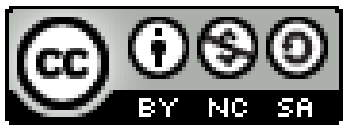
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Αντικειμενοστρεφής Προγραμματισμός

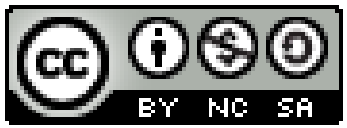
Ενότητα 10: Έλεγχος και Αποσφαλμάτωση

Γρηγόρης Τσουμάκας, Επικ. Καθηγητής
Τμήμα Πληροφορικής



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.

Έλεγχος και Αποσφαλμάτωση

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

Ποιο το Αποτέλεσμα;

```
public static void main(String[] args)
{
   int sum = 1;
   for (int i = 0; i <= 4; i++);
   {
      sum = sum + 1;
   }
   System.out.println("Result: " + sum);
   System.out.println("Double: " + sum + sum);
}
```

Quiz.java

Ουδείς Άσφαλτος

- Συντακτικά σφάλματα.
  - Τα εντοπίζει ο μεταγλωττιστής.
- Λογικά σφάλματα (γνωστά και ως bugs).
  - Δεν μπορεί να βοηθήσει ο μεταγλωττιστής.
  - Ορισμένα λογικά σφάλματα δεν εμφανίζουν προφανή παρατηρήσιμα συμπτώματα.
  - Τα εμπορικά λογισμικά σπανίως είναι 100% ορθά.
- Μείωση πιθανότητας σφαλμάτων;
  - Χρήση αρχών ορθής σχεδίασης (π.χ. ενθυλάκωση).

Έλεγχος και Αποσφαλμάτωση

- ‘Ελεγχος (testing).
  - Διερεύνηση σφαλμάτων σε ένα τμήμα κώδικα.
- Αποσφαλμάτωση (debugging).
  - Εύρεση της αιτίας ενός σφάλματος, η οποία μπορεί να είναι σε αρκετή «απόσταση» μακριά από το σημείο εμφάνισης του σφάλματος.
  - Διόρθωση του σφάλματος.
- Σημαντικές δεξιότητες τεχνολογίας λογισμικού.
  - Δικός σας κώδικας, τρίτος κώδικας.

Καλές Πρακτικές Ελέγχου

- Έλεγχος κατά την ανάπτυξη της εφαρμογής.
  - Σταδιακή προσθήκη επιπλέον ελέγχων.
  - Ενημέρωση προηγούμενων ελέγχων.
  - Όλοι οι έλεγχοι πρέπει να εκτελούνται ξανά έπειτα από κάθε βήμα ανάπτυξης της εφαρμογής.
- Πλεονέκτημα.
  - Εύρεση προβλημάτων νωρίς => μικρότερο χρονικό κόστος διόρθωσης τους.
- Πως και τι ελέγχουμε στην πράξη;

Έλεγχος Μονάδας (Unit Testing)

- Μονάδα κώδικα = μέθοδος, κλάση, πακέτο.
  - Κατανόηση της αναμενόμενης συμπεριφοράς της μονάδας (το συμβόλαιο της).
  - Εξέταση περιπτώσεων παράβασης του συμβολαίου.
- Θετικά και αρνητικά τεστ.
  - Κάνει αυτά που πρέπει.
  - Δεν κάνει αυτά που δεν πρέπει.
- Οριακά τεστ.
  - Κενή συλλογή, ένα στοιχείο, γεμάτος πίνακας.

Κλάσεις Ελέγχου

- Κατασκευή κλάσης ελέγχου.
  - Μια κλάση ελέγχου για κάθε κλάση της εφαρμογής.
- Κατασκευή μεθόδων για περιπτώσεις ελέγχου.
  - Δημιουργία αντικειμένων.
  - Κλήση μεθόδων.
  - Εξέταση της κατάστασης των αντικειμένων.
- Απαιτείται φαντασία και δημιουργικότητα.
  - Δημιουργία χρήσιμων περιπτώσεων ελέγχου.

Παράδειγμα (1/6)

- Ηλεκτρονικό κατάστημα.
  - Σε αρχικό στάδιο ανάπτυξης.
- Προϊόντα.
  - Περιγραφή, τιμή, λίστα με σχόλια.
  - Επιτρέπεται ένα μόνο σχόλιο από κάθε χρήστη.
- Σχόλια .
  - Χρήστης, κείμενο, βαθμολογία, άθροισμα θετικών και αρνητικών ψήφων.
  - Βαθμολογία [1..5].

Παράδειγμα (2/6)

```
public class Comment {
  private String author;
  private String text;
  private int rating;
  private int votes;

  public Comment(String a, String t, int r) { … }
  public void upvote() { votes++; }
  public void downvote() { votes--; }
  public String getAuthor() { return author; }
  public int getRating() { return rating; }
  public int getVoteCount() { return votes; }
  public String getFullDetails() { … }
}
```

Παράδειγμα (3/6)

```
public class SalesItem {
  private String name;
  private int price;
  private ArrayList<Comment> comments;

  public SalesItem(String name, int price) { … }
  public boolean addComment(String a, String t, int r) {
    …
  }
  public void removeComment(int index) { … }
  public void upvoteComment(int index) { … }
  public void downvoteComment(int index) { … }
  public Comment findMostHelpfulComment() { … }
  private boolean ratingInvalid(int rating) { … }
  private String priceString(int price) { … }
}
```

Παράδειγμα (4/6)

```
private boolean ratingInvalid(int rating) {
  return rating < 0 || rating > 5;
}

-----------------------------------------------
public Comment findMostHelpfulComment() {
  Iterator<Comment> it = comments.iterator();
  Comment best = it.next();
  while(it.hasNext()) {
    Comment current = it.next();
    if (current.getVoteCount() > best.getVoteCount())
      best = current;
  }
  return best;
}
```

Παράδειγμα (5/6)

```
public class SalesItemTest {


  public void testRatingInvalid {
    SalesItem i = new SalesItem("x",10.00);
    if (i.ratingInvalid(0) && i.ratingInvalid(6) &&
        !i.ratingInvalid(1) && !i.ratingInvalid(5))
       System.out.println("ok");
    else
       System.out.println("σφάλμα");
  }

  …
```

Παράδειγμα (6/6)

```
...
public void testFindMostHelpfulComment {
  SalesItem i = new SalesItem("x",10.00);

  Comment comment = i.findMostHelpfulComment();
  if (comment == null)
    System.out.println("ok");
  else
    System.out.println("σφάλμα");
}

public static void main(String[] args) {
  SalesItemTest test = new SalesItemTest();
  test.testRatingInvalid();
  test.testFindMostHelpfulComment();
}

}
```

Το Πλαίσιο Ελέγχου JUnit (1/2)

- JUnit.
  - Ένα δημοφιλές πλαίσιο ελέγχου κώδικα Java.
- @Test.
  - Δηλώνει μια μέθοδο ως περίπτωση ελέγχου.
- assertEquals(Object, Object).
  - Έλεγχος αναμενόμενης συμπεριφοράς μεθόδων.
- fail(String).
  - Σήμανση αποτυχίας του ελέγχου.

Το Πλαίσιο Ελέγχου JUnit (2/2)

- @Before public void setUp().
  - Εκτελείται πριν από κάθε περίπτωση ελέγχου.
  - Π.χ. δημιουργία αντικειμένου της κλάσης, άνοιγμα κάποιου ρεύματος.
- @After public void tearDown().
  - Εκτελείται έπειτα από κάθε περίπτωση ελέγχου.
  - Π.χ. κλείσιμο κάποιου ρεύματος.

Παράδειγμα (1/2)

```
@Test
public void testFindMostHelpfulComment() {
  System.out.println("findMostHelpfulComment");
  SalesItem instance = new SalesItem("wine", 10);
  Comment expResult = null;
  Comment res = instance.findMostHelpfulComment();
  assertEquals(expResult, res);
}

@Test
public void testAddComment() {
  System.out.println("addComment");
  SalesItem instance = new SalesItem("wine", 10);
  boolean result = instance.addComment("greg","ok",0);
  boolean expResult = false;
  assertEquals(expResult, result);
}
```

Παράδειγμα (2/2)

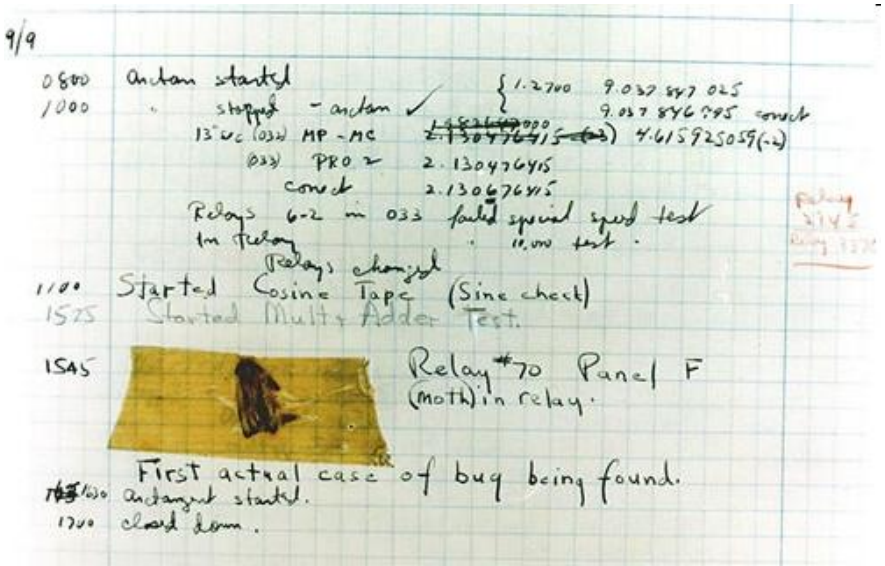
```
private SalesItem instance;

@Before
public void setUp() {
  instance = new SalesItem("wine", 10);
}

@Test
public void testFindMostHelpfulComment() {
  Comment res = instance.findMostHelpfulComment();
  assertEquals(null, res);
}

@Test
public void testAddComment() {
  boolean result = instance.addComment("greg","ok",0);
  assertEquals(false, result);
}
```

Αποσφαλμάτωση

- Ανάπτυξη δεξιότητας ανίχνευσης σφαλμάτων.
  - Όσο καλύτερος προγραμματιστής => τόσο χειρότερος ανιχνευτής σφαλμάτων.
  - Όσο χειρότερος προγραμματιστής => τόσο καλύτερος ανιχνευτής σφαλμάτων.
- Ανάπτυξη δεξιότητας ανάγνωσης κώδικα.
  - Συχνά η αποσφαλμάτωση θα εφαρμόζεται σε κώδικα που δεν έχετε γράψει εσείς.
  - Διευκόλυνση ανάγνωσης κώδικα με συγγραφή καλής τεκμηρίωσης και δομοστοιχειοποίηση.

Τεχνικές Αποσφαλμάτωσης

- Χειροκίνητη εκτέλεση.
- Εντολές εμφάνισης στην οθόνη.
- Χρήση αποσφαλματωτή (debugger).



Χειροκίνητη Εκτέλεση (1/2)

- Δεν απαιτεί εξεζητημένη τεχνολογία.
  - Απομακρυνθείτε από τον υπολογιστή.
  - Τρέξτε ένα πρόγραμμα με το χέρι/μυαλό σας.
- Επίπεδο εξέτασης του κώδικα.
  - Υψηλό (βασικές λειτουργίες).
  - Χαμηλό (εξειδικευμένες μέθοδοι).
- Παρατηρήσεις.
  - Δεν χρησιμοποιείται συχνά.
  - Πιο ισχυρή απ’ ότι φαίνεται.

Χειροκίνητη Εκτέλεση (2/2)

- Η συμπεριφορά ενός αντικειμένου εξαρτάται σημαντικά από την κατάστασή του.
  - Εσφαλμένη συμπεριφορά είναι συνήθως αποτέλεσμα εσφαλμένης κατάστασης.
- Πίνακας κατάστασης αντικειμένου.
  - Φτιάξτε ένα πίνακα με τιμές για τα σημαντικότερα πεδία.
  - Καταγράψτε τις αλλαγές κατάστασης έπειτα από κάθε κλήση μεθόδων του αντικειμένου.

calculator

Προφορικές Εκτελέσεις

- Εξηγήστε σε κάποιον τι κάνει ο κώδικας σας.
  - Μπορεί να εντοπίσουν το σφάλμα.
  - Η διαδικασία της εξήγησης θα σας κάνει να το εντοπίσετε οι ίδιοι.

Εντολές Εμφάνισης στην Οθόνη

- Η πιο διαδεδομένη τεχνική.
- Δεν απαιτούνται ιδιαίτερα εργαλεία, αφού την υποστηρίζουν όλες οι γλώσσες προγραμματισμού.
- Αποτελεσματική μόνο αν οι εντολές προστεθούν στις κατάλληλες μεθόδους.
- Πιθανή ύπαρξη μεγάλου όγκου εξόδου.
- (Απ)ενεργοποίηση θέλει καλό σχεδιασμό.

Παράδειγμα

```
public class Debug {
    public static final boolean DEBUG = false;
    public static void debug(String message) {
        if (DEBUG) {
            System.err.println(message);
        }
    }
}
```

```
import static Debug.*;

public class Quiz {

    public static void main(String[] args) {

        debug("debug");

    }

}
```

Αποσφαλματωτές

- Κάθε γλώσσα και ολοκληρωμένο περιβάλλον έχει το δικό της αποσφαλματωτή.
- Υποστηρίζουν σημεία διακοπής.
- Ελεγχόμενη εκτέλεση με Step και Step-into.
- Στοίβα κλήσεων.
- Κατάσταση αντικειμένων.

calculator

Σύνοψη

- Τα σφάλματα είναι αναπόφευκτα.
- Καλές πρακτικές προγραμματισμού μειώνουν την εμφάνιση των σφαλμάτων.
- Σημαντικές οι δεξιότητες ελέγχου και αποσφαλματωσης.
- Κάντε τον έλεγχο συνήθεια.
- Αυτοματοποιήστε τον έλεγχο σε κάθε ευκαιρία.
- Συνεχής εκτέλεση ελέγχων.
- Δοκιμάστε διάφορες τεχνικές αποσφαλμάτωσης.

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