨΗΣ-ΔΙΑΧΕΙΡΙΣΗΣ

1. Όταν εγερθεί εξαίρεση, από τον κώδικα ενός try μπλοκ, ο έλεγχος μεταφέρεται έξω από το μπλοκ και γίνεται αναζήτηση του κατάλληλου catch μπλοκ
2. Αν βρεθεί, ο έλεγχος μεταφέρεται στο catch μπλοκ και γίνεται ο χειρισμός της εξαίρεσης
3. Αν δεν εγερθεί εξαίρεση, τα catch μπλοκ παραλείπονται
4. Μετά τον χειρισμό, ο έλεγχος δεν επιστρέφει στο σημείο που ηγέρθη η εξαίρεση, αλλά συνεχίζει με τον κώδικα που υπάρχει μετά τα try και catch μπλοκ

ΠΑΡΑΔΕΙΓΜΑ (Χωρίς σύλληψη)

```
import java.io.*;
public class ExceptionTest {
 public static void main (String args[ ]) {
    int num;
    num= getNumber();
    System.out.println ("Το διπλάσιο: " + 2*num);
 }
 public static int getNumber () {
       String line;
       BufferedReader br= new BufferedReader (
                        new InputStreamReader (System.in));
       System.out.println ("Δώσε αριθμό: ");
       line = br.readLine();
       return Integer.parseInt(line);
 }
}
```

```
Exception java.io.IOException must be caught or it
must be declared in the throws clause of this method.
line = br.readLine ();
           ^
```

ΠΑΡΑΔΕΙΓΜΑ (Άμεση σύλληψη)

```
import java.io.*;
public class ExceptionTest {
  public static void main (String args[ ]) {
       int num;
       num= getNumber();
       System.out.println ("Το διπλάσιο: " + 2*num);
  }
  public static int getNumber () {
       String line;
       BufferedReader br= new BufferedReader(
                          new InputStreamReader (System.in));
       System.out.println ("Δώσε αριθμό: ");
       try {
            line = br.readLine();
            return Integer.parseInt(line);
       }
       catch (IOException e) {
           System.out.println (e);
       }
       catch (NumberFormatException e) {
           System.out.println (e);
       }
   }
}
```

ΠΑΡΑΔΕΙΓΜΑ (Έμμεση σύλληψη ή μετάδοση σύλληψης)

```
public class ExceptionTest {
 public static void main (String args[ ]) {
  try{int num;
       num= getNumber();
       System.out.println ("Το διπλάσιο: " + 2*num); }
  catch (IOException e) {System.out.println (e);}
  catch (NumberFormatException e) {System.out.println (e);}
 }

 public static int getNumber () throws Exception {
    String line;
    BufferedReader br= new BufferedReader
                        (new InputStreamReader (System.in));
    System.out.println ("Δώσε αριθμό: ");
    line = br.readLine();
    return Integer.parseInt(line);
  }
}
```

Σύλληψη εξαίρεσης

Καλούσα μέθοδος

Μέθοδος που περιέχει μέθοδο που προκαλεί εξαίρεση

FINALLY ΜΠΛΟΚ

Για την εκτέλεση ενεργειών που πρέπει να εκτελεστούν οπωσδήποτε, ανεξάρτητα από τον αν προκληθεί ή όχι εξαίρεση (π.χ. απόδοση πόρων του συστήματος), χρησιμοποιείται το **finally** μπλοκ.

Π.χ.

```
try {readFile ();}
        …..
catch (IOException e) { … }
finally {close Textfile4;}
```

ΔΗΜΙΟΥΡΓΙΑ ΝΕΩΝ ΤΥΠΩΝ ΕΞΑΙΡΕΣΕΩΝ (ΑΠΟ ΤΟΝ ΧΡΗΣΤΗ)

- Πρέπει να είναι υποκλάσεις της Exception ή κάποιας υποκλάσης της
- Πρέπει να έχει δύο δημιουργούς (ένα χωρίς όρισμα, ένα με όρισμα String)

Π.χ.

```
public class xxException extends Exception {
    public xxException () { }
    public xxException (String msg) {
       super(msg);
    }
}
```

ΠΑΡΑΔΕΙΓΜΑ

```
public class DivideByZeroException
                              extends ArithmeticException {
    public DivideByZeroException (){
        super ("Attempted to divide by zero");
    }
}
```

Μέθοδος modulo

```
double modulo (double num, double den)
                        throws DivideByZeroException {
    if (den == 0)
         throw new DivideByZeroException ();
    return (double) (num/den);
}
```

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 3: JAVA: ΕΞΑΙΡΕΣΕΙΣ, ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ, ΝΗΜΑΤΑ

Είσοδος - Έξοδος

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ ΔΕΔΟΜΕΝΩΝ

ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ ΔΕΔΟΜΕΝΩΝ

Στην java οι πληροφορίες αποθηκεύονται και ανακαλούνται/ ανασύρονται με τη χρήση ενός συστήματος επικοινωνίας που χρησιμοποιεί την έννοια του **stream** (κανάλι επικοινωνίας).

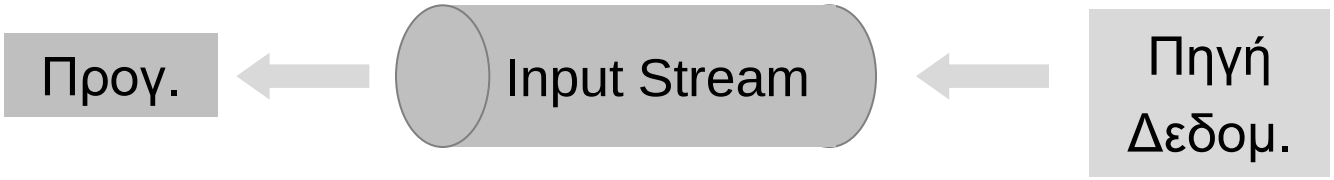
Σαν stream ορίζεται μια «διαδρομή» μέσα από την οποία μεταφέρονται δεδομένα από μια θέση σε μια άλλη. Κάθε stream είναι ένα αντικείμενο (στιγμιότυπο) java.

Η πληροφορία που μεταφέρεται μέσα από ένα stream είναι μια ακολουθία δεδομένων που έχουν μια πηγή ή ένα αποδέκτη.

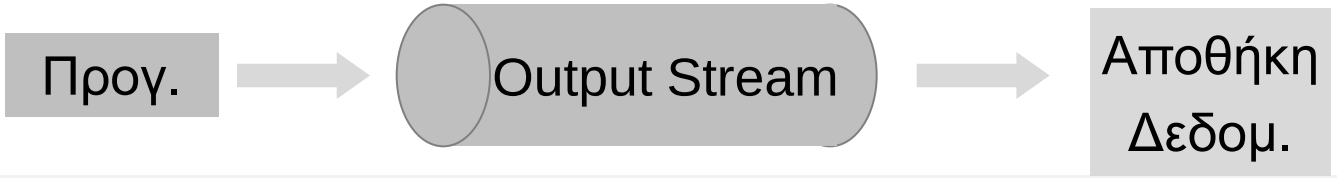
ΤΥΠΟΙ STREAM (1)

Με βάση την κατεύθυνση

- input streams (για ανάγνωση πληροφορίας, δηλ. αποστολή δεδομένων από μια εξωτερική πηγή στο πρόγραμμα)



- output streams (για αποθήκευση πληροφορίας, δηλ. αποστολή δεδομένων από το πρόγραμμα σ' ενα εξωτερικό προορισμό)



ΤΥΠΟΙ STREAM (2)

Με βάση τον τύπο της πληροφορίας

- byte streams(για μεταφορά δεδομένων πρωτογενών τύπων ή αλφαριθμητικών της java)

- character streams (για μεταφορά χαρα κτήρων/κειμένου από αρχεία που τα παριστάνουν μέσω κωδικοποιήσεων ASCII ή Unicode)

ΚΛΑΣΕΙΣ STREAM (I/O)

Περιέχονται στο πακέτο `java.io`

Για byte streams

`InputStream, OutputStream (abstract)`

`FileInputStream, FileOutputStream`

`BufferedInputStream, BufferedOutputStream`

`DataInputStream, DataOutputStream`

Για character streams

`Reader, Writer (abstract)`

`FileReader, FileWriter`

`BufferedReader, BufferedWriter`

ΚΛΑΣΕΙΣ STREAM (I/O)

```
Object
 InputStream
   FileInputStream
   FilterInputStream
     BufferedInputStream
     DataInputStream
 OutputStream
   FileOutputStream
   FilterOutputStream
     BufferedOutputStream
     DataOutputStream
 RandomAccessFile
 File
```

ΚΥΡΙΑ ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ(1)

Με την εκτέλεση κάθε προγράμματος (εφαρμογής) δημιουργούνται τρία (3) (στιγμιότυπα) streams:

- **`System.in`** (κύρια είσοδος)
- **`System.out`** (κύρια έξοδος)
- `System.err` (κύριο σφάλμα)

ΚΥΡΙΑ ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ(2)

- **Βασική έξοδος:**

```
System.out.println(String s);        (έξοδος string)
System.out.write(int x);             (έξοδος ενός byte)
System.out.write(byte b[]);          (έξοδος ενός αριθμού bytes από πίνακα)
```

(Μέθοδοι της OutputStream)

- **Βασική είσοδος:** (Επιστρέφει έναν απρόσημο ακέραιο 0-255 ή -1)

```
System.in.read();                    (είσοδος ενός byte)
System.in.read(byte b[]);            (είσοδος ενός αριθμού bytes σε πίνακα)
```

(Μέθοδοι της InputStream)

Όλες, πλην της println(…):
"throws IOException"

ΠΑΡΑΔΕΙΓΜΑ (1)

```
import java.io.*;
public class Inpdata{
    public static void main(){
      char b='0';
      try {
        System.out.println("Give a character: ");
        if ((b = (char)System.in.read()) != '\n')
            System.out.println(b);
      }
      catch(IOException e){
        System.out.println(e);
      }
    }
}
```

ΠΑΡΑΔΕΙΓΜΑ (2)

```
import java.io.*;
public class MInpdata1{
 public static void main(){
   char b='0';
     try {
            System.out.println("Give a sequence of characters: ");
      for (int i=0; (b=(char)System.in.read()) != '\n'; i++)
        System.out.println(b);
     }
     catch(IOException e){
       System.out.println(e);
     }
 }
}
```

ΠΑΡΑΔΕΙΓΜΑ (3)

```
import java.io.*;
public class MInpdata2{
    public static void main()  {
      byte [] b = new byte[10];
      try {
        System.out.println("Write a string (max length=10): ");
        System.in.read(b);
        String s = new String(b);
        System.out.println(s);
      }
      catch(IOException e) {
        System.out.println(e);
      }
    }
}
```

ΠΑΡΑΔΕΙΓΜΑ (4)

```
import java.io.*;
public class MInpdata3{
    public static void main() {
      byte [] b = new byte[10];
      try {
        System.out.println("Write a string (max length=10): ");
        System.in.read(b);
        System.out.write(b);
      }
      catch(IOException e){
        System.out.println(e);
      }
    }
}
```

ΔΙΑΔΙΚΑΣΙΑ ΕΙΣΟΔΟΥ

1. Άνοιγμα/Δημιουργία ενός stream (π.χ. προκειμένου για αρχείο, δημιουργούμε ένα στιγμιότυπο της FileInputStream και το συσχετίζουμε με το αρχείο εισόδου)
2. Ανάγνωση δεδομένων, ενόσω υπάρχουν, από την πηγή εισόδου (π.χ. με τη βοήθεια μεθόδων της FileInputStream, όπως η read())
3. Κλείσιμο του stream (π.χ. χρήση της μεθόδου close())

ΔΙΑΔΙΚΑΣΙΑ ΕΞΟΔΟΥ

1. Άνοιγμα/Δημιουργία ενός stream (π.χ. προκειμένου για αρχείο, δημιουργούμε ένα στιγμιότυπο της FileOutputStream και το συσχετίζουμε με το αρχείο εξόδου)
2. Αποστολή δεδομένων, ενόσω υπάρχουν, στον προορισμό εξόδου (π.χ. με τη βοήθεια μεθόδων της FileOutputStream, όπως η write())
3. Κλείσιμο του stream (π.χ. χρήση της μεθόδου close())

ΦΙΛΤΡΑΡΙΣΜΑ STREAM

Ορισμός

Φίλτρο είναι ένας τύπος stream που τροποποιεί τον τρόπο χειρισμού ενός άλλου stream.

Διαδικασία

1. Δημιουργία ενός stream (εισόδου ή εξόδου).
2. Συσχέτιση φίλτρου με το stream.
3. Ανάγνωση/αποστολή δεδομένων από/προς το φίλτρο (αντί του stream).

BYTE STREAMS ΑΡΧΕΙΩΝ (1)

Είναι στιγμιότυπα των FileInputStream και FileOutputStream, που είναι υποκλάσεις των InputStream και OutputStream αντίστοιχα.

(Για να μπορούμε να διαβάσουμε δεδομένα από ή να γράψουμε δεδομένα σε ένα αρχείο πρέπει να το συσχετίσουμε μ' ένα stream εισόδου ή εξόδου αντίστοιχα.)

Εισόδου

Δημιουργία: **`FileInputStream(String fname)`**

Ανάγνωση: **`read()`** (επόμενο byte: ακέραιος ή –1)

`read(byte[], int, int)` (αριθ. Bytes ή –1)

θέση 1ου byte

αριθμός bytes

ΠΑΡΑΔΕΙΓΜΑ

```
import java.io.*;
public class ReadFile{
   public static void main(){
   try {FileInputStream fis1 = new FileInputStream("test.dat");
         boolean eof = false;
         int count = 0;
         while (!eof) {
           int inp = fis1.read();
           System.out.print(inp + " ");
           if (inp = = -1)
              eof = true;
           else count++;}
         fis1.close();
         System.out.println("\nBytes read: " + count);}
      catch(IOException e)
       {System.out.println(e);}
   }
}
```

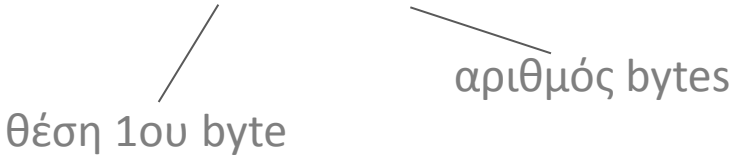
BYTE STREAMS ΑΡΧΕΙΩΝ (2)

Εξόδου

Δημιουργία: **`FileOutputStream(String fname)`**

Εγγραφή: **`write(int)`** (εγγραφή byte)

`write(byte[], int, int)` (εγγραφή πολλών bytes)



ΠΑΡΑΔΕΙΓΜΑ

```
import java.io.*;
public class WriteFile{
   public static void main(){
     int [] data = {71, 73, 65, 0, 56, 33, 18, 22, 0, 0, 250, 178, 4, 2, 0, 0, 65, 0}
     try {FileOutputStream fos1 = new FileOutputStream("testo.dat");
         for (int i=0; i < data.length; i++)
             fos1.write(data[i]);
         fos1.close();
     }
     catch(IOException e) {
        System.out.println(e);
     }
   }
}
```

BUFFERED (ΕΝΤΑΜΙΕΥΜΕΝΑ) STREAMS

Για αποδοτικότερη χρήση, χρησιμοποιούν ένα buffer (ενταμιευτή) για ενδιάμεση αποθήκευση δεδομένων.

- **Εξόδου**

  Δημιουργία: `BufferedOutputStream(OutputStream)`
  `BufferedOutputStream(OutputStream, int)`

  Εγγραφή: `write(int),write(byte[], int, int)`

  Άδειασμα buffer: `flush()`

- **Εισόδου**

  Δημιουργία: `BufferedInputStream(InputStream)`
  `BufferedInputStream(InputStream, int)`

  Ανάγνωση: `read(), read (byte[], int, int)`

ΠΑΡΑΔΕΙΓΜΑ

```
import java.io.*;
public class CopyFile{
  public static void main(String args []){
   try {FileInputStream fis = new FileInputStream(args[0]);
        BufferedInputStream bis = new BufferedInputStream(fis);
        FileOutputStream fos = new FileOutputStream(args[1]);
        BufferedOutputStream bos = new BufferedOutputStream(fos);
        int nbytes;
        while (bis.available() > 0) {
           nbytes = bis.read();
           bos.write(nbytes); }
        bis.close(); bos.flush(); bos.close();
      }
      catch(IOException e)
       {System.out.println(e);}
   }
}
```

BYTE STREAMS ΔΕΔΟΜΕΝΩΝ

Για να εργαστούμε με δεδομένα που δεν παριστάνονται σαν ακολουθίες bytes, αλλά σαν λογικά τμήματα.

- **Εξόδου**

  Δημιουργία: `DataOutputStream(OutputStream)`

  Εγγραφή: `writeDouble(double),writeFloat(float) writeInt(int),writeLong(long),writeShort(int), writeByte (int), writeBoolean(boolean)`

- **Εισόδου**

  Δημιουργία: `DataInputStream(InputStream)`

  Ανάγνωση: `readDouble(),readFloat(),readInt(),readLong(), readShort(), readByte (), readBoolean()`

ΠΑΡΑΔΕΙΓΜΑ

```
import java.io.*;
public class WriteFile{
   public static void main(){
   double [] data = {71.2, 73.1, 65.0, 0.0, 56.4, 33.4, 18.7, 22.0, 0, 0,
                                                  250.0, 178.3, 4.2, 2.1, 0, 0, 6.35, 0};
   try {FileOutputStream fos1 = new FileOutputStream("testr.dat");
         DataOutputStream fods1 = new DataOutputStream(fos1);
         for (int i=0; i < data.length; i++)
              fods1.writeDouble(data[i]);
         fods1.close();
    }
    catch(IOException e)
      {System.out.println(e);}
   }
}
```

ΠΑΡΑΔΕΙΓΜΑ

```
import java.io.*;
public class ReadFile{
  public static void main(){
    try { FileInputStream fis1 = new FileInputStream("testr.dat");
          DataInputStream fids1 = new DataInputStream(fis1);
          int count = 0;
           try {   while (true) {
                      double inp = fids1.readDouble();
                      System.out.print(inp + " \n");
                    count++;}  }
            catch(EOFException eof)
                 {fids1.close();}
          System.out.println("\nNums read: " + count);}
     catch(IOException e)
        {System.out.println("Error-- " + e.toString());}
  }
}
```

ΠΑΡΑΔΕΙΓΜΑ

```
import java.io.*;
public class WritePrimeNums{
  public static void main(String args []){
      int num = 2;
      try { FileOutputStream fis = new FileOutputStream("primes.dat");
          BufferdOutputStream bos = new BufferedInputStream(fis);
          DataOutputStream dos = new DataOutputStream(bos);
          while (num < 400) {
            if (isPrime(num))
              dos.writInt(num);
            num++;
          }
          dos.close();}
      catch(IOException e)
          {System.out.println(e);}
   }
}
```

ΠΑΡΑΔΕΙΓΜΑ

```
import java.io.*;
public class ReadPrimeNums{
  public static void main(String args []){
    try {DataInputStream dos = new DataInputStream(
                                new BufferedInputStream(
                                new FileOutputStream("primes.dat")));
      try {while (true) {
            int num = dos.readInt();
            System.out.println(num + " ");}
      }
      catch (EOFException eof) {}
      finally {dos.close()};
    }
    catch(IOException e)
      {System.out.println("Error: " + e.toString() );}
  }
}
```

CHARACTER STREAMS ΑΡΧΕΙΩΝ (1)

Είναι στιγμιότυπα των FileReader και FileWriter, που είναι υποκλάσεις των InputStreamReader και OutputStreamWriter αντίστοιχα.

- **Εισόδου**

  Δημιουργία: `FileReader(String fname)`

  Ανάγνωση: `read()`

  `read(char[], int, int)`

  αριθμός χαρακτήρων

  θέση 1ου χαρακτήρα

CHARACTER STREAMS ΑΡΧΕΙΩΝ (2)

Για ανάγνωση ολόκληρης γραμμής, αντί χαρακτήρα.

Δημιουργία: `BufferedReader(Reader)`

`BufferedReader(Reader, int)`

Ανάγνωση: `read(), read (byte[], int, int)`

`readLine()` (επιστρέφει ένα String, που είναι μια γραμμή κειμένου)

ΔΙΑΧΕΙΡΙΣΗ ΑΡΧΕΙΩΝ ΑΜΕΣΗΣ ΠΡΟΣΠΕΛΑΣΗΣ (1)

Μέχρι τώρα αναφερόμαστε σε αρχεία σειριακής προσπέλασης (για να βρούμε μια πληροφορία πρέπει να περάσουμε από όλες τις προηγούμενες)

Μπορούμε όμως να δημιουργήσουμε και αρχεία τυχαίας προσπέλασης (random access), όπου μπορούμε να μεταβαίνουμε σε ένα τμήμα πληροφορίας χωρίς να είναι απαραίτητο να περάσουμε απ' όλα τα προηγούμενα.

ΔΙΑΧΕΙΡΙΣΗ ΑΡΧΕΙΩΝ ΑΜΕΣΗΣ ΠΡΟΣΠΕΛΑΣΗΣ (2)

Η δημιουργία και διαχείριση αρχείων τυχαίας προσπέλασης γίνεται μέσω της κλάσης RandomAccessFile.

Δημιουργοί: (και οι δύο throws FileNotFoundException)

```
RandomAccessFile (String name, String mode)
RandomAccessFile (File name, String mode)
```

αντικείμενο αρχείου (βλ. αργότερα)

Κανονίζει τον τρόπο ανοίγματος, π.χ. “r” (για ανάγνωση μόνο), “rw” (για ανάγνωση και εγγραφή)

ΔΙΑΧΕΙΡΙΣΗ ΑΡΧΕΙΩΝ ΑΜΕΣΗΣ ΠΡΟΣΠΕΛΑΣΗΣ (3)

Οι μέθοδοι ανάγνωσης και εγγραφής είναι οι ίδιες με αυτές των byte streams αρχείων και δεδομένων.

Ένα χαρακτηριστικό των αρχείων τυχαίας προσπέλασης είναι ότι τα δεδομένα τους διαβάζονται και γράφονται αρχίζοντας όχι απαραίτητα από την αρχή, αλλά από κάποια θέση του δείκτη που καθορίζουμε. Αυτό γίνεται με τη μέθοδο (public void):

`seek (long index)` **`(η οποία throws IOException)`**

Μετακινεί τον δείκτη ακριβώς πριν από το (index+1) byte του αρχείου. Το πρώτο byte ενός αρχείου είναι στη θέση 0.

ΔΙΑΧΕΙΡΙΣΗ ΑΡΧΕΙΩΝ ΑΜΕΣΗΣ ΠΡΟΣΠΕΛΑΣΗΣ (4)

Οι εντολές/μέθοδοι ανάγνωσης και εγγραφής ενεργούν πάντα στην τρέχουσα θέση του δείκτη στο αρχείο. Η εκτέλεσή τους έχει σαν αποτέλεσμα τη μετακίνηση του δείκτη στην επόμενη θέση.

Η RandomAccessFile παρέχει και άλλες χρήσιμες μεθόδους, όπως:

```
public long getFilePointer()
```
Επιστρέφει την τιμή του δείκτη.

```
public long length()
```
Επιστρέφει το μήκος του αρχείου σε bytes.

```
public void setlength(long newlen)
```
Κάνει το μήκος του αρχείου ίσο με newlen.

(Όλες `throws IOException`)

ΑΝΤΙΚΕΙΜΕΝΑ ΑΡΧΕΙΩΝ(1)

Τα αντικείμενα αρχείων αντιπροσωπεύουν συγκεκριμένα αρχεία ή καταλόγους και χρησιμοποιούνται όχι για ανάγνωση ή εγγραφή, αλλά για τη διαχείριση πληροφοριών σχετικών με τα «φυσικά» χαρακτηριστικά αρχείων ή καταλόγων.

Δημιουργούνται ως στιγμιότυπα της κλάσης File (3 δημιουργοί):

```
File (String name)

File (String path, String name)

File (File directory, String name)
```

ΑΝΤΙΚΕΙΜΕΝΑ ΑΡΧΕΙΩΝ(2)

Παράδειγμα

File myDir = new File("c:\\ihatz\\java"); (1ος δημ.)
File myFile = new File(myDir, "test.dat"); (3ος δημ.)
Οι δύο αυτές προτάσεις μπορούν να συνοψιστούν:
File myFile=new File("c:\\ihatz\\java", "test.dat") (2ος δημ.)

Η File παρέχει πολλές μεθόδους:

```
exists(),    isFile(),    isDirectory(),    isAbsolute(),
canRead(),   canWrite(),  delete()  (όλες τύπου boolean)

getName(), getParent(), getPath()   (όλες τύπου String)
```

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.1

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 3: JAVA: ΕΞΑΙΡΕΣΕΙΣ, ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ, ΝΗΜΑΤΑ

Νήματα

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΝΗΜΑΤΑ ΣΤΗ JAVA

ΝΗΜΑΤΑ ΣΤΗ JAVA (1)

Ορισμός

Νήμα (thread) είναι μια ακολουθιακή ροή ελέγχου (δηλ. κάτι που έχει αρχή, ακολουθία εντολών και τέλος) σ' ένα πρόγραμμα.

Αιτία

Η δυνατότητα απομόνωσης (ή αυτονόμησης) κάποιων εργασιών, ώστε να μπορούν να εκτελούνται ταυτόχρονα.

Παρατηρήσεις

- Ένα νήμα δεν είναι πρόγραμμα αφ' εαυτού του.
- Ένα νήμα εκτελείται μέσα σ' ένα πρόγραμμα.
- Ένα μόνο νήμα δεν προσφέρει κάτι το καινούργιο.
- Δύο ή περισσότερα νήματα μπορούν να εκτελούνται
- ταυτόχρονα πραγματοποιώντας διαφορετικές εργασίες.

ΝΗΜΑΤΑ ΣΤΗ JAVA (2)

Κάθε νήμα αντιπροσωπεύει μια διεργασία. Διεργασίες που εκτελούνται ταυτόχρονα λέγονται ταυτόχρονες διεργασίες (concurrent processes)

Τη δυνατότητα ταυτόχρονης εκτέλεσης διεργασιών την παρέχει το λειτουργικό σύστημα, όπως π.χ. τα MS Windows, Linux. Τα συστήματα αυτά επιτρέπουν την ταυτόχρονη εκτέλεση πολλών προγραμμάτων (multitasking). Π.χ. ενώ διορθώνουμε ένα κείμενο στον επεξεργαστή κειμένου, γίνεται μια εκτύπωση.

Παρόμοια, ένα μόνο πρόγραμμα μπορεί να εκτελεί ταυτόχρονα πολλές διεργασίες (νήματα). Τέτοια προγράμματα αποτελούν πολυνηματικές εφαρμογές (multithreaded applications).

Π.χ. ο φυλλομετρητής HotJava είναι μια τέτοια εφαρμογή.

ΝΗΜΑΤΑ ΣΤΗ JAVA (3)

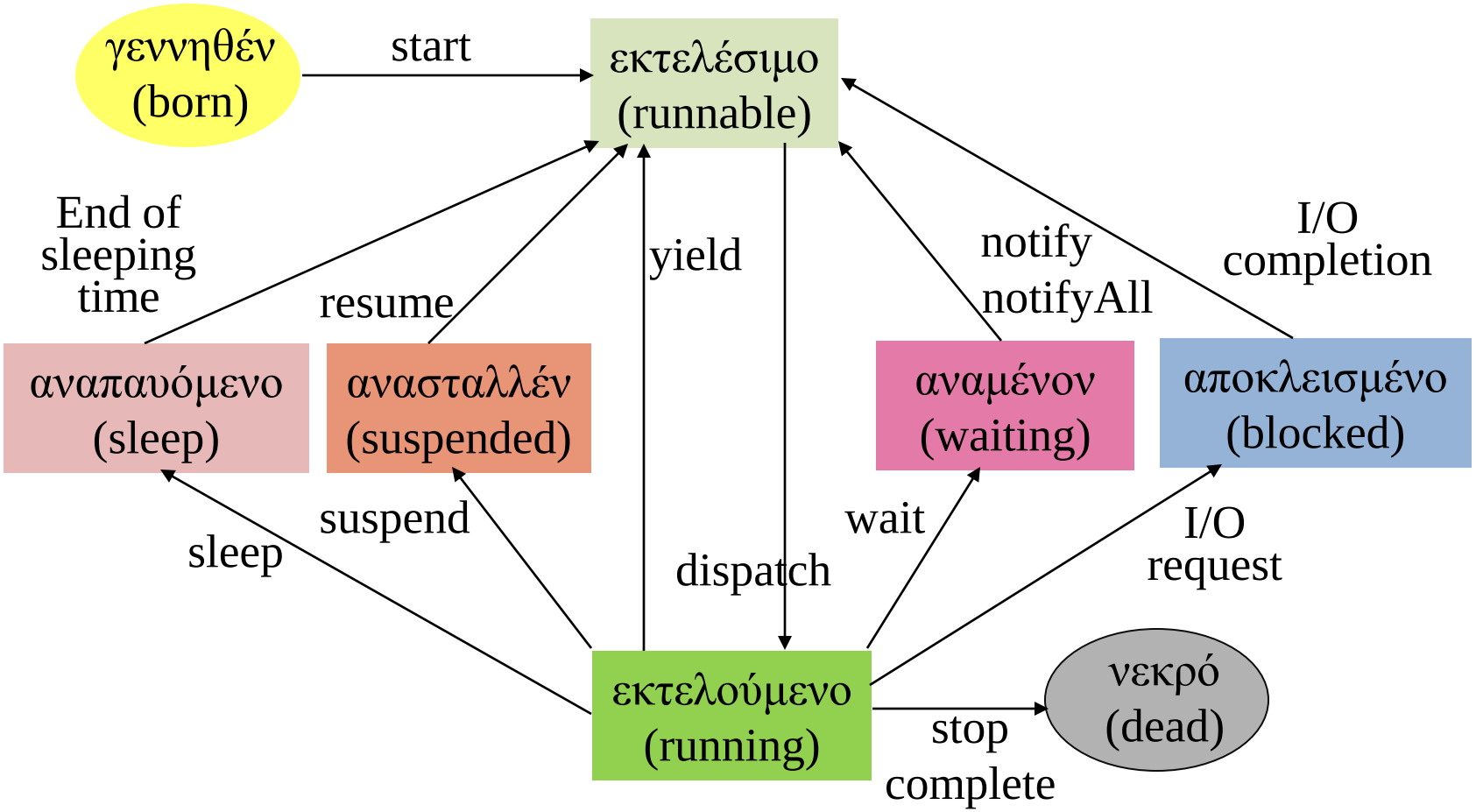
Παραδείγματα ταυτόχρονων διεργασιών:

- Κύλιση μιας σελίδας-κατέβασμα μιας εικόνας
- Εκτέλεση μιας προσομοίωσης-παίξιμο ενός ήχου
- Εκτύπωση μιας σελίδας-κατέβασμα μιας νέας σελίδας

Για να δημιουργήσουμε μια εφαρμογή που χρησιμοποιεί αυτή τη δυνατότητα πρέπει να χρησιμοποιήσουμε μια γλώσσα που να μπορεί να διαχειριστεί ταυτόχρονες διεργασίες (δηλ. νήματα). Μια τέτοια γλώσσα είναι η Java.

Το είδος προγραμματισμού που ασχολείται με τη διαχείριση ταυτόχρονων διεργασιών ονομάζεται ταυτόχρονος προγραμματισμός (concurrent programming).

ΚΥΚΛΟΣ ΖΩΗΣ ΝΗΜΑΤΟΣ



ΔΗΜΙΟΥΡΓΙΑ ΝΗΜΑΤΩΝ ΣΤΗ JAVA

Δύο τρόποι δημιουργίας νημάτων:

(α) Σαν στιγμιότυπα υποκλάσεων της κλάσης Thread

(β) Σαν στιγμιότυπα υλοποιήσεων της διεπαφής Runnable

(όταν η κλάση που δημιουργείται πρέπει να είναι υποκλάση κάποιας άλλης κλάσης, π.χ. της Applet)

Και στις δύο περιπτώσεις πρέπει να οριστεί το σώμα της μεθόδου void run(), που είναι κενό και προσδιορίζει τι κάνει το νήμα.

Η μέθοδος run() είναι (αφηρημένη) μέθοδος της Runnable, αλλά και της Thread (διότι η Thread υλοποιεί την Runnable).

ΚΛΑΣΗ THREAD-ΜΕΘΟΔΟΙ

Άλλες μέθοδοι της Thread είναι:

`start()` :Απαιτείται η κλήση της για να ενεργοποιηθεί το νήμα.

`sleep(n)` :Καθορίζει το χρόνο n (σε msec) που (θέλουμε να) καθυστερεί το νήμα πριν αρχίσει η εκτέλεσή του (το n είναι τύπου long).
Εγείρει εξαίρεση τύπου InterruptedException.

`interrupted()` :Επιστρέφει true, αν το νήμα είναι σε διακοπή, αλλιώς false.

`interrupt()` :Διακόπτει την εκτέλεση του νήματος.

`suspend()` :Αναστέλλει την εκτέλεση του νήματος.

`resume()` :Επαναρχίζει η εκτέλεση του νήματος.

`stop()` :Τερματίζει την εκτέλεση του νήματος.

`isAlive()` :Επιστρέφει true, αν έχει ενεργοποιηθεί το νήμα (έχει κληθεί η start και δεν έχει τερματιστεί), αλλιώς false.

ΚΛΑΣΗ THREAD-ΔΗΜΙΟΥΡΓΟΙ

Η Thread διαθέτει αρκετούς δημιουργούς. Οι πιο δημοφιλείς είναι:

Thread ()

Thread (String name)

Thread (Runnable object)

Thread (Runnable object, String name)

όπου name είναι το όνομα του νήματος και object ένα αντικείμενο τύπου Runnable.

ΝΗΜΑΤΑ ΜΕ ΧΡΗΣΗ ΤΗΣ THREAD (1)

Διαδικασία δημιουργίας νήματος

- Δημιουργία μιας υποκλάσης της Thread
- Υλοποίηση της run στο σώμα της υποκλάσης.
- Δημιουργία στιγμιοτύπου της υποκλάσης
- Κλήση της start από το στιγμιότυπο

Λειτουργία μεθόδου start

- Δημιουργεί το νέο νήμα εκτέλεσης
- Καλεί τη μέθοδο run
- Επιστρέφει τον έλεγχο στο σημείο κλήσης της

ΝΗΜΑΤΑ ΜΕ ΧΡΗΣΗ ΤΗΣ THREAD (2)

(δημιουργός)

(υλοποίηση της run)

```
public class SimpleThread extends Thread {
    public SimpleThread(String str) {
        super(str);
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(i + " " + getName());
            try {
                Thread.sleep((int) (Math.random() * 1000));
                } catch (InterruptedException e) {}
            }
            System.out.println("DONE! " + getName()); }
}
```

ΝΗΜΑΤΑ ΜΕ ΧΡΗΣΗ ΤΗΣ THREAD (3)

```
public class TwoThreadsDemo {
        public static void main (String[] args) {
                thread1 = new SimpleThread("Patra");
                thread2 = new SimpleThread("Athina");
                thread1. start();
                thread2. start();
        }
}
```

(δημιουργία νημάτων ως στιγμιότυπα της SimpleThread και κλήση της start)

ΝΗΜΑΤΑ ΜΕ ΧΡΗΣΗ ΤΗΣ THREAD (4)

```
public class TwoThreadsDemo {
        public static void main (String[] args) {
                new SimpleThread("Patra").start();
                new SimpleThread("Athina").start();
        }
}
```

(δημιουργία νημάτων ως στιγμιότυπα της SimpleThread και κλήση της start)

ΕΚΤΕΛΕΣΗ ΝΗΜΑΤΩΝ

0 Patra
0 Athina
1 Athina
1 Patra
2 Patra
2 Athina
3 Athina
3 Patra
4 Patra
4 Athina
5 Patra
5 Athina
6 Athina
6 Patra
7 Patra
7 Athina
8 Athina
8 Patra
9 Athina
9 Patra
DONE! Athina
DONE! Patra

ΝΗΜΑΤΑ ΜΕΣΩ ΤΗΣ RUNNABLE

Διαδικασία δημιουργίας νήματος

- Δημιουργία κλάσης που υλοποιεί την Runnable
- Υλοποίηση της run στο σώμα της κλάσης.
- Δημιουργία στιγμιοτύπου της κλάσης
- Δημιουργία στιγμιοτύπου της Thread με όρισμα το στιγμιότυπο της κλάσης
- Κλήση της start από το στιγμιότυπο της Thread

ΝΗΜΑΤΑ ΜΕΣΩ ΤΗΣ RUNNABLE (1)

1. Δημιουργία κλάσης – 2. Υλοποίηση run

```
public class CountUp implements Runnable {
    public void run() {
      for (int i=0; i < 0; i++)
        System.out.println("CountUp-" + i);
      }
}
```

3. Δημιουργία στιγμιοτύπου της CountUp

```
CountUp counter1 = new CountUp();
```

4. Δημιουργία στιγμιοτύπου Thread με όρισμα δημιουργού counter1

```
Thread thread1 = new Thread(counter1);
```

5. Κλήση μεθόδου start

```
thread1.start();
```

ΝΗΜΑΤΑ ΜΕΣΩ ΤΗΣ RUNNABLE (2)

```
public class Clock extends Applet implements Runnable {
    private Thread clockThread = null;
    public void start() {
        if (clockThread == null) {
            clockThread = new Thread(this, "Clock");
            clockThread.start();
        }
    }
    public void run() {
        Thread myThread = Thread.currentThread();
        while (clockThread == myThread) {
            repaint();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e){}
    }
}
```

ΣΥΓΧΡΟΝΙΣΜΟΣ ΝΗΜΑΤΩΝ (1)

- Μέχρι τώρα αναφερθήκαμε σε νήματα εκτελούνται ανεξάρτητα το ένα από το άλλο. Δεν απαιτείται το ένα να γνωρίζει σχετικά με τις εργασίες των άλλων.
- Υπάρχουν όμως περιπτώσεις που τα νήματα πρέπει να μοιράζονται δεδομένα. Στην περίπτωση αυτή πρέπει να εξασφαλίσουμε ότι το ένα νήμα δεν θα αλλάξει τα δεδομένα ενόσω τα διαχειρίζεται άλλο νήμα.
- Περίπτωση που μπορεί να δημιουργηθεί τέτοιο πρόβλημα υπάρχει όταν διαφορετικά αντικείμενα μιας κλάσης διαχειρίζονται στατικές μεταβλητές ή στατικές μεθόδους της κλάσης.

ΣΥΓΧΡΟΝΙΣΜΟΣ ΝΗΜΑΤΩΝ (2)

- Για την αντιμετώπιση τέτοιων καταστάσεων η Java επιτρέπει το κλείδωμα αντικειμένων και μεθόδων με τη χρήση της δεσμευμένης λέξης synchronized.
- Έτσι επιτυγχάνεται ο λεγόμενος συγχρονισμός νημάτων, ο οποίος επιτρέπει την προσπέλαση μιας μεθόδου (ή ενός αντικειμένου) σ' ένα μόνο νήμα κάθε φορά. Μια μέθοδος συγχρονίζεται ως εξής:

  public synchronized void xmethod {…}

- Για να προσπελάσει ένα νήμα μια μέθοδο που έχει δηλωθεί ως synchronized, πρέπει να περιμένει να τελειώσει το τρέχον νήμα (blocked). (Έλεγχος μέσω monitor object).

ΣΥΓΧΡΟΝΙΣΜΟΣ ΝΗΜΑΤΩΝ (3)

Μπορούμε επίσης να δηλώσουμε ως συγχρονισμένο ένα τμήμα κώδικα στο σώμα μιας μεθόδου, αντί όλης της μεθόδου:

```
public int ymethod {
        …
   synchronized (this) {
                    …
    }
}
```

ΣΥΓΧΡΟΝΙΣΜΟΣ ΝΗΜΑΤΩΝ (4)

Για μεγαλύτερη ασφάλεια και αποφυγή αδιεξόδων, τα μέρη του κώδικα που έχουν δηλωθεί ως synchronized πρέπει να περιλαμβάνουν κλήσεις της μεθόδου wait():

```
try {
  <κώδικας>
   wait();
}
catch (InterruptedException e) {
}
```

Η κλήση της wait προκαλεί αναστολή της εκτέλεσης του νήματος, μέχρις ότου ένα άλλο νήμα καλέσει τη μέθοδο notify() ή notifyAll() (μεταθέτουν ένα ή όλα τα νήματα από κατάσταση αναστολής σε κατάσταση ετοιμότητας).

ΠΡΟΤΕΡΑΙΟΤΗΤΕΣ ΝΗΜΑΤΩΝ

- Υπάρχουν περιπτώσεις που θέλουμε τα νήματα να μην είναι ισοδύναμα, αλλά κάποια να εκτελούνται περισσότερο χρόνο από άλλα.
- Αυτό επιτυγχάνεται με τον καθορισμό προτεραιότητας για κάθε νήμα (1-10). Η κανονική προτεραιότητα είναι 5.
- Η Thread διαθέτει τρεις σταθερές MAX_PRIORITY, MIN_PRIORITY και NORM_PRIORITY με τιμές 10, 1 και 5.
- Η προτεραιότητα ενός νήματος τίθεται με την setPriority (int priority), ενώ με την getPriority() μπορούμε να ανιχνεύσουμε την τρέχουσα τιμή προτεραιότητας ενός νήματος.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.1

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 4: JAVA: ΔΙΕΠΑΦΕΣ ΧΡΗΣΤΗ, ΓΡΑΦΙΚΑ, APPLETS

Γραφικές Διεπαφές Χρήστη - awt

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΓΡΑΦΙΚΕΣ ΔΙΕΠΑΦΕΣ ΧΡΗΣΤΗ -AWT

- Γραφική διεπαφή χρήστη είναι ένα γραφικός τρόπος επικοινωνίας ενός προγράμματος με τον χρήστη του. Στηρίζεται στην έννοια των παραθύρων. Η επικοινωνία γίνεται μέσω μενού, κουμπιών κλπ.

- **Εργαλεία**
  - Χρήση εργαλειοθήκης awt - abstract windowing tool (στάνταρντ: java 1 - σε αυτό θα αναφερθούμε εδώ)
  - Χρήση εργαλειοθήκης Swing (εξέλιξη του awt: java 2)
  - Είναι δύο σύνολα κλάσεων για δημιουργία γραφικών διεπαφών χρήστη

ΓΡΑΦΙΚΕΣ ΔΙΕΠΑΦΕΣ ΧΡΗΣΤΗ

Εργαλειοθήκη AWT

ΔΗΜΙΟΥΡΓΙΑ ΔΙΕΠΑΦΗΣ

- 1. Δημιουργία υποδοχέα
  - Δημιουργούμε το βασικό παράθυρο της εφαρμογής
- 2. Δημιουργία συστατικών
  - Δημιουργούμε τα συστατικά του παραθύρου
- 3. Καθορισμός διαχειριστή διάταξης
  - Προσδιορίζουμε τον τρόπο διάταξης των συστατικών στο βασικό παράθυρο
- 4. Προσθήκη συστατικών
  - Εισάγουμε τα συστατικά στο παράθυρο

ΔΗΜΙΟΥΡΓΙΑ ΥΠΟΔΟΧΕΑ ΠΑΡΑΘΥΡΟΥ

- **Διαδικασία**
  - Δημιουργία υποκλάσης της Frame
  - Δημιουργία στιγμιοτύπου της υποκλάσης στη μέθοδο main της υποκλάσης
- **Δημιουργοί της Frame**
  - Frame() → Δημιουργία παραθύρου χωρίς τίτλο
  - Frame(String) → Δημιουρ. παραθ. με τίτλο
- Καλούνται μέσω του super από το δημιουργό της υποκλάσης
- Αρχικά το παράθυρο δεν είναι ορατό και οι διαστάσεις του είναι μηδενικές

ΔΗΜΙΟΥΡΓΙΑ ΥΠΟΔΟΧΕΑ ΠΑΡΑΘΥΡΟΥ

```
import java.awt.*;

class MyApp extends Frame{
  public MyApp(String title){
        super(title)
        <δημιουργία διαχειριστή διάταξη>
        <δημιουργία συστατικών>
        <προσθήκη συστατικών>
  }

  public static void main(){
    MyApp app1 = new MyApp("Application Window");
      <διαχείριση παραθύρου>
  }
}
```

Συμπερίληψη Εργαλειοθήκης

Δημιουργός

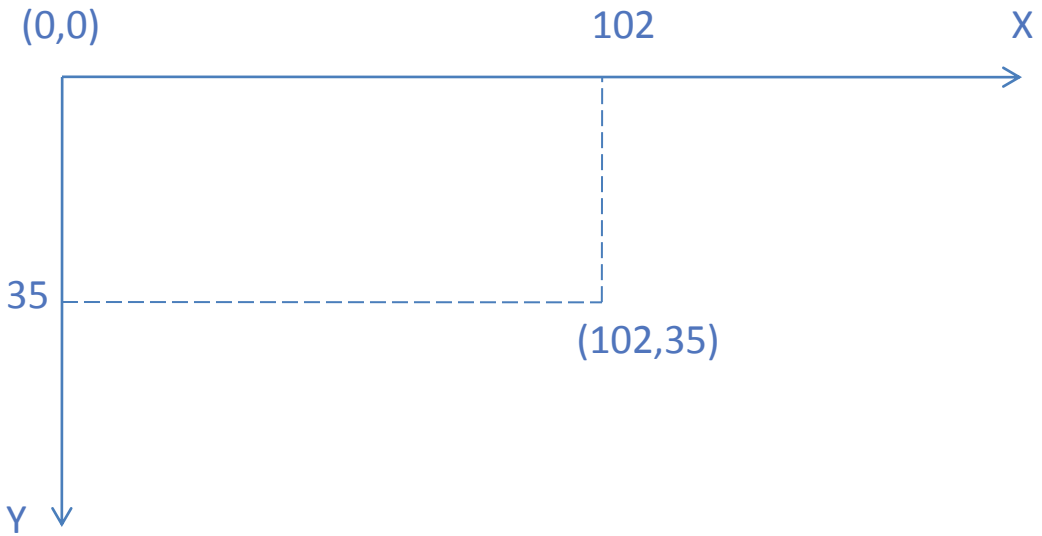
Δημιουργία στιγμιότυπου

ΜΕΘΟΔΟΙ ΧΕΙΡΙΣΜΟΥ ΠΑΡΑΘΥΡΟΥ

- **setSize(int w, int h)**
- **setSize(java.awt.Dimension d)**
  - καθορισμός διαστάσεων παραθύρου (σε pixels)
- **setBounds(int x,int y,int w,int h)**
- **setBounds(java.awt.Rectangle r)**
  - καθορισμός διαστάσεων και θέσης παραθύρου(σε pixels)
- **pack()**
  - μικρότερο δυνατό μέγεθος παραθ.
- **setVisible(boolean)**
  - εμφάνιση ή απόκρυψη παραθύρου

ΣΥΣΤΗΜΑ ΣΥΝΤΕΤΑΓΜΕΝΩΝ

- Αρχή αξόνων (0,0) η πάνω αριστερή γωνία είτε της οθόνης είτε του παραθύρου, είτε του χώρου του Applet
- Συντεταγμένες: (πλάτος, ύψος)



ΔΙΑΧΕΙΡΙΣΤΕΣ ΔΙΑΤΑΞΗΣ (1)

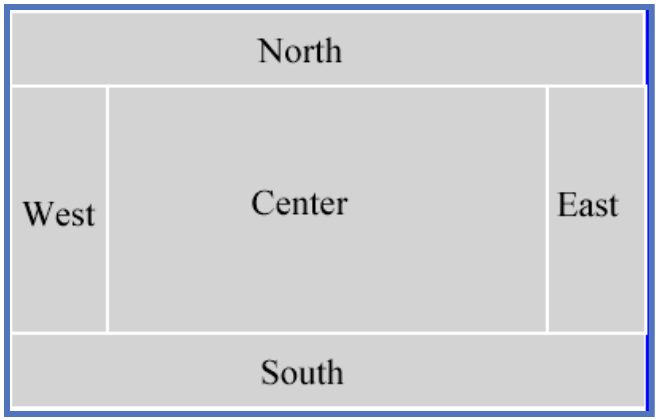
Διάταξη ροής (Κλάση: FlowLayout)

- **FlowLayout()**
  - συστατικά το ένα μετά το αλλο (αριστ. δεξιά) στο κέντρο
- **FlowLayout(int)**
  - στοίχηση ανάλογα με το int (FlowLayout.LEFT/RIGHT/CENTER)
- **FlowLayout(int,int,int)**
  - + οριζόντιο και κατακόρυφο διάκενο σε pixels (2ο, 3ο int)

ΔΙΑΧΕΙΡΙΣΤΕΣ ΔΙΑΤΑΞΗΣ (2)

Περιφερειακή διάταξη (Κλάση: BorderLayout)

- **BorderLayout()**
  - διάταξη χωρίς διάκενα
- **BorderLayout(int,int)**
  - οριζόντιο και κατακόρυφο διάκενο σε pixels



ΔΙΑΧΕΙΡΙΣΤΕΣ ΔΙΑΤΑΞΗΣ (3)

Διάταξη πλέγματος (Κλάση: GridLayout)

- **GridLayout(int,int)**
  - συστατικά σε κελιά πλέγματος int x int.
  - Τα Συστατικά καταλαμβάνουν όλο τον χώρο των κελιών
  - Όλα τα κελιά έχουν το ίδιο μέγεθος
  - Αν ένας (όχι και οι δύο) int είναι 0 → το πλέγμα επεκτείνεται απεριόριστα προς αυτή την κατεύθυνση καθώς προσθέτουμε συστατικά
- **GridLayout(int,int,int,int)**
  - + οριζόντιο και κατακόρυφο διάκενο (3ο, 4ο int) σε pixels
- **GridLayout()**
  - Πλέγμα μιας γραμμής και απεριόριστων στηλών

ΔΙΑΧΕΙΡΙΣΤΕΣ ΔΙΑΤΑΞΗΣ (4)

Καθορισμός Διαχειριστή Διάταξης

- **1. Δημιουργία διαχειριστή**
  (δηλ. Δημιουργία στιγμιοτύπου μέσω των δημιουργών των προηγούμενων κλάσεων)
- **2. Ενεργοποίηση διαχειριστή**
  - Μέθοδος setLayout (<layout instance>)

ΥΠΟΔΟΧΕΑΣ PANEL

- **Panel:** Ένας υποδοχέας που περιέχει συστατικά
- Είναι σαν ένας αόρατος υποδοχέας που πρέπει να τοποθετηθεί σε ένα top-level υποδοχέα (όπως ένα Frame)
- Ένα panel μπορεί να είναι ένθετο (nested) μέσα σε ένα άλλο panel.
- Δύο Δημιουργοί:
  - Panel() → Default layout manager: **FlowLayout**
  - Panel(LayoutManager)

ΚΑΘΟΡΙΣΜΟΣ ΓΡΑΜΜΑΤΟΣΕΙΡΑΣ

1. Δημιουργία στιγμιοτύπου

Font(“<font>”, <style>, <size>);

<style>: Font.PLAIN/BOLD/ITALIC

Π.χ. Font f1 = new Font(“Helvetica”, Font.BOLD +Font.ITALIC, 14);

2. Ανάθεση γραμματοσειράς

setFont(<font instance>);

Π.χ. setFont (f1);

ΠΡΟΣΘΗΚΗ ΣΥΣΤΑΤΙΚΩΝ

Μέθοδοι:

add(c)

add(String, c)

(περίπτωση διαχειριστή περιφερειακής διάταξης)

ΚΑΘΟΡΙΣΜΟΣ ΧΡΩΜΑΤΟΣ ΥΠΟΒΑΘΡΟΥ

setBackground(Color.<color>);

Π.χ. setBackground (Color.green);

ΕΤΙΚΕΤΕΣ

Κλάση: Label

Δημιουργοί:

Label()

Label(String)

Label(String, int)

(όπου int → Label.LEFT/CENTER/RIGHT)

Project BlueJ: GuiLab

ΠΛΗΚΤΡΑ/ΚΟΥΜΠΙΑ

Κλάση: Button

Δημιουργοί:

Button()

Button(String)

Μέθοδοι:

setLabel(String)

getLabel()

Project BlueJ: GuiBut, GuiLabBut

ΠΛΑΙΣΙΑ ΕΛΕΓΧΟΥ

Κλάση: Checkbox

Δημιουργοί:

Checkbox()

Checkbox(String)

Μέθοδοι:

setState(boolean)

getState()

Project BlueJ: GuiCheck

Δημιουργία ανεξάρτητων πλαισίων (μπορεί να είναι επιλεγμένα οποιαδήποτε κάθε φορά).

ΟΜΑΔΟΠΟΙΗΜΕΝΑ ΠΛΑΙΣΙΑ ΕΛΕΓΧΟΥ

Κλάση: CheckboxGroup

Δημιουργός:

CheckboxGroup()

Δημιουργία ομαδοποιημένων πλαισίων (μπορεί να είναι επιλεγμένο ένα κάθε φορά).

Διαδικασία:

1. Δημιουργία στιγμιοτύπου ομάδας πλαισίων
2. Συσχετισμός πλαισίων με το στιγμιότυπο Δημιουργός Checkbox(String, boolean, instance)

ΠΑΡΑΔΕΙΓΜΑ

Βήμα 1 :

```
CheckboxGroup lang = new CheckboxGroup();
```

Βήμα 2 :

```
Checkbox c1 = new Checkbox (“Pascal”, false, lang);
Checkbox c2 = new Checkbox (“Java”, false, lang);
Κλπ
```

Project BlueJ: GuiCheckGroup

ΛΙΣΤΕΣ ΕΠΙΛΟΓΗΣ (1)

Κλάση: Choice

Δημιουργός: Choice()

Διαδικασία:

1. Δημιουργία στιγμιοτύπου λίστας

```
Choice langCh = new Choice ();
```

2. Προσθήκη στοιχείων λίστας

```
langCh.add("Pascal");
```

3. Προσθήκη της λίστας στον υποδοχέα

```
add(langCh);
```

Project BlueJ: GuiList1

Δημιουργία αναδιπλούμενης λίστας (μπορεί να είναι επιλεγμένο ένα στοιχείο κάθε φορά).

ΛΙΣΤΕΣ ΕΠΙΛΟΓΗΣ (2)

Μέθοδοι:

`getItem(int)`
Επιστρέφει το στοιχείο στη θέση int (πρώτη θέση: 0)

`getItemCount()`
Επιστρέφει τον αριθμό των στοιχείων της λίστας

`select(int)`
Επιλέγει το στοιχείο στη θέση int

`select(String)`
Επιλέγει το πρώτο στοιχείο με όνομα το String

`getSelectedIndex()`
Επιστρέφει τη θέση του τρέχοντος επιλεγμένου στοιχείου

`getSelectedItem()`
Επιστρέφει το όνομα του τρέχοντος επιλεγμένου στοιχείου

ΛΙΣΤΕΣ ΚΥΛΙΣΗΣ (1)

Δημιουργία μη αναδιπλούμενης λίστας (μπορεί να είναι επιλεγμένα περισσότερα από ένα στοιχείο κάθε φορά).

Κλάση: List

Δημιουργοί:

`List()`

(Δημιουργεί κενή λίστα κύλισης, επιτρέπει επιλογή μόνο ενός στοιχείου)

`List(int, boolean)`

(Δημιουργεί λίστα με καθορισμένο αριθμό: int στοιχείων, επιτρέπει πολλαπλή επιλογή μόνο αν το 2ο όρισμα είναι true)

Διαδικασία:

ίδια με λίστες επιλογής

Project BlueJ: GuiList2

ΛΙΣΤΕΣ ΚΥΛΙΣΗΣ (2)

Μέθοδοι:

Τις ίδιες με μιας λίστας επιλογής συν:

`getSelectedIndexes()`

Επιστρέφει τις θέσεις των επιλεγμένων στοιχείων (περίπτωση πολλαπλής επιλογής)

`getSelectedItems()`

Επιστρέφει τα ονόματα των επιλεγμένων στοιχείων (περίπτωση πολλαπλής επιλογής)

ΠΕΔΙΑ ΚΕΙΜΕΝΟΥ (1)

Κλάση: TextField

Δημιουργοί:

`TextField()`

(Δημιουργεί κενό πεδίο κειμένου χωρίς καθορισμένο μήκος)

`TextField(String)`

(Δημιουργεί ένα πεδίο με περιεχόμενο String, χωρίς καθορισμένο μήκος)

`TextField(String, int)`

(Δημιουργεί ένα πεδίο με περιεχόμενο String, και καθορισμένο μήκος, για int χαρακτήρες)

Project BlueJ: GuiText1

ΠΕΔΙΑ ΚΕΙΜΕΝΟΥ (2)

Μέθοδοι:

`setEchoChar(char)`

Απόκρυψη πληκτρολογούμενων χαρακτήρων με ένα κοινό χαρακτήρα.

`getText()`

Επιστρέφει το κέιμενο του πεδίου

`setText(String)`

Γεμίζει το πεδίο με το String

`setEditable(boolean)`

Κάνει το πεδίο επεξεργάσιμο/προεπιλεγμένο (με true) ή όχι (με false)

`isEditable()`

Επιστρέφει true αν το πεδίο είναι επεξεργάσιμο και false αν όχι

ΠΕΡΙΟΧΕΣ ΚΕΙΜΕΝΟΥ (1)

Κλάση: TextArea

Δημιουργοί:

`TextArea()`

(Δημιουργεί κενή περιοχή κειμένου χωρίς καθορισμένο ύψος, πλάτος)

`TextArea(int, int)`

(Δημιουργεί κενή περιοχή κειμένου με καθορισμένο ύψος, πλάτος)

`TextArea(String)`

(Δημιουργεί μια περιοχή κειμένου που περιέχει το String χωρίς καθορισμένο ύψος, πλάτος)

`TextArea(String, int, int)`

(Δημιουργεί μια περιοχή κειμένου που περιέχει το String με καθορισμένο ύψος, πλάτος)

Project BlueJ: GuiText2

ΠΕΡΙΟΧΕΣ ΚΕΙΜΕΝΟΥ(2)

Μέθοδοι:

Τις ίδιες με των πεδίων κειμένου συν

`insert(String, int)`

Εισάγει το String στη θέση που δείχνει το int

`replaceRange(String, int, int)`

Αντικαθιστά το κείμενο μεταξύ των θέσεων int με το String

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.1

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 4: JAVA: ΔΙΕΠΑΦΕΣ ΧΡΗΣΤΗ, ΓΡΑΦΙΚΑ, APPLETS

Γραφικές Διεπαφές Χρήστη - Swing

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΓΡΑΦΙΚΕΣ ΔΙΕΠΑΦΕΣ ΧΡΗΣΤΗ

Εργαλειοθήκη SWING

ΓΡΑΦΙΚΕΣ ΔΙΕΠΑΦΕΣ ΧΡΗΣΤΗ-SWING (1)

Πλεονεκτήματα έναντι του awt

- Περισσότερα συστατικά
- Επεκτεταμένα χαρακτηριστικά συστατικών
- Καλύτερη έμφάνιση και αίσθηση
- Καλύτερος χειρισμός συμβάντων
- Πιό συμβατά προγράμματα στις διάφορες πλατφόρμες (συστατικά πλήρως υλοποιημένα σε java)

ΓΡΑΦΙΚΕΣ ΔΙΕΠΑΦΕΣ ΧΡΗΣΤΗ-SWING (2)

Ομοιότητες-Διαφορές από το awt

- Η βασική δομή του προγράμματος παραμένει.
- Οι βασικές έννοιες ‘υποδοχέας’, ‘τομέας’ ‘συστατικό’, και ‘διαχειριστής διάταξης’ παραμένουν.
- Η διαδικασία προσθήκης συστατικών διαφέρει.
- Η χρήση ενός συστατικού παραμένει ίδια.
- Αλλάζουν ελαφρώς τα ονόματα των κλάσεων (προσθήκη ενός «J» μπροστά συνήθως).
- Πακέτο: javax.swing

ΔΗΜΙΟΥΡΓΙΑ ΥΠΟΔΟΧΕΑ (1)

Διαδικασία (ίδια)

- Δημιουργία υποκλάσης της JFrame.
- Δημιουργία στιγμιοτύπου της υποκλάσης στη μέθοδο main της υποκλάσης.

Δημιουργοί της JFrame

JFrame()→Δημιουργία παραθύρου χωρίς τίτλο

JFrame (String) →Δημιουρ. παραθ. με τίτλο

(Καλούνται μέσω του super απο τον
δημιουργό της υποκλάσης)

ΔΗΜΙΟΥΡΓΙΑ ΥΠΟΔΟΧΕΑ (2)

Βασική δομή προγ/τος:

```
import java.awt.*;
import javax.swing.*;

class MyApp extends JFrame {
   public MyApp (String title) {
      super(title);
     <δημιουργία διαχειριστή διάταξης>
     <δημιουργία-προσθήκη συστατικών>
   }

   public static void main() {
      MyApp app1 = new MyApp ("Application Window");
     <διαχείριση παραθύρου>
   }
}
```

Συμπερίληψη εργαλειοθήκης

Δημιουργός

Δημιουργία στιγμιοτύπου

ΒΑΣΙΚΟΙ ΜΕΘΟΔΟΙ ΧΕΙΡΙΣΜΟΥ ΠΑΡΑΘΥΡΟΥ

pack() → μικρότερο δυνατό μέγεθος παραθ.

setVisible(boolean) → εμφάνιση/απόκρυψη παραθύρου

setBounds(int, int, int, int) → προσδιορισμός θέσης στην οθόνη (x, y, w, h)

ΔΙΑΧΕΙΡΙΣΤΕΣ ΔΙΑΤΑΞΗΣ

Καθορισμός Διαχειριστή Διάταξης

- Ισχύουν τα του awt
- Επιπλέον υπάρχει διαχειριστής εμφάνισης και αίσθησης

ΠΡΟΣΘΗΚΗ ΣΥΣΤΑΤΙΚΩΝ-ΤΟΜΕΑΣ ΠΕΡΙΕΧΟΜΕΝΟΥ (1)

- Η προσθήκη συστατικών δεν γίνεται απ' ευθείας στον κύριο υποδοχέα, αλλά σ' εναν ενδιάμεσο υποδοχέα, που ονομάζεται τομέας περιεχομένου (content pane)
- Βασικά, ο κύριος υποδοχέας (πλαίσιο παραθύρου) αποτελείται από διάφορους τομείς. Ο κεντρικός τομέας είναι ο τομέας περιεχομένου

ΠΡΟΣΘΗΚΗ ΣΥΣΤΑΤΙΚΩΝ-ΤΟΜΕΑΣ ΠΕΡΙΕΧΟΜΕΝΟΥ (2)

Διαδικασία

1. Δημιουργία ενός (στιγμιοτύπου) τομέα (δημιουργός κλάσης JPanel).
2. Δημιουργία και προσθήκη συστατικών στον τομέα αυτό (add(Component)).
3. Προσθήκη του τομέα (χρήση add) ή καθορισμός του τομέα ως τομέα περιεχομένου (setContentPane(Container))

```
JFrame f = new JFrame ();
JButton b = new JButton ();
Container contentPane = f.getContentPane ();
contentPane.add(b );
```

Όλα τα συστατικά έχουν κοινή υπερκλάση την Jcomponent, απ' όπου κληρονομούν μεθόδους (setEnabled, setVisible, setSize κλπ)

ΕΤΙΚΕΤΕΣ

Κλάση: JLabel

Δημιουργοί:

```
JLabel(String)
JLabel(String, int)
```

(όπου int → SwingConstants.LEFT/CENTER/RIGHT)

```
JLabel(String, Icon, int)
```

Δημιουργία εικονιδίου

```
ImageIcon ic1 = new ImageIcon("icon1.gif");
```

ΠΛΗΚΤΡΑ/ΚΟΥΜΠΙΑ

Κλάση:

```
JButton
```

Δημιουργοί:

```
JButton(String)
JButton(Icon)
JButton(String, Icon)
```

Μέθοδοι:

```
setLabel(String)
getLabel()
```

ΠΛΑΙΣΙΑ ΕΛΕΓΧΟΥ

Κλάση:

JCheckbox

Δημιουργοί:

```
JCheckbox(String)
JCheckbox(String, boolean)
JCheckbox(Icon)
JCheckbox(Icon, boolean)
JCheckbox(String, Icon)
JCheckbox(String, Icon, boolean)
```

Μέθοδοι:

```
setEnabled(boolean)
```

ΟΜΑΔΟΠΟΙΗΜΕΝΑ ΠΛΑΙΣΙΑ ΕΛΕΓΧΟΥ/ΚΟΥΜΠΙΑ ΕΠΙΛΟΓΗΣ

Κλάση: ButtonGroup

Δημιουργός: ButtonGroup()

Διαδικασία:

1. Δημιουργία στιγμιοτύπου ButtonGroup
2. Δημιουργία πλαισίων/κουμπιών
3. Προσθήκη πλαισίων/κουμπιών: add(Component)

ΠΑΡΑΔΕΙΓΜΑ

Βήμα 1

```
ButtonGroup lang = new ButtonGroup();
```

Βήμα 2

```
JCheckbox c1 = new JCheckbox ("Pascal", false);
JCheckbox c2 = new JCheckbox ("Java", false);
```

Βήμα 3

```
lang.add(c1);
lang.add(c2);
```

ΚΟΥΜΠΙΑ ΕΠΙΛΟΓΗΣ

Κλάση: `JRadioButton`

Δημιουργοί:

```
JRadioButton(String)
JRadioButton(String, boolean)
JRadioButton(Icon)
JRadioButton(Icon, boolean)
JRadioButton(String, Icon)
JRadioButton(String, Icon, boolean)
```

Μέθοδοι:

```
setEnabled(boolean)
```

ΛΙΣΤΕΣ ΕΠΙΛΟΓΗΣ (1)

Κλάση: `JComboBox`

Δημιουργός: `JComboBox()`

Διαδικασία:

1. Δημιουργία σύνθετου πλαισίου

```
JComboBox langBox = new JComboBox();
```

2. Προσθήκη στοιχείων

```
langBox.addItem("Pascal");
```

3. Μετατροπή σύνθετου πλαισίου σε λίστα

```
setEditable(false);
```

ΛΙΣΤΕΣ ΕΠΙΛΟΓΗΣ (2)

Μέθοδοι:

`getItem(int)`

Επιστρέφει το στοιχείο στη θέση int (πρώτη θέση: 0)

`getItemCount()`

Επιστρέφει τον αριθμό των στοιχείων της λίστας

`setSelectedIndex(int)`

Επιλέγει το στοιχείο στη θέση int

`getSelectedIndex()`

Επιστρέφει τη θέση του τρέχοντος επιλεγμένου στοιχείου

`getSelectedItem()`

Επιστρέφει το όνομα του τρέχοντος επιλεγμένου στοιχείου

ΓΡΑΜΜΕΣ ΚΥΛΙΣΗΣ

Κλάση: `JScrollBar`

Επιλογή τιμής με ολίσθηση ενός πλαισίου.

Δημιουργοί:

`JScrollBar(int)`

(int → SwingConstants.HORIZONTAL/VERTICAL)

`JScrollBar(int, int, int, int, int)`

(int → SwingConstants.HORIZONTAL/VERTICAL)

(int →αρχική τιμή)

(int →μέγεθος πλαισίου κύλισης)

(int →ελάχιστη τιμή)

(int →μέγιστη τιμή)

ΠΕΔΙΑ ΚΕΙΜΕΝΟΥ (1)

Κλάση: `JTextField, JPasswordField`

Δημιουργοί:

```
JTextField(int)
JTextField(String, int)
JPasswordField(int)
JPasswordField(String, int)
(Χρήση της setEchoChar(char))
```

ΠΕΡΙΟΧΕΣ ΚΕΙΜΕΝΟΥ

Κλάση: JTextArea

Δημιουργοί:

```
JTextArea(int, int)
JTextArea(String)
JTextArea(String, int, int)
```

ΚΑΘΟΡΙΣΜΟΣ ΑΙΣΘΗΣΗΣ ΚΑΙ ΕΜΦΑΝΙΣΗΣ (1)

- Γίνεται μέσω της κλάσης **UIManager**
- Επιλογή από τρεις τύπους
  - Τύπος Windows
  - Τύπος Motif X-Windows
  - Τύπος Metal (java)
- Η επιλογή γίνεται από τη μέθοδο `setLookAndFeel(LookAndFeel)`
- Δημιουργία στιγμιοτύπου **LookAndFeel:**

  (α) `getCrossPlatformLookAndFeelClassName()`

  (για επιλογή metal)

  (β) `getSystemLookAndFeelClassName()`

  (για επιλογή αίσθησης & εμφάνισης του λειτουργικού συστήματος που χρησιμοποιείται)

ΚΑΘΟΡΙΣΜΟΣ ΑΙΣΘΗΣΗΣ ΚΑΙ ΕΜΦΑΝΙΣΗΣ (2)

Η **`setLookAndFeel`** χρειάζεται χειρισμό εξαιρέσεων.

```
try {
   UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName();
}
catch (Exception e){
     System.err.println("Can't set look and feel: "+e);
}
```

ΝΕΑ ΣΤΟΙΧΕΙΑ: ΠΡΟΤΥΠΑ ΠΑΡΑΘΥΡΑ ΔΙΑΛΟΓΟΥ

Κλάση: `JOptionPane`

Υπάρχουν τέσσερις τύποι παραθύρων διαλόγου

- Επιβεβαίωσης (ConfirmDialog)
- Εισόδου (InputDialog)
- Μηνύματος (MessageDialog)
- Επιλογής (OptionDialog)

ΝΕΑ ΣΤΟΙΧΕΙΑ: ΠΑΡΑΘΥΡΑ ΕΠΙΒΕΒΑΙΩΣΗΣ (1)

Ερώτηση με κουμπιά Yes, No, Cancel

Μέθοδος 1:

`showConfirmDialog(Component, Object)`

Component: ο υποδοχέας (null: κέντρο οθόνης)

Object: String, συστατικό ή Icon

Επιστρέφει: μία από τρεις ακέραιες τιμές:
`(YES_OPTION, NO_OPTION, CANCEL_OPTION)`

Π.χ.

```
int answer = JOptionPane.showConfirmDialog(null, "Are you sure to delete the file? ");
```

ΝΕΑ ΣΤΟΙΧΕΙΑ: ΠΑΡΑΘΥΡΑ ΕΠΙΒΕΒΑΙΩΣΗΣ (2)

Ερώτηση με κουμπιά Yes, No, Cancel

Μέθοδος 2:

```
showConfirmDialog(Component, Object, String, int, int)
```

String: κείμενο γραμμής τίτλου παραθύρου

int: `YES_NO_CANCEL_OPTION, YES_NO_OPTION`

int: `ERROR_MESSAGE, INFORMATION_MESSAGE, PLAIN_MESSAGE, QUESTION_MESSAGE, WARNING_MESSAGE`

Π.χ.

```
int answer = JOptionPane.showConfirmDialog(null, “Error reading file”, “File Input Error”,
    JOptionPane.YES_NO_OPTION, JOptionPane. ERROR_MESSAGE);
```

ΝΕΑ ΣΤΟΙΧΕΙΑ: ΠΑΡΑΘΥΡΑ ΕΙΣΟΔΟΥ

Ερώτηση με πεδίο κειμένου για απάντηση

Μέθοδος 1:

`showInputDialog(Component, Object)`

Επιστρέφει: String (το κείμενο της απάντησης)

Π.χ.

```
string answer = JOptionPane.showInputDialog(null, "Enter your title:");
```

Μέθοδος 2:

`showInputDialog(Component, Object, String, int)`

Π.χ.

```
string answer = JOptionPane.showInputDialog(null, "Enter your title:",
                                            "Enter Title", JOptionPane. QUESTION_MESSAGE);
```

ΝΕΑ ΣΤΟΙΧΕΙΑ: ΠΑΡΑΘΥΡΑ ΜΗΝΥΜΑΤΟΣ

Εμφάνιση πληροφορίας

Μέθοδος 1:

```
showMessageDialog(Component, Object)
```

Επιστρέφει: Δεν επιστρέφει τιμή

Π.χ.

```
JOptionPane.showMessageialog(null, "Title is missing");
```

Μέθοδος 2:

```
showMessageDialog(Component, Object, String, int)
```

Π.χ.

```
int answer = JOptionPane.showInputDialog(null, "The title is missing",
    "Missing Componnet Message", JOptionPane. WARNING_MESSAGE);
```

ΝΕΑ ΣΤΟΙΧΕΙΑ: ΠΑΡΑΘΥΡΑ ΕΠΙΛΟΓΗΣ (1)

- **Πιο περίπλοκο από τα άλλα**
- **Συνδυάζει χαρακτηριστικά όλων των άλλων**

Μέθοδος:

```
showOptionDialog(Component, Object, String, int, int, Icon,
                                                Object[], Object)
```

Icon: στιγμιότυπο Icon αντί του υπάρχοντος

Object[]: πίνακας που περιέχει τα συστατικά που είναι οι επιλογές του παραθύρου.

Object: Η προεπιλεγμένη επιλογή, αν δεν χρησιμοποιηθούν τα `YES_NO_CANCEL_OPTION, YES_NO_OPTION`

Επιστρέφει: int

ΠΑΡΑΘΥΡΑ ΕΠΙΛΟΓΗΣ (2)

Π.χ.

```
Jbutton fav [] = new Jbutton[3];
fav[0] = new Jbutton("C++");
fav[1] = new Jbutton("Java");
fav[2] = new Jbutton("None");
int answer = JOptionPane.
      showInputDialog(null,
                  "What is your favorite programming language?",
                  "Favorite Language",
                  0,
                  JOptionPane.INFORMATION_MESSAGE,
                  null,
                  fav,
                  fav[2]);
```

ΧΕΙΡΙΣΜΟΣ ΣΥΜΒΑΝΤΩΝ

- Μια κλάση που θέλει να αποκρίνεται σε συμβάντα πρέπει να υλοποιεί αντίστοιχη διεπαφή (interface), που λέγεται **ακροατής συμβάντων** (event listener).
- Κάθε ακροατής χειρίζεται ένα συγκεκριμένο είδος συμβάντος
- Μια κλάση μπορεί να υλοποιήσει όσους ακροατές χρειάζεται.

Διαδικασία

1. Υλοποίηση ακροατή συμβάντων
2. Συσχέτιση όποιων συστατικών επιθυμούμε με ένα ή περισσότερους ακροατές συμβάντων

ΒΑΣΙΚΟ ΠΛΑΙΣΙΟ ΠΡΟΓ/ΤΟΣ

```
import java.awt.GridLayout;
import javax.swing.*;
import java.awt.event;
class MyApp extends JFrame {
   public MyApp (String title) {
      super(title);
      <δημιουργία διαχειριστή διάταξης>
      <δημιουργία-προσθήκη συστατικών>}

  public static void main() {
   MyApp app = new MyApp ("Application Window");
   WindowListener L = new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
         System.exit(0);}};
   app.addWindowListener(L);
  app.pack();
  app.setVisible(true);} }
```

Πακέτο χειρισμού συμβάντων.

Κώδικας χειρισμού προτύπων συμβάντων: κλείνει την εφαρμογή όταν κλείνει το πλαίσιο (παράθυρο).

ΑΚΡΟΑΤΕΣ ΣΥΜΒΑΝΤΩΝ (1)

- `ActionListener`

  Συμβάντα ενέργειας: παράγονται από ενέργεια σε συστατικό,
  π.χ. πάτημα σε κουμπί

- `AdjustmentListener`

  Συμβάντα ρύθμισης: παράγονται από ρύθμιση συστατικού
  π.χ. μετακίνηση γραμμής κύλισης

- `FocusListener`

  Συμβάντα εστίασης: παράγονται από συστατικό που παίρνει ή χάνει την εστίαση, π.χ. πεδίο κειμένου

- `ItemListener`

  Συμβάντα στοιχείου: παράγονται όταν ένα στοιχείο,
  π.χ. σε πλαίσιο ελέγχου, αλλάζει

- `KeyListener`

  Συμβάντα πληκτρολογίου: παράγονται όταν εισάγουμε δεδομένα από το πληκτρολόγιο

ΑΚΡΟΑΤΕΣ ΣΥΜΒΑΝΤΩΝ (2)

- `MouseListener`

  Συμβάντα ποντικιού: παράγονται από πατήματα του ποντικιού

- `MouseMotionListener`

  Συμβάντα κίνησης ποντικιού: παράγονται από κίνηση του ποντικιού σε συστατικό)

- `TextListener`

  Συμβάντα κειμένου: παράγονται από μεταβολές σε κείμενο

- `WindowListener`

  Συμβάντα παραθύρου: παράγονται από μεταβολές σε παράθυρο, π.χ. ελαχιστοποίηση

ΣΥΣΧΕΤΙΣΗ ΑΚΡΟΑΤΩΝ ΣΥΜΒΑΝΤΩΝ ΜΕ ΣΥΣΤΑΤΙΚΑ (1)

- `addActionListener()` Jbutton, JCheckBox, JTextField, JRadioButton
- `addAdjustmentListener()` JScrollBar
- `addFocusListener()` όλα του Swing
- `addItemListener()` Jbutton, JCheckBox, JRadioButton
- `addKeyListener` όλα του Swing
- `addMouseListener` όλα του Swing
- `addMouseMotionListener` όλα του Swing
- `addTextListener` JTextField)
- `addWindowListwner` όλα των Jwindow, JFrame

ΣΥΣΧΕΤΙΣΗ ΑΚΡΟΑΤΩΝ ΣΥΜΒΑΝΤΩΝ ΜΕ ΣΥΣΤΑΤΙΚΑ (2)

Όλες οι μέθοδοι add.. παίρνουν ένα όρισμα: το αντικείμενο που «ακούει» τα συμβάντα αυτού του είδους. Η χρήση του this, στην περίπτωση αυτή, δηλώνει την τρέχουσα κλάση ως ακροατή.

Π.χ.

```
Jbutton b = new Jbutton("button");
b.addActionListener(this);
```

Μπορούμε να δηλώσουμε και ένα διαφορετικό αντικείμενο, αρκεί η κλάση του να υλοποιεί τη σωστή διεπαφή ακροατή.

ΜΕΘΟΔΟΙ ΧΕΙΡΙΣΜΟΥ ΣΥΜΒΑΝΤΩΝ

- Όταν συσχετίζουμε ένα ακροατή συμβάντων (διεπαφή) με μια κλάση, τότε η κλάση αυτή πρέπει να υλοποιεί όλες τις μεθόδους της αντίστοιχης διεπαφής.
- Κάθε τέτοια μέθοδος καλείται αυτόματα από το παραθυρικό σύστημα όταν συμβεί αντίστοιχο συμβάν.

ActionListener

ActionListener

Έχει μια και μοναδική μέθοδο, την actionPerformed(). Κάθε κλάση που υλοποιεί την ActionListener πρέπει να υλοποιεί την παρακάτω μέθοδο:

```
public void actionPerformed(ActionEvent e){
    <χειρισμός συμβάντος>
}
```

Η ActionEvent είναι υποκλάση της EventObject (πακέτο java.awt.event)

getSource()

Μπορεί να χρησιμοποιηθεί για να προσδιορίσουμε το συστατικό στο οποίο στάλθηκε ένα συμβάν (ή με άλλα λόγια το συστατικό που «άκουσε» το συμβάν).

Π.χ.

```
public void actionPerformed(ActionEvent, e){
    Object comp = e.getSource();
    if (comp == quitButton)
        quitProgram();
    else if (comp == sortRecords)
        sortRecords();
}
```

instanceof

Μπορεί να χρησιμοποιηθεί για να έλεγχο του είδους του συστατικού:

Π.χ.

```
public void actionPerformed(ActionEvent, e){
    Object comp = e.getSource();
    if (comp instanceof JTextField)
         calculateScore();
    else if (comp instance of JButton)
         quitProgram();
}
```

AdjustementListener

Έχει μια και μοναδική μέθοδο, την

```
public void adjustmentValueChanged(AdjustmentEvent e){
        <χειρισμόςσυμβάντος>
}
```

FocusListener

Μέθοδοι διασύνδεσης:

```
public void FocusGained(FocusEvent e){
        <χειρισμόςσυμβάντος>
}

public void FocusLost(FocusEvente ){
        <χειρισμόςσυμβάντος>
}
```

ItemListener

JButton, JCheckBox, JComboBox, JRadioButton

Όταν ένα στοιχείο επιλέγεται ή αποεπιλέγεται

Μέθοδος διασύνδεσης:

```
public  void  itemStateChanged(ItemEvent e){
<χειρισμόςσυμβάντος>
}
```

Για να καθορίσετε το στοιχείο όπου συνέβη το συμβάν: μέθοδος getItem() στο αντικείμενο ItemEvent.

KeyListener

Μέθοδοι διασύνδεσης:

```
public void keyPressed(KeyEvent e){
        <χειρισμόςσυμβάντος>
}
public void keyReleased(KeyEvent e){
        <χειρισμόςσυμβάντος>
}
public void keyTyped(KeyEvent e){
        <χειρισμόςσυμβάντος>
}
```

Χρήση μεθόδου getKeyChar() του KeyEvent

MouseListener

Μέθοδοι διασύνδεσης:

```
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
```

Χρήση μεθόδου getPoint() του MouseEvent

MouseMotionListener

Μέθοδοι διασύνδεσης:

```
public void mouseDragged(MouseEvent e)
public void mouseMoved(MouseEvent e)
```

Χρήση μεθόδων του MouseEvent

WindowListener

Μέθοδοι διασύνδεσης:

```
public void windowActivated(WindowEvent e)
public void windowClosed(WindowEvent e)
public void windowClosing(WindowEvent e)
public void windowDeactivated(WindowEvent e)
public void windowDeiconified(WindowEvent e)
public void windowIconified(WindowEvent e)
public void windowOpened(WindowEvent e)
```

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.1

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 4: JAVA: ΔΙΕΠΑΦΕΣ ΧΡΗΣΤΗ, ΓΡΑΦΙΚΑ, APPLETS

Γραφικά, Java Applets

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΓΡΑΦΙΚΑ με JAVA2D

ΧΡΗΣΗ ΓΡΑΦΙΚΩΝ

- Μέσω της Java2D:
  - Ένα σύνολο κλάσεων που υποστηρίζει υψηλής ποιότητας 2D γραφικά, εικόνες, χρώμα και κείμενο
  - Περιλαμβάνει κλάσεις από awt και swing
- Βασική κλάση: Graphics2D
  - Επέκταση της Graphics για καλύτερης ποιότητας γραφικά

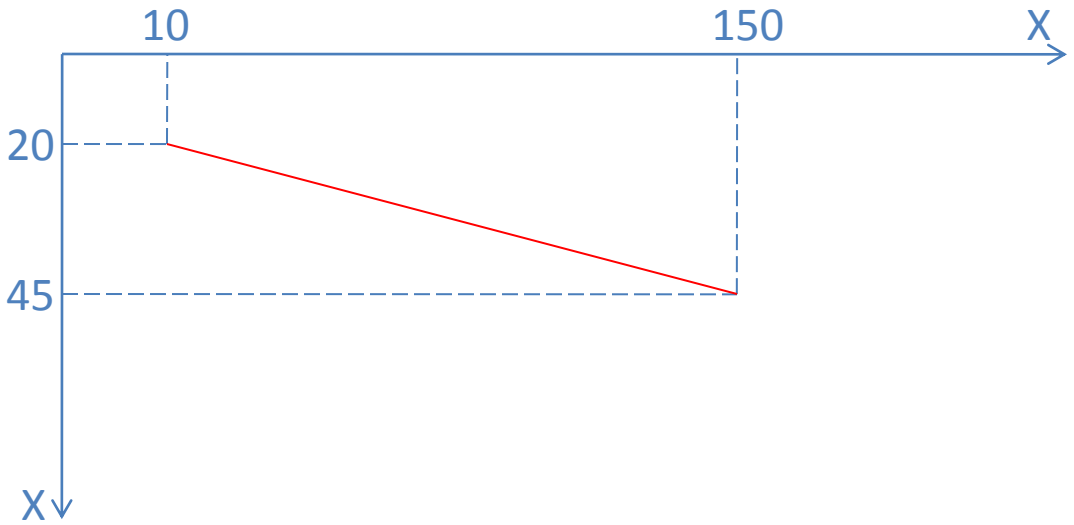
ΜΕΘΟΔΟΙ ΚΛΑΣΗΣ GRAPHICS

Σχήματα

- Ένα σχήμα μπορεί να είναι filled ή unfilled, ανάλογα με τη μέθοδο
- Οι παράμετροι των μεθόδων καθορίζουν συντεταγμένες και μεγέθη

ΜΕΘΟΔΟΙ ΚΛΑΣΗΣ GRAPHICS

- **Γραμμές**



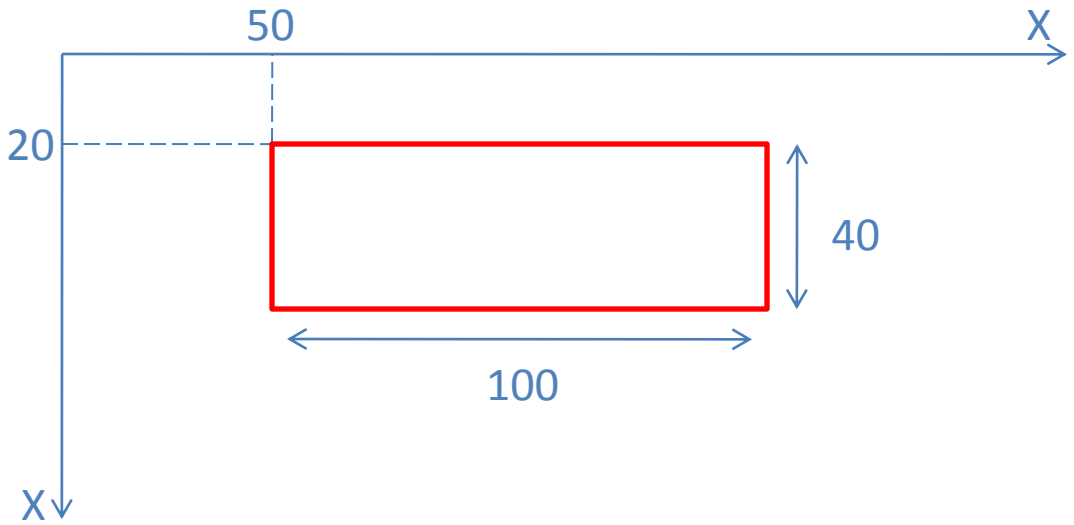

```
g.drawLine(10, 20, 150, 45);
```

ή

```
g.drawLine(150, 45, 10, 20);
```

ΜΕΘΟΔΟΙ ΚΛΑΣΗΣ GRAPHICS

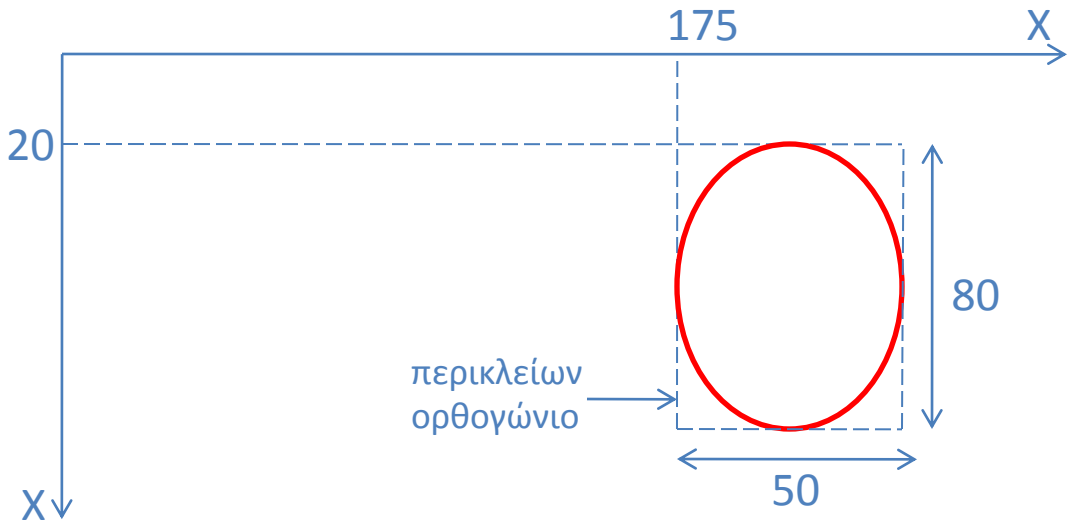
- **Ορθογώνια**




```
g.drawRect(50, 20, 100, 40);
```

ΜΕΘΟΔΟΙ ΚΛΑΣΗΣ GRAPHICS

- **Ελλείψεις**




```
g.drawOval(175, 20, 50, 80);
```

Ένα τόξο(arc) είναι ένα τμήμα μιας έλλειψης

```
g.drawArc(175, 20, 50, 80, 0, 45);
```

ΜΕΘΟΔΟΙ ΚΛΑΣΗΣ GRAPHICS

Μέθοδοι Σχεδίασης Σχημάτων (υποσύνολο):

drawLine(startX, startY, endX, endY)
drawRect(startX, startY, width, height)
fillRect(startX, startY, width, height)
drawRoundRect(startX, startY, width, height, arcWidth, arcHeight)
fillRoundRect(startX, startY, width, height, arcWidth, arcHeight)
draw3DRect(startX, startY, width, height, boolean value)
fill3DRect(startX, startY, width, height, boolean value)
drawPolygon(arrayX, arrayY, array length)
fillPolygon(arrayX, arrayY, array length)
drawPolyline(arrayX arrayX, arrayY arrayY, array length)
fillPolyline(arrayX arrayX, arrayY arrayY, array length)
drawOval(TopCornerX, TopCornerY, width, height)
fillOval(TopCornerX, TopCornerY, width, height)
drawArc(TopCornerX, TopCornerY, width, height, arcStart, arcStop)
fillArc(TopCornerX, TopCornerY, width, height, arcStart, arcStop)

Μέθοδοι αντιγραφής - καθαρισμού περιοχών:

copyArea(StartX, StartY, Width, Height, dX, dY)
clearRect(StartX, StartY, Width, Height)

ΚΛΑΣΗ COLOR

- Ένα χρώμα ορίζεται με χρήση ενός στιγμιότυπου της κλάσης Color
  - `Color myColor = new Color(200, 90, 30);`
- Κάθε drawing surface έχει ένα **background color**
- Κάθε graphics context έχει ένα τρέχον **foreground color** και **font** (type, size).
- Και τα δύο μπορούν να αλλάξουν δυναμικά
  - `setBackground(c)`
  - `g.setColor(c)` `g.setFont(f)`
- Μερικά predefined colors:

Object	**RGB Value**
Color.black	0 , 0, 0
Color.blue	0, 0, 0
Color.cyan	0, 255, 255
Color.orange	255, 255, 0
Color.white	255, 255, 255
Color.yellow	255, 255, 0

Graphics 2D

- Σχεδιασμός γραφικών σε ένα Jpanel

```
Public void paintComponent (Graphics comp){
     Graphics2D comp2D = (Graphics2D) comp;
     Font myFont = new Font("Arial", Font.ITALIC, 14);
     comp2D.setFont(myFont);
     comp2D.setColor(Color.orange);
     comp2D.drawString("Hello !!! ", 100, 50);
}
```

- Η μέθοδος paintComponent() καλείται αυτόματα κάθε φορά που ο υποδοχέας πρέπει να σχεδιαστεί ξανά
- Συνήθης μέθοδος: Δημιουργούμε μια υποκλάση της Jpanel και υπερσκελίζουμε την paintComponent().

Graphics2D
Καθορισμός πένα σχεδίασης

- Τα αντικείμενα που προσδιορίζουν σχήματα ανήκουν στο πακέτο κλάσεων java.awt.geom.
- Όλα τα σχήματα σχεδιάζονται με γραμμές που έχουν πλάτος ένα pixel
- Η αλλαγή του πάχους των γραμμών γίνεται με την δημιουργία μιας «πένας» ως εξής:

```
BasicStroke brush = new BasicStroke(5);
comp2D.setStroke(brush);
```

- Επίσης μπορούμε με την BasicStroke να καθορίσουμε το στυλ τελειώματος μια γραμμής και το στυλ της ένωσης ανάμεσα σε δυο ευθύγραμμα τμήματα

Graphics2D
Δημιουργία Αντικειμένων Σχεδίασης

- Γραμμές:

```
Line2D.Float line1 = new Line2D.Float( 40F, 200F, 70F, 130F);
comp2D.draw(line1);
```

- Ορθογώνια:

```
Rectangle2D.Float rect1 = new Rectangle2D.Float (10F,20F,50F,60F);
comp2D.draw(rect1);
```

- Ορθογώνια (με στρογγυλεμένες γωνίες):

```
Rectangle2D.Float.rect2 = new Rectangle2D.Float(10F,20F,50F,60F, 15F, 15F);
comp2D.draw(rect2);
```

- Μετά την δημιουργία του σχήματος αυτό μπορεί να σχεδιαστεί με τις μεθόδους draw() και fill().
- Η fill() γεμίζει ένα σχήμα με μοτίβα γεμίσματος ή με ένα μόνο χρώμα.

Graphics2D
Δημιουργία Αντικειμένων Σχεδίασης

- Κύκλοι ή ελλείψεις:

```
Ellipse2D..Float circle = new Ellipse2D.Float(50F, 100F, 20F, 20F);
```

- Τόξα:

```
Arc2D.Float arc = new Arc2D.Float (10F,20F,30F,30F,45F,120F, Arc2D.Float.OPEN);
```

- Η σχεδίαση πολυγώνου γίνεται ώς:

```
GeneralPath poligono = new GeneralPath(); // Δημιουργία
poligono.moveTo(10F, 20F) // Αρχικό σημείο
poligono.lineTo(20F, 10F)     //Επόμενο σημείο
poligono.lineTo(30F, 60F)     //Επόμενο σημείο
…
poligono.closePath();              //Κλείσιμο πολυγώνου
```

JAVA APPLETS

JAVA APPLETS

- Ένα **Java application** είναι ένα αυτόνομο (stand-alone) πρόγραμμα που διαθέτει μια main μέθοδο.
- Ένα **Java applet** είναι ένα πρόγραμμα που θα πρέπει να μεραφερθεί μέσω του Web σε ένα web browser και να εκτελεστεί στα πλαίσια του.
- Ένα applet μπορεί επίσης να εκτελεσθεί και με τη χρήση του appletviewer tool του Java Software Development Kit
- Ένα applet δεν έχει main μέθοδο
- Αντίθετα, υπάρχουν μια σειρά από μεθόδους που εξυπηρετούν συγκεκριμένους σκοπούς.

JAVA APPLETS

Βήματα δημιουργίας ενός Java Applet

- Γράφουμε τον πηγαίο κώδικα,
  π.χ. TestApplet.java
- Μεταφράζουμε το Applet για να παραχθεί το αρχείο TestApplet.class
- Ενσωματώνουμε το Applet σε μια ιστοσελίδα χρησιμοποιώντας (κατ' ελάχιστο) την ετικέτα <Applet> η οποία έχει την εξής (ελάχιστη) μορφή:
  <applet code= TestApplet.class width=πλάτος height=ύψος></applet>
- Ένα Applet εκτελείται όταν φορτώνεται η ιστοσελίδα που το περιέχει σε κάποιον browser

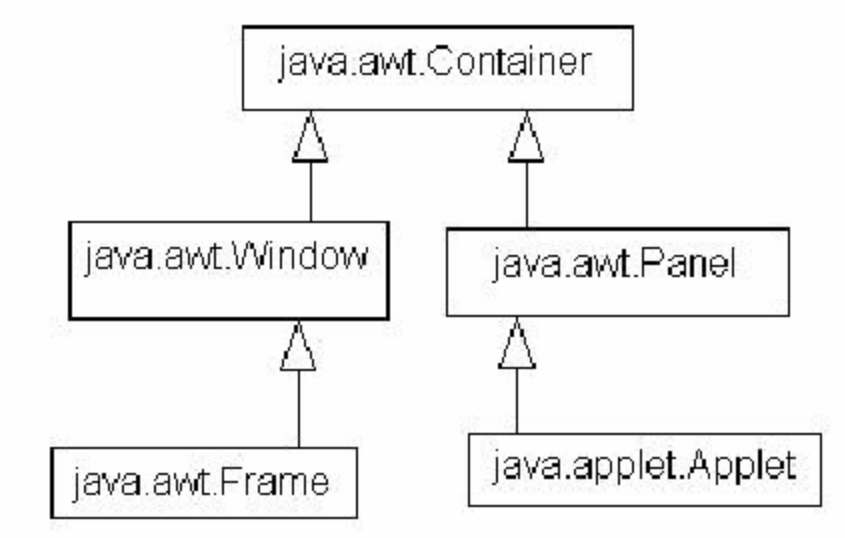
JAVA APPLETS

HTML αρχείο:

```
<HTML>
    <HEAD>
        <TITLE>Test Applet</TITLE>
    </HEAD>
    <BODY>
        <APPLET CODE= " TestApplet.class"
                WIDTH=200 HEIGHT=200>
        </APPLET>
    </BODY>
</HTML>
```

JAVA APPLETS

Υποκλάση της κλάσης
`java.applet.Applet`



JAVA APPLETS

HelloWorld:

```
import java.applet.*;
import java.awt.*;

public class HelloWorldApplet extends Applet{
    public void paint(Graphics g){
      g.drawString("Hello world!", 50, 25);
    }
}
```

ΚΥΚΛΟΣ ΖΩΗΣ ΕΝΟΣ APPLET

1. Ο browser δημιουργεί ένα στιγμιότυπο της κλάσης Applet με χρήση του default constructor
2. Το applet εμφανίζεται στην ιστοσελίδα στη θέση και με το μέγεθος που καθορίζεται στον HTML κώδικα
3. Ο browser εκτελεί την **init()** μέθοδο του applet
4. Ο browser εκτελεί την **start()** μέθοδο του applet
5. Ο browser εκτελεί την **paint()** μέθοδο του applet
6. Το applet είναι ενεργό και «τρέχει»
7. Ο browser καλεί την **paint()** κάθε φορά που το applet πρέπει να «ξαναζωγραφιστεί»
8. Ο browser εκτελεί την **stop()** μέθοδο όταν ο χρήστης φεύγει από την ιστοσελίδα ή το applet είναι έτοιμο να καταστραφεί
9. Ο browser εκτελεί την **destroy()** μέθοδο αμέσως πριν καταστρέψει το applet

ΒΑΣΙΚΕΣ ΜΕΘΟΔΟΙ APPLET

- `init()` . Αρχικοποίηση – Εκτελείται μία φορά όταν δημιουργείται το applet. Χρησιμοποιείται για αρχικοποίηση, π.χ. μεταβλητών του applet.
- `start()` . Εκκίνηση εκτέλεσης – Εκτελείται την πρώτη φορά αλλά και κάθε φορά που ο έλεγχος επανέρχεται από τον browser στην HTML σελίδα που περιέχει το applet (εκκίνηση threads, animation, playing sound, κλπ.)
- `stop()` . Τερματισμός εκτέλεσης – Πριν την καταστροφή του applet και όταν ο χρήστης εγκαταλείπει την ιστοσελίδα. Τυπικά, τερματίζεται ότι ξεκίνησε στη start().
- `destroy()` . Καταστροφή στιγμιότυπου – garbage collection.

ΒΑΣΙΚΕΣ ΜΕΘΟΔΟΙ APPLET

- **`paint(Graphics g)`** . Ζωγράφισε το applet – Εκτελείται κάθε φορά που πρέπει να ζωγραφιστεί το περιεχόμενο του applet
  - Μετά την start()
  - Όταν το παράθυρο έρχεται στο προσκήνιο
  - Όταν το ζητάει το πρόγραμμα
- Η paint() κληρονομείται από την κλάση java.awt.Container
- Η παράμετρος java.awt.Graphics αναπαριστά το τμήμα της οθόνης που το applet μπορεί να ζωγραφίσει

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.1

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων