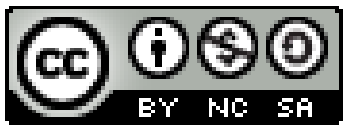
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Αντικειμενοστρεφής Προγραμματισμός

Ενότητα 11: Χειρισμός Σφαλμάτων

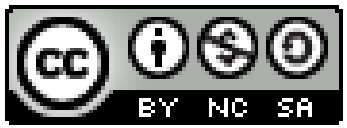
Γρηγόρης Τσουμάκας, Επικ. Καθηγητής
Τμήμα Πληροφορικής



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» 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.

Χειρισμός Σφαλμάτων

Τα παραδείγματα κώδικα που χρησιμοποιούνται σε κάποιες από τις ακόλουθες διαφάνειες μπορούν να βρεθούν στον παρακάτω σύνδεσμο:
http://users.auth.gr/greg/oop.zip

Σφάλματα

- Εσφαλμένη κατανόηση των προδιαγραφών.
  - Π.χ. υπολογισμός μέσης τιμής αντί για διάμεσο.
- Εσφαλμένη κλήση μεθόδου.
  - Π.χ. παράμετρος εκτός έγκυρου εύρους.
- Μη αναμενόμενη χρήση αντικειμένου που το οδηγεί σε ασυνεπή ή ακατάλληλη κατάσταση.
  - Π.χ. μπορεί να προέκυψε λόγω κληρονομικότητας.

Δεν υπάρχουν προγράμματα χωρίς σφάλματα!

Φταίει Πάντα ο Προγραμματιστής;

- Λάθος διεύθυνση URL σε έναν browser.
  - Ανύπαρκτη διεύθυνση.
- Δικτυακό σφάλμα σύνδεσης.
  - Διακοπή σύνδεσης.
- Αρχεία.
  - Λάθος όνομα αρχείου.
  - Γεμάτος δίσκος.
  - Ανεπαρκή δικαιώματα πρόσβασης.

Αμυντικός Προγραμματισμός

- Αλληλεπίδραση πελάτη – εξυπηρέτη.
  - Ένα αντικείμενο πελάτης (client) καλεί μεθόδους ενός άλλου αντικειμένου εξυπηρέτη (server).
- Προγραμματιστής εξυπηρέτη.
  - Να υποθέσει ότι οι πελάτες θα είναι φιλικοί ή εχθρικοί; Τι θα πρέπει να προσέξει;
- Αμυντικός προγραμματισμός.
  - Έλεγχος των παραμέτρων που στέλνει ο πελάτης κατά την δημιουργία αντικειμένων και την κλήση μεθόδων με σκοπό την αποφυγή σφαλμάτων.

Παράδειγμα (1/3)

```
public class Student {

    private int aem;
    private String name;
    private int ects;

    …

    public void addEcts(int x) {
        ects += x;
    }

}
```

Παράδειγμα (2/3)

```
public class Department1 {

    Map<Integer,Student> students;

    public Department1() {
        students = new HashMap<>();
    }

    …

    public void addEcts(int aem, int ects) {

        students.get(aem).addEcts(ects);

    }

}
```

Υποθέτει ότι ο πελάτης είναι φιλικός

Παράδειγμα (3/3)

```
public class Department2 {

    Map<Integer,Student> students;

    public Department2() {
        students = new HashMap<>();
    }

    …

    public void addEcts(int aem, int ects) {
        if (students.containsKey(aem)) {

            students.get(aem).addEcts(ects);

        }
    }
}
```

Αμυντικός προγραμματισμός

Αναφορά των Σφαλμάτων

- Ειδοποίηση του χρήστη.
  - Πολλές εφαρμογές εκτελούνται χωρίς επίβλεψη από κάποιο χρήστη.
  - Ακόμα και αν υπάρχει χρήστης, αυτός συνήθως δεν μπορεί να κάνει κάτι για το σφάλμα.
- Ειδοποίηση του αντικειμένου πελάτη.
  - Επιστροφή τιμής από την κλήση της μεθόδου.
  - Παραγωγή μιας εξαίρεσης.

Επιστροφή Τιμής (1/2)

- Αν η μέθοδος ήταν *void.*
  - Επιστροφή λογικής τιμής που φανερώνει αν πήγαν όλα καλά .
- Αν επιστρέφει αντικείμενο.
  - Επιστροφή τιμής *null* για να επισημάνουμε πρόβλημα.
- Αν επιστρέφει αριθμό.
  - Επιστροφή αριθμού εκτός αναμενόμενων ορίων.

Παράδειγμα

```
public class Department3 {
  Map<Integer,Student> students;
  public Department3() {
    students = new HashMap<>();
  }
  …
  public boolean addEcts(int aem, int ects) {
    if (students.containsKey(aem)) {
      students.get(aem).addEcts(ects);
      return true;
    } else
      return false;
  }
}
```

Επιστροφή Τιμής (2/2)

- Πως θα ενημερώσουμε τον πελάτη για εσφαλμένες παραμέτρους σε κατασκευαστή;
  - Δεν γίνεται με επιστροφή τιμής.
- Θα γίνει διαχείριση του σφάλματος;
  - Δεν αναγκάζεται ο προγραμματιστής να ελέγξει την επιστρεφόμενη τιμή.

Εξαιρέσεις και Ρίψη Εξαιρέσεων

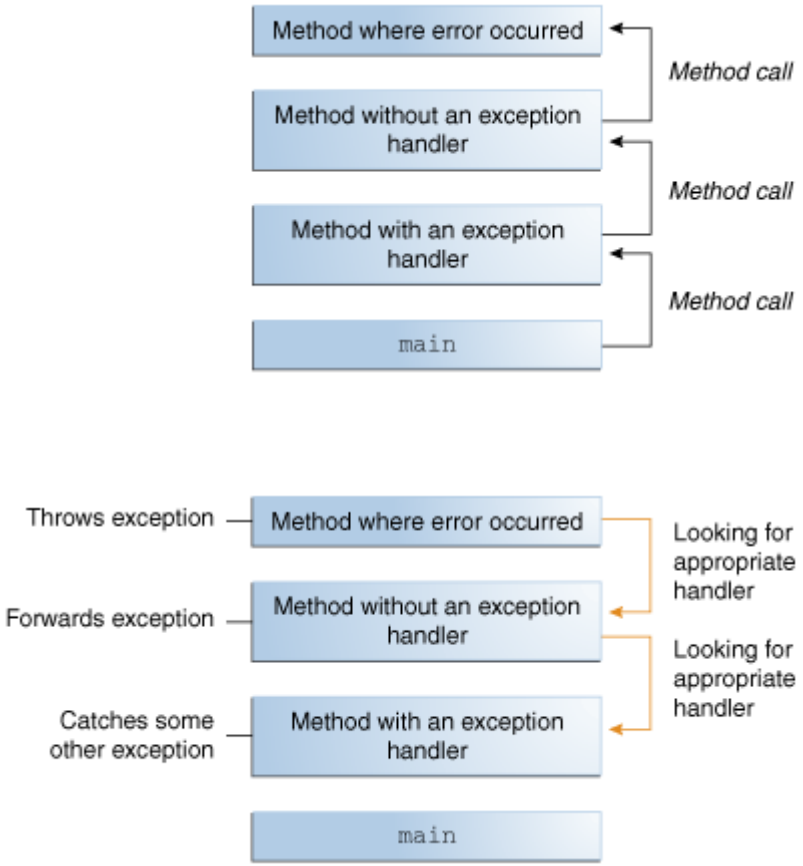
- Εξαίρεση (exception).
  - Σφάλμα κατά την εκτέλεση ενός προγράμματος που διακόπτει τη φυσιολογική ροή των εντολών.
- Ρίψη εξαίρεσης (throwing an exception).
  - Όταν συμβεί ένα σφάλμα μέσα σε μία μέθοδο, τότε αυτή δημιουργεί ένα αντικείμενο εξαίρεσης που χαρακτηρίζει αυτό το σφάλμα, και στη συνέχεια το παραδίδει στο σύστημα χρόνου εκτέλεσης (runtime system) με την εντολή *throw.*

Παράδειγμα

```
public class Department4 {
   Map<Integer,Student> students;
   public Department4() {
      students = new HashMap<>();
   }
   …
   public void addEcts(int aem, int ects) {
      if (students.containsKey(aem)) {
         students.get(aem).addEcts(ects);
      } else
         throw new IllegalArgumentException();
   }
}
```

Αναζήτηση Χειριστή

Το σύστημα χρόνου εκτέλεσης αναζητεί στη στοίβα κλήσεων (call stack) να βρει αν ο κώδικας που οδήγησε στην εξαίρεση βρίσκεται μέσα σε ένα block κώδικα που παρακολουθείται για εξαιρέσεις και αν αυτό συνοδεύεται από κατάλληλο χειριστή (exception handler) για τη συγκεκριμένη εξαίρεση.



Παρακολούθηση & Χειρισμός

```
try {
    // κώδικας που παρακολουθείται
    // για εμφάνιση σφαλμάτων
}
catch (ExceptionType exception) {
  // χειριστής για εξαίρεση ExceptionType
}
```

Τα δύο αυτά τμήματα πάντα συνυπάρχουν ως ζεύγος

Καταλληλότητα Χειριστή (1/2)

- Κατάλληλος χειριστής εξαίρεσης.
  - Ο τύπος της εξαίρεσης ταιριάζει (είναι ίδιος ή υποκλάση) με τον τύπο στον χειριστή.
- Αν βρεθεί κατάλληλος χειριστής.
  - Τότε λέμε ότι συλλαμβάνει (catches) την εξαίρεση.
  - Εκτελείται ο κώδικας του και η ροή της εκτέλεσης συνεχίζει μετά το πέρας του try/catch.

Καταλληλότητα Χειριστή (2/2)

- Κατάλληλος χειριστής εξαίρεσης.
  - Ο τύπος της εξαίρεσης ταιριάζει (είναι ίδιος ή υποκλάση) με τον τύπο στον χειριστή.
- Αν δε βρεθεί (κατάλληλος) χειριστής,
  - Η εξαίρεση θα συλληφθεί από την JVM, με αποτέλεσμα να σταματήσει η εκτέλεση του προγράμματος και να εμφανιστεί το ίχνος της στοίβας κλήσεων και ένα μήνυμα με πληροφορίες για το σφάλμα.

Ορισμένα Πρώτα Παραδείγματα

- Μη εύρεση χειριστή.
- Εύρεση ακατάλληλου χειριστή.
- Εύρεση κατάλληλου χειριστή.
  - Εκεί που προκλήθηκε η εξαίρεση.
  - Σε προηγούμενη κλήση (προώθηση εξαίρεσης).

Example1?.java, Example2?.java

Είδη Εξαιρέσεων (1/4)

- Η κλάση *Throwable.*
  - Ένα αντικείμενο εξαίρεσης (πρέπει να) ανήκει σε κλάση που κληρονομεί την κλάση *Throwable.*
  - Έχει δύο παιδιά, τις κλάσεις *Error* και *Exception.*
- Εξαιρέσεις της κλάσης *Exception.*
  - Οφείλονται στην εφαρμογή που εκτελείται.
  - Διακρίνονται σε (α) μη αναμενόμενα λογικά σφάλματα, και (β) σε σφάλματα που ενδέχεται να προκύψουν.

Είδη Εξαιρέσεων (2/4)

- Μη αναμενόμενα λογικά σφάλματα.
  - Διόρθωση του κώδικα ώστε να μην προκύπτουν.
  - Συνήθως δεν μπορούμε να διορθώσουμε το λάθος κατά την εκτέλεση και να ανακάμψει η εφαρμογή.
  - Η Java παρέχει πολλές τέτοιες εξαιρέσεις, όπως υπέρβαση ορίων πίνακα, ακέραια διαίρεση με 0, …
  - Εξαιρέσεις αυτού του είδους ανήκουν στην υποκλάση *RuntimeException* της κλάσης *Exception.*

Είδη Εξαιρέσεων (3/4)

- Σφάλματα που ενδέχεται να προκύψουν.
  - Π.χ. ο χρήστης δίνει ένα όνομα αρχείου για άνοιγμα, το οποίο δεν υπάρχει στο σύστημα.
  - Υπάρχει πάντα τρόπος να διορθώσουμε το λάθος κατά την εκτέλεση και να ανακάμψει η εφαρμογή.
  - Μια καλοσχεδιασμένη εφαρμογή οφείλει να έχει χειριστές και να ανακάμπτει από τέτοια σφάλματα.

Είδη Εξαιρέσεων (4/4)

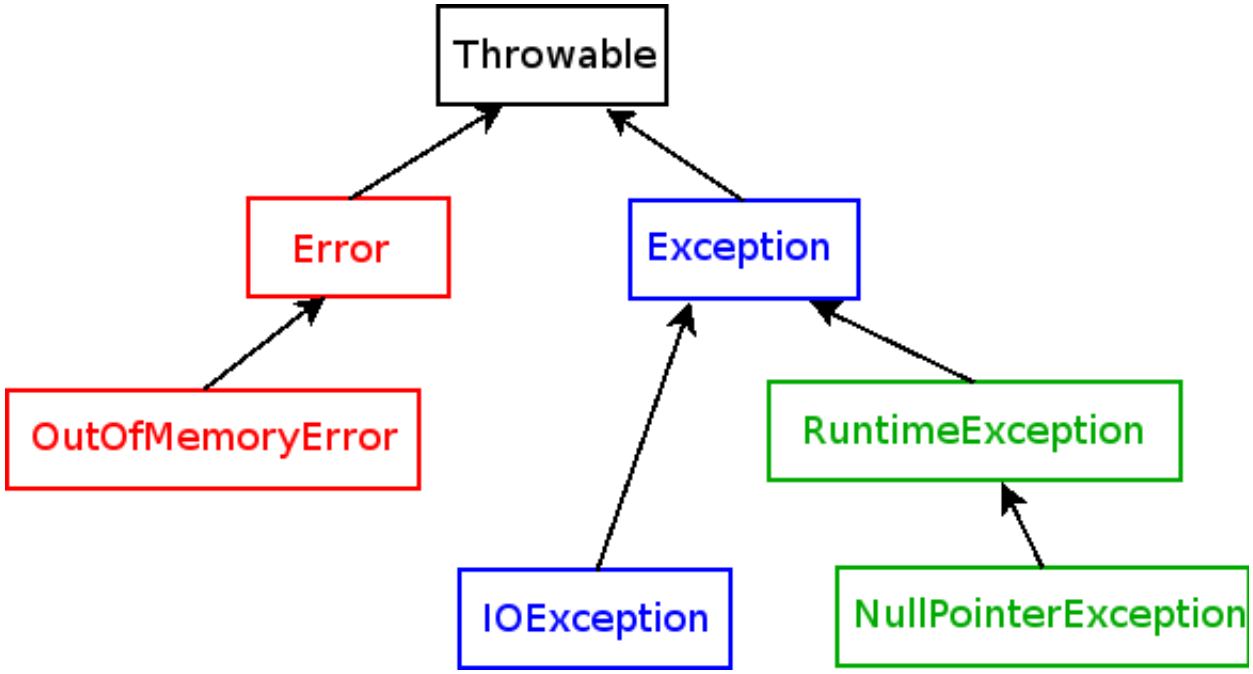
- Εξαιρέσεις της κλάσης *Error.*
  - Δεν οφείλονται στην εφαρμογή που εκτελείται.
  - Συνήθως είναι μη αναμενόμενες και χωρίς δυνατότητα ανάκαμψης (διόρθωσης του σφάλματος).
    - Π.χ. αδυναμία ανάγνωσης ενός αρχείου που υπάρχει στο σύστημα λόγω προβλήματος στο υλικό.
  - Αν προσδιορίσουμε χειριστή, αποφεύγουμε το κρέμασμα της εφαρμογής.

Υποχρέωση Χειρισμού

- Μη ελεγκτέες εξαιρέσεις (unchecked).
  - Δεν απαιτείται ο χειρισμός των εξαιρέσεων που είναι τύπου *Error* και *RuntimeException.*
  - Δεν θα έπρεπε να προκύψουν και δεν μπορούμε πάντα να κάνουμε κάτι για να τις διορθώσουμε.
- Ελεγκτέες εξαιρέσεις (checked).
  - Απαιτείται χειρισμός όσων είναι τύπου *Exception.*
  - Είναι αναμενόμενες, υπάρχει πάντα τρόπος ανάκαμψης και υποχρεώνουμε τον προγραμματιστή να τις λάβει υπόψη.

Σύνοψη Ειδών Εξαιρέσεων

- Μη ελεγκτέες: *Error* και *RuntimeException.*
- Ελεγκτέες: *Exception.*



Παραδείγματα

- Παράδειγμα *RuntimeException.*
  - Χειρισμός τους και διόρθωση σφαλμάτων για αποφυγή χειρισμού τους.

Example3?.java

- Παραδείγματα *Exception.*
  - Υποχρέωση χειρισμού τους.
  - Χειρισμός τους στο σημείο της εξαίρεσης.
  - Ο χειρισμός σε προηγούμενη κλήση απαιτεί τη χρήση της λέξης κλειδί *throws* (μη αυτόματη προώθηση) στην υπογραφή της μεθόδου.

Example4?.java

Σύλληψη Εξαιρέσεων

- Πολλαπλοί χειριστές.
  - Ένας χειριστής εξαιρέσεων μπορεί να αποτελείται από πολλαπλά catch, τα οποία εξετάζονται με τη σειρά που γράφονται μέσα στο πρόγραμμα.
- Σύλληψη μέσω υπερκλάσης.
  - Ένας χειριστής εξαιρέσεων κάποιου τύπου, μπορεί να συλλάβει οποιαδήποτε εξαίρεση είτε αυτού του τύπου είτε των υποκλάσεων του.

Example5.java

Example6.java

Γιατί Σύλληψη μέσω Υπερκλάσης;

- Αποφυγή τερματισμού της εφαρμογής.
  - Με έναν χειριστή εξαίρεσης τύπου *Throwable* θα αποφύγουμε τον τερματισμό της εφαρμογής.
  - Θα συλλάβει κάθε εξαίρεση.
- Διαφορετικές εξαιρέσεις, ίδια αντιμετώπιση.
  - Όταν ο ίδιος κώδικας μπορεί να χειριστεί ένα σύνολο διαφορετικών εξαιρέσεων, κάνουμε χρήση ενός διαχειριστή της υπερκλάσης τους.
  - Αποφυγή διπλότυπου κώδικα.

Τελικό Τμήμα Κώδικα

- Finally..
  - Ένα block κώδικα finally μπαίνει στο τέλος ενός try/catch block και εκτελείται πάντα μετά την ολοκλήρωση του try/catch.
  - Είτε προκύψει εξαίρεση, είτε όχι, ακόμη και αν ζητηθεί επιστροφή από τη μέθοδο με *return.*
- Χρησιμεύει για ρύθμιση εκκρεμοτήτων.
  - Π.χ. ανοικτό αρχείο, ανοικτή διαδικτυακή σύνδεση.

Example7.java

Δημιουργία Εξαίρεσης

- Δημιουργία εξαίρεσης.
  - Μπορείτε να δημιουργήσετε τις δικές σας νέες εξαιρέσεις επεκτείνοντας την κατάλληλη κλάση.

NonIntDivException.java, Example8.java

Πλεονεκτήματα Εξαιρέσεων

- Διαχωρισμός του κώδικα χειρισμού σφαλμάτων από τον φυσιολογικό κώδικα.
- Προώθηση σφαλμάτων στη στοίβα κλήσεων.
- Ομαδοποίηση και διαχωρισμός ειδών σφαλμάτων.

Διαχωρισμός Κώδικα (1/3)

- Έστω ο παρακάτω κώδικας για την ανάγνωση ενός αρχείου.

```
readFile {
   open the file;
   read the file into memory;
   close the file;
}
```

Διαχωρισμός Κώδικα (2/3)

```
errorCodeType readFile {
  initialize errorCode = 0;
  open the file;
  if (theFileIsOpen) {
    read the file into memory;
    if (readFailed) { errorCode = -1; }
    close the file;
    if (theFileDidntClose && errorCode == 0) {
      errorCode = -4;
    } else { errorCode = -3; }
  } else { errorCode = -5; }
  return errorCode;
}
```

Διαχωρισμός Κώδικα (3/3)

```
readFile {
  try {
    open the file;
    read the file into memory;
    close the file;
  } catch (fileOpenFailed) {
    doSomething;
  } catch (readFailed) {
    doSomething;
  } catch (fileCloseFailed) {
    doSomething;
  }
}
```

Προώθηση Σφαλμάτων (1/4)

- Στον παρακάτω κώδικα μόνο η *method1* ενδιαφέρεται για σφάλματα στη readFile.

```
method1 {
   call method2;
}
method2 {
   call method3;
}
method3 {
   call readFile;
}
```

Προώθηση Σφαλμάτων (2/4)

```
errorCodeType method3 {
    errorCodeType error;
    error = call readFile;
    if (error) return error;
}

errorCodeType method2 {
    errorCodeType error;
    error = call method3;
    if (error) return error;
}
```

Προώθηση Σφαλμάτων (3/4)

```
method1 {
    errorCodeType error;
    error = call method2;
    if (error)
        doErrorProcessing;
}
```

Προώθηση Σφαλμάτων (4/4)

```
method1 {
     try {
          call method2;
     } catch (exception e) {
          doErrorProcessing;
     }
}

method2 throws exception { call method3;  }

method3 throws exception { call readFile; }
```

Ομαδοποίηση και Διαχωρισμός (1/2)

- Οι εξαιρέσεις είναι αντικείμενα, τα οποία ανήκουν σε μια ιεραρχία κλάσεων με πρόγονο κλάση την *Throwable.*
- Αυτό μας δίνει τη δυνατότητα να χειριστούμε ομαδικά κάποια σφάλματα με βάση την υπερκλάση τους.

Ομαδοποίηση και Διαχωρισμός (2/2)

```
catch (EOFException e) {
  // Διάβασμα πέρα από το τέλος του αρχείου
}
catch (FileNotFoundException e) {
  // Δεν βρέθηκε το αρχείο
}
catch (IOException e) {
  // Γενική διαχείριση σφαλμάτων I/O
}
catch (Throwable e) {
  // Συλλαμβάνει όλα τα λάθη
}
```

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Τέλος Ενότητας

Επεξεργασία: Εμμανουήλ Ρήγας

Θεσσαλονίκη, Εαρινό Εξάμηνο 2013-2014

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης