



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

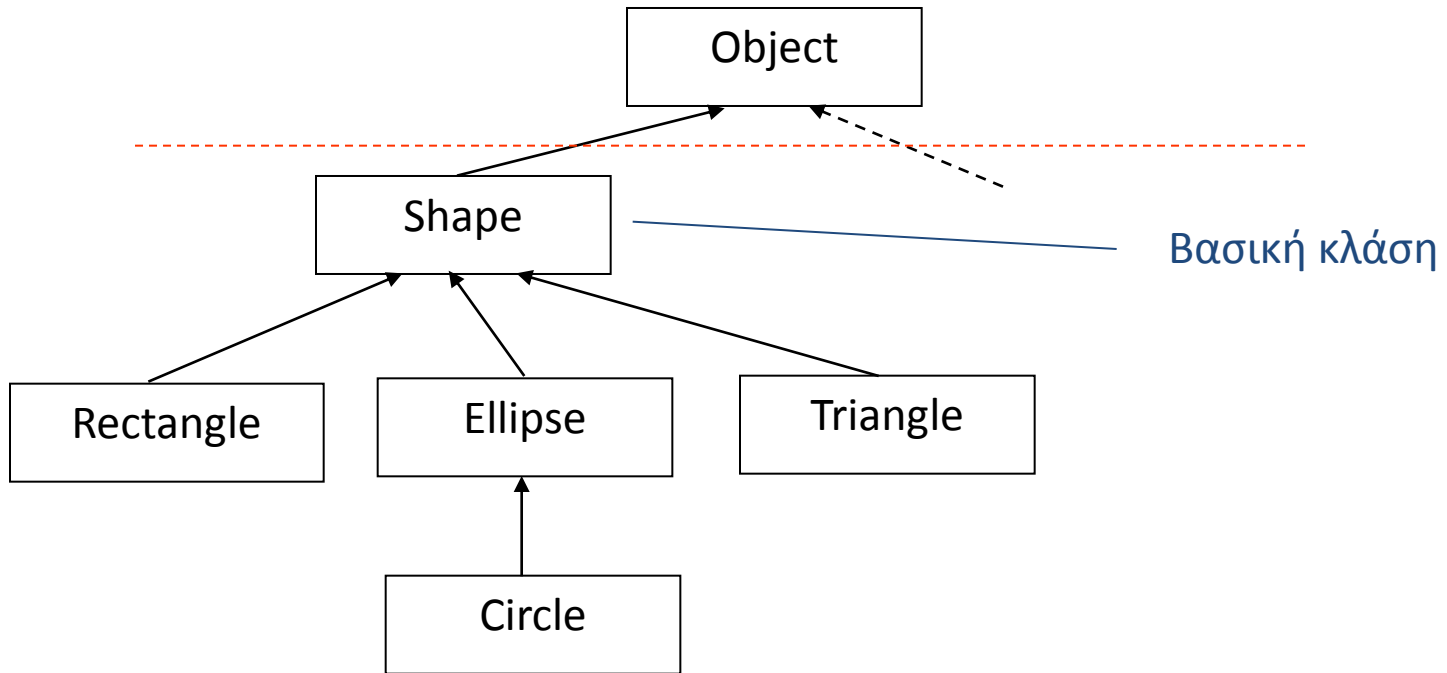


ΑΠΛΗ ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ

- 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

Rectangle

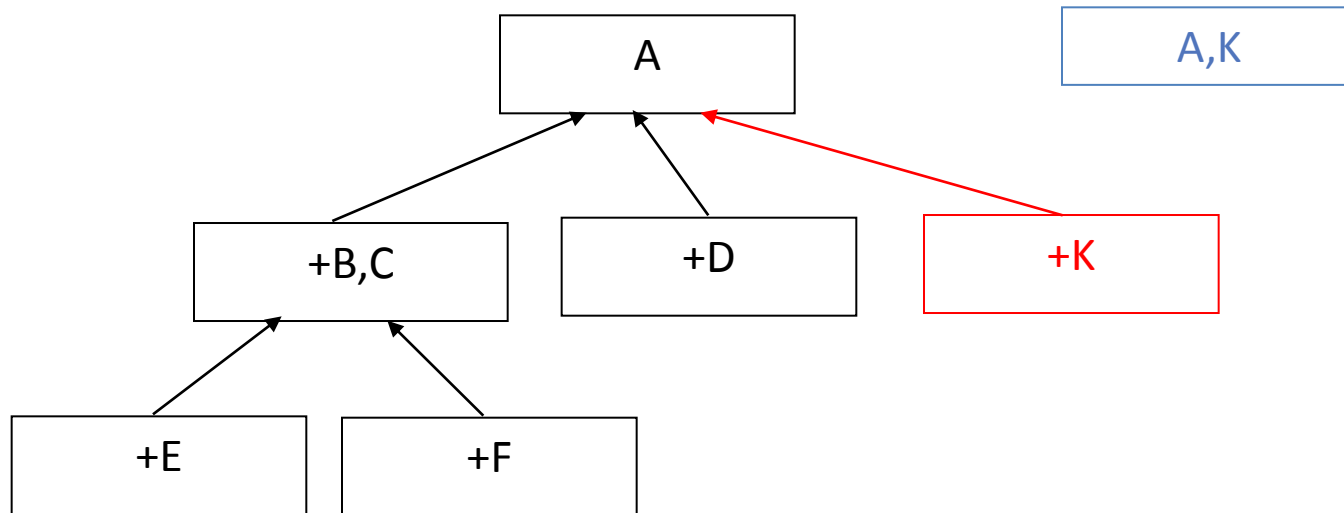
Ellipse

Triangle

Circle

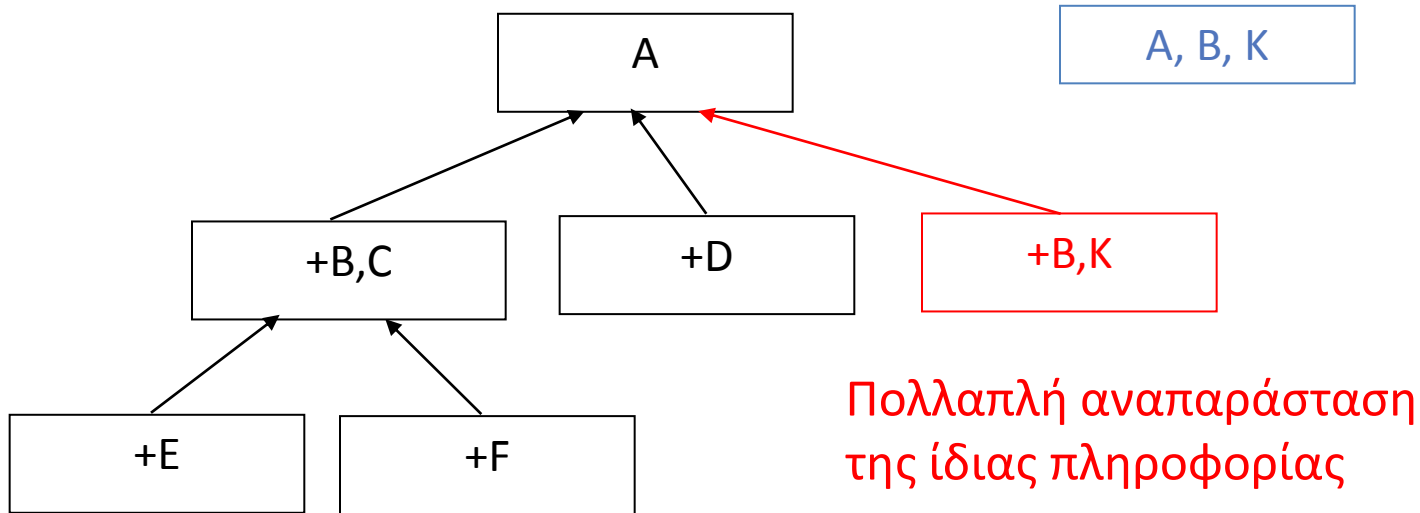


ΠΡΟΣΘΗΚΗ ΝΕΑΣ ΚΛΑΣΗΣ: ΧΩΡΙΣ ΑΝΑΔΟΜΗΣΗ ΙΕΡΑΡΧΙΑΣ



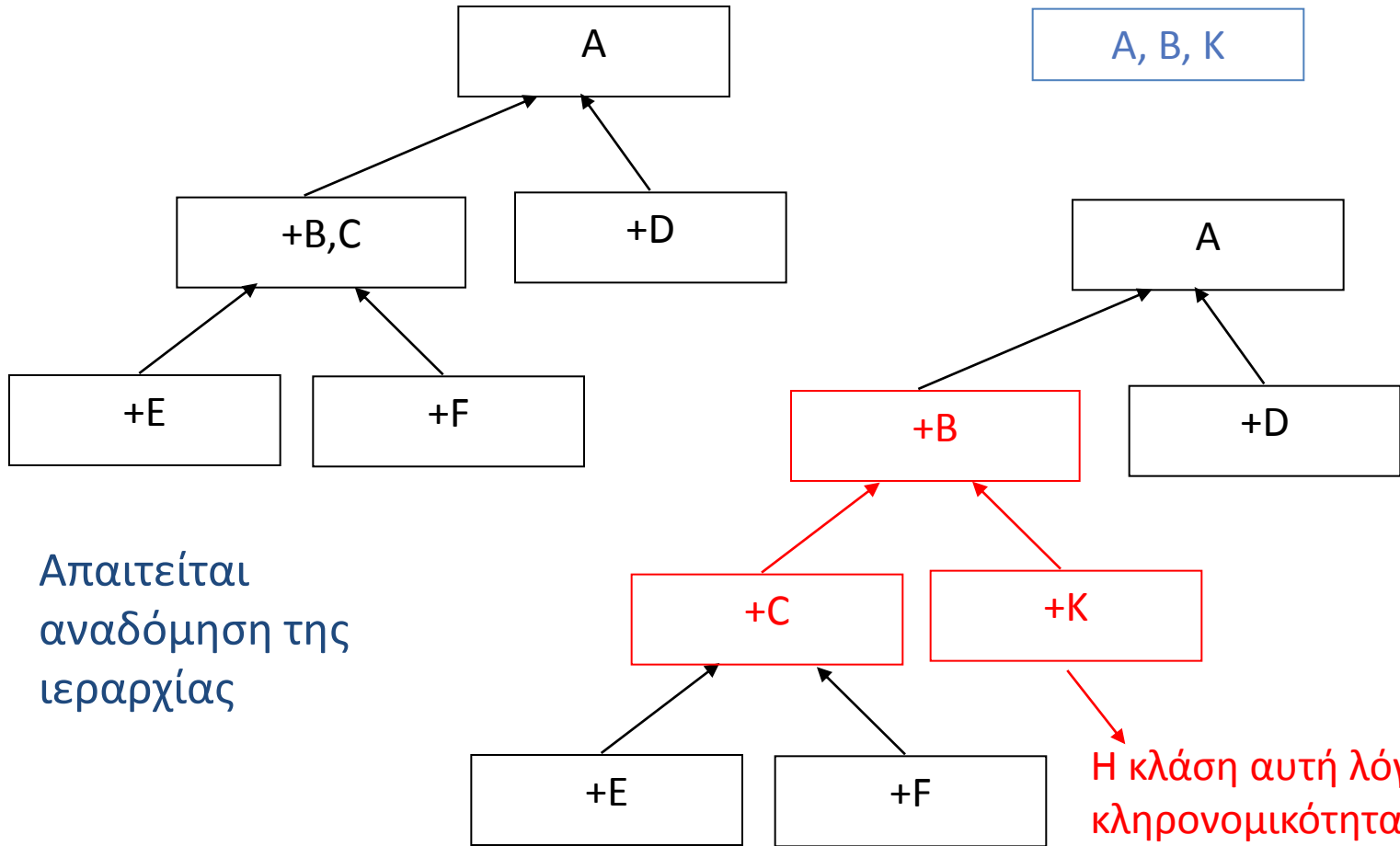
Προσθήκη μιας κλάσης με χαρακτηριστικά A, K

ΠΡΟΣΘΗΚΗ ΝΕΑΣ ΚΛΑΣΗΣ: ΜΕ ΑΝΑΔΟΜΗΣΗ ΙΕΡΑΡΧΙΑΣ (1)



Προσθήκη μιας κλάσης με χαρακτηριστικά A, B, K

ΠΡΟΣΘΗΚΗ ΝΕΑΣ ΚΛΑΣΗΣ: ΜΕ ΑΝΑΔΟΜΗΣΗ ΙΕΡΑΡΧΙΑΣ (2)



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>);
```

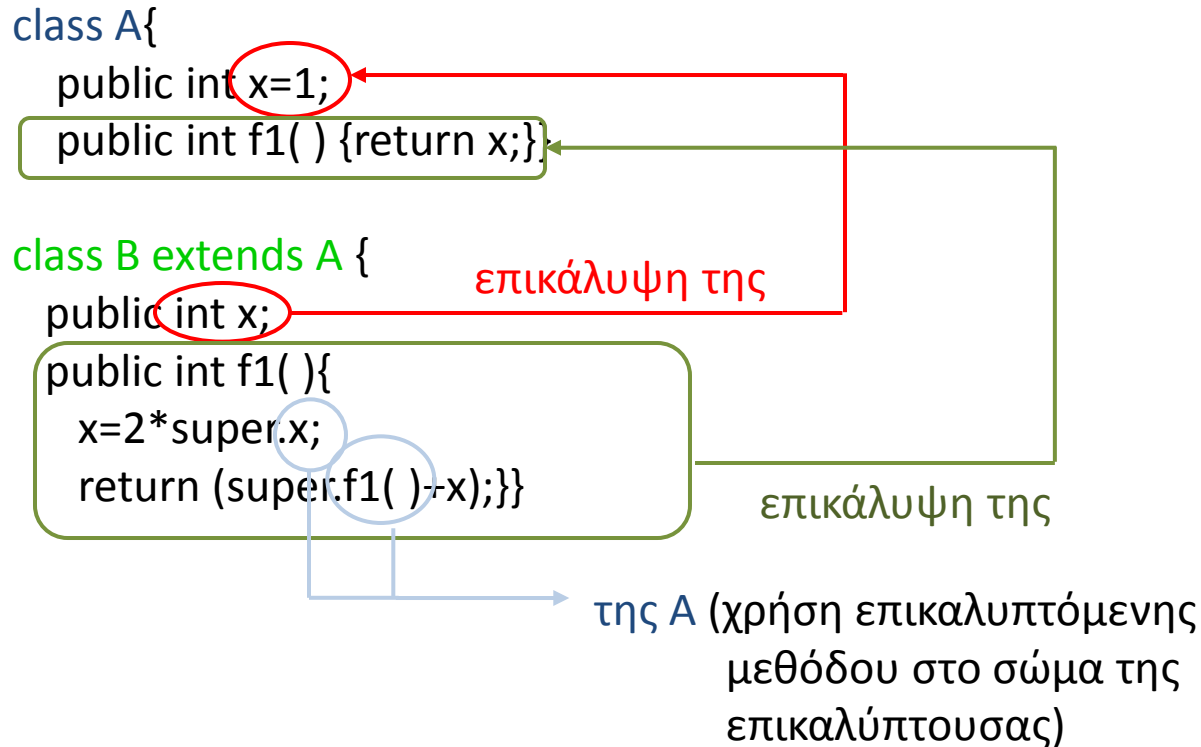
- Επίσης, μπορούμε να καλέσουμε απ' ευθείας μια μεταβλητή της υπερκλάσης:

```
super.<όνομα-μεταβλητής>;
```

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



ΠΑΡΑΔΕΙΓΜΑ



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

Copyright: Πανεπιστήμιον Πατρών, Ιωάννης Χατζηλυγερούδης, 2015.
«Οντοκεντρικός Προγραμματισμός». Έκδοση: 1.0.1 Πάτρα 2015. Διαθέσιμο
από τη δικτυακή διεύθυνση:

<https://eclass.upatras.gr/courses/CEID1105/>



Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

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

