6 Καλώς Συμπεριφερόμενα Αντικείμενα

Πώς ελέγχουμε το λογισμικό για σφάλματα

Αντιμετώπιση Σφαλμάτων

- Τα **συντακτικά σφάλματα** εμφανίζονται πρώτα, αλλά τα εντοπίζει ο μεταγλωττιστής και η διόρθωσή τους είναι (σχετικά) απλή υπόθεση.
- Τα **λογικά σφάλματα** είναι δυσκολότερο να εντοπιστούν και να διορθωθούν.
  - Μερικά δεν 'εκδηλώνονται' άμεσα.
  - Είναι στατιστικώς σίγουρο ότι υπάρχουν σε κάθε λογισμικό.
  - Μερικά αντιμετωπίζονται κατά τη (διορθωτική) συντήρηση.
  - Νέα σφάλματα εισάγονται κατά την (τροποποιητική) συντήρηση.

Γραμματική Δήλωσης

Ότι είναι μέσα σε [ ], είναι προαιρετικό.

δήλωση-ονόματος ::= εμβέλεια [**static**] [**final**]
τύπος-ονόματος *όνομα* **;**

εμβέλεια ::= **public** | **private**

τύπος-ονόματος ::= **int** | **long** | **double** | **char** | **boolean** | **String** | *όνομα-κλάσης*

Το | σημαίνει ένα από (ή).

Γραμματική Κλάσης

κλάση ::= **public class** *όνομα-κλάσης* **{**
σώμα-κλάσης
}

Επαναλαμβάνεται 0, 1 ή περισσότερες φορές.

σώμα-κλάσης ::= [δήλωση-ονόματος | δήλωση-μεθόδου]*

δήλωση-μεθόδου ::= εμβέλεια τύπος-μεθόδου *όνομα-μεθόδου*
([τυπικές-παράμετροι] **)** **{** σώμα-μεθόδου **}**

τύπος-μεθόδου ::= τύπος-ονόματος | **void**

τυπικές-παράμετροι ::= τύπος-ονόματος *όνομα-παραμέτρου*
[**,** τύπος-ονόματος *όνομα-παραμέτρου*]*

Γραμματική Εντολών

σώμα-μεθόδου ::= [εντολή]*

εντολή ::= δήλωση-τοπικού-ονόματος | εντολή-ανάθεσης | εντολή-if | εντολή-switch | εντολή-for | εντολή-while | εντολή-return | **{** [εντολή]* **}**

εντολή-ανάθεσης ::= *όνομα* **=** *παράσταση* **;**

εντολή-if ::= **if** **(** *παράσταση* **)** εντολή [**else** εντολή]

εντολή-while ::= **while** **(** *παράσταση* **)** εντολή

Οι πλαγιαστοί όροι δεν αναλύονται παραπάνω (αν και θα έπρεπε).

ΓραμματικήBNF.doc

Άσκηση με Γραμματική

1. πρόταση-πολυλογά ::= πρόταση [*και* πρόταση]* **.**
2. πρόταση ::= υποκείμενο ρήμα
3. Υποκείμενο ::= άρθρο [επίθετο] ουσιαστικό
4. άρθρο ::= *ο* | *η* | *το*
5. επίθετο ::= *όμορφος* | *άσχημη*
6. ουσιαστικό ::= *ημέρα* | *ήλιος*
7. ρήμα ::= *λάμπει* | *φαίνεται*

- *το ήλιος φαίνεται*
- *ο όμορφος ήλιος λάμπει .*
- *ο άσχημη ήλιος λάμπει και το όμορφος φαίνεται.*

Ιωάννης Γαβιώτης
gaviotis@aegean.gr
http://www.syros.aegean.gr/users/gaviotis/tmp2

Αποφυγή & Ανίχνευση Σφαλμάτων

- Με ελέγχους του προγράμματος ανιχνεύουμε κάποια σφάλματα.
- Επιπρόσθετα, πρέπει να γράφουμε τον κώδικα έτσι ώστε να αποφεύγονται τα σφάλματα ή τουλάχιστον να είναι εύκολος ο εντοπισμός και η διόρθωσή τους.
  - Σε αυτό συντελούν τεχνικές της τεχνολογίας λογισμικού (software engineering), όπως:
    - ο εκτενής σχολιασμός,
    - η αρθρωσιμότητα
- Πρέπει να προάγουμε ικανότητες ανίχνευσης.

166

Έλεγχος Μονάδας

- Με τον **έλεγχο μονάδας** (unit testing) δοκιμάζουμε κάθε τμήμα του προγράμματος.
  - Σε επίπεδο κλάσης, μεθόδου
  - Κατανόηση του τι πρέπει να κάνει μια μονάδα **Προγραμματισμός-Με-Συμβόλαιο** (Contract programming): πρέπει να συμφωνεί αυτό που κάνει (υλοποίηση) με αυτό που όφειλε να κάνει (προδιαγραφή)
- Αντιδιαστέλλεται με τον **έλεγχο εφαρμογής** (application testing) που ελέγχει το πρόγραμμα συνολικά.

167

Τεχνικές Ελέγχου

- Παραπανίσιες εντολές εκτύπωσης
- Μη αυτόματες διελεύσεις κώδικα (manual walkthroughs)
- Έλεγχος κατάστασης αντικειμένων
  - Επιθεώρηση τιμών μεταβλητών στιγμιότυπου
- **Έλεγχος ορίων** (boundary checks)
  - Εντατικοί έλεγχοι όταν το πρόγραμμα λειτουργεί στα όριά του, πχ όταν ξεκινά μια επανάληψη κι όταν τερματίζει
- Έλεγχος ότι οι κλήσεις γίνονται με σωστές παραμέτρους.

168

Θετικός & Αρνητικός Έλεγχος

- Όταν ελέγχουμε ότι αυτά που πρέπει να λειτουργούν, λειτουργούν όπως πρέπει κάνουμε **θετικό έλεγχο**.
- Όταν ελέγχουμε τη συμπεριφορά του προγράμματος σε περιπτώσεις που θα έπρεπε να αποτυγχάνει, εκτελούμε **αρνητικό έλεγχο**.
  - Το πρόγραμμα πρέπει να αντιμετωπίζει τις συνθήκες αποτυχίας με προδιαγεγραμμένο τρόπο, πχ να μην 'κρεμάει'.
- Παράδειγμα: πρόγραμμα που επιστρέφει την ημέρα της εβδομάδας μιας ημερομηνίας.
  - Θετικός έλεγχος: `weekDay("2005-04-22")` → `"Fri"`
  - Αρνητικός έλεγχος: `weekDay("2005-02-29")` ???

169

Παλινδρομικός Έλεγχος

- Διορθώνοντας ένα σφάλμα μπορεί να εισάγουμε ένα άλλο. Γι' αυτό μετά από κάθε αλλαγή, οι έλεγχοι πρέπει να επαναλαμβάνονται. Αυτή η διαδικασία ονομάζεται **παλινδρομικός έλεγχος** (regression testing).
- Μπορούμε να αυτοματοποιήσουμε τις δοκιμές και τη συλλογή των αποτελεσμάτων από ελέγχους μονάδας του προγράμματος. Αν εντοπιστεί διαφοροποίηση, τότε αναζητούμε την αιτία.

170

Έλεγχος με Ισχυρισμούς

- Σε σημεία του κώδικα που θέλουμε να ελέγχουμε, 'σπέρνουμε' **ισχυρισμούς** με εντολές
  `assert` *συνθήκη* `:` *μήνυμα*
  Αν η συνθήκη είναι ψευδής, η εκτέλεση θα διακοπεί.

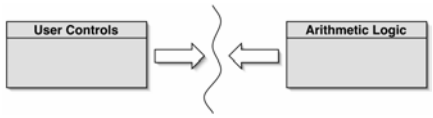
Επιστροφή

```
public void bidFor(int lotNumber, Person bidder, long value) {
  assert value > 0 : "Αρνητική προσφορά " + value;
  assert bidder != null : "Ανύπαρκτος άνθρωπος";
  Lot selectedLot = getLot(lotNumber);
  if (selectedLot != null) {
    if (selectedLot.bidFor(new Bid(bidder, value)))
      System.out.println("Επιτυχής προσφορά για το " + lotNumber);
    else
      System.out.println(lotNumber + " ήδη έχει προσφορά " +
                         selectedLot.getHighestBid().getValue());
  }
  assert selectedLot.getHighestBid() != null : "πρέπει να υπάρχει";
}
```

171

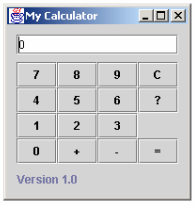
Αρθρώματα Μιας Αριθμομηχανής

- Κάθε άρθρωμα (module) δεν χρειάζεται να γνωρίζει τις λεπτομέρειες της υλοποίησης του άλλου. Χρειάζεται να ξέρει και να χρησιμοποιεί τη δημόσια διεπαφή.
- Η διεπαφή πρέπει να είναι σαφής και καλώς ορισμένη.



Οι Υπογραφές Μεθόδων Είναι η Διεπαφή

```
// Return the value to be displayed.
public int getDisplayValue();

// Call when a digit button is pressed.
public void numberPressed(int number);

// Call when a plus operator is pressed.
public void plus();

// Call when a minus operator is pressed.
public void minus();

// Call to complete a calculation.
public void equals();

// Call to reset the calculator.
public void clear();
```



Διελεύσεις Κώδικα

- **Μη αυτόματη διέλευση** (manual walkthrough)
- Δεν χρειάζεται υπολογιστή
  - Τύπωσε το πρόγραμμα
  - Εκτέλεσέ το με το μυαλό σου
- Κατά τη διέλευση μπορούμε να επιλέξουμε ποιες μεθόδους θα εκτελούμε ως μια εντολή (σαν Step over) και σε ποιες θα μπαίνουμε μέσα στο σώμα τους (σαν Step Into).
- **Φραστική διέλευση** (verbal walkthrough): Εξηγήστε σε κάποιον άλλο τι κάνει ο κώδικας.
  - Μπορεί να εντοπίσει το σφάλμα.
  - Μπορεί να το εντοπίσετε εσείς.

Επιθεώρηση Κατάστασης Αντικειμένου

- Η συμπεριφορά ενός αντικειμένου συνήθως προσδιορίζεται από την κατάστασή του, δηλ. το σύνολο των τιμών των ιδιοτήτων του.
- Σε επιλεγμένα σημεία της εκτέλεσης του κώδικα επιθεωρούμε την κατάσταση συγκεκριμένων αντικειμένων.
- Μπορούμε να παραγγείλουμε στο περιβάλλον ανάπτυξης μπορεί να διακόψει την εκτέλεση όταν μια ιδιότητα παίρνει συγκεκριμένη τιμή ή απλώς αλλάζει.

Εντολές Εκτύπωσης

- Η πιο δημοφιλής τεχνική
- Δεν απαιτεί ειδικά εργαλεία.
- Υποστηρίζεται από όλες τις γλώσσες προγραμματισμού.
- Δίνει αποτέλεσμα αν οι εντολές τοποθετηθούν στα κρίσιμα σημεία, αλλιώς ο όγκος των εκτυπώσεων είναι τεράστιος και δυσκολεύει τον εντοπισμό του σφάλματος.
- Υποστηρίζονται μηχανισμοί ενεργοποίησης κατά την ανάπτυξη και απενεργοποίησης κατά την παραγωγή.

Παράδειγμα Ελέγχου Προγράμματος

- Πρόβλημα: μια εταιρία παράγει τούβλα που τα συσκευάζει σε παλέτες. Υπολογίστε το ύψος και το βάρος τους.
- Διαβάστε τον κώδικα της κλάσης `Brick` και προσπαθήστε να κατανοήσετε τι κάνει.
- Εντοπίστε σφάλματα.
- Επαναλάβετε με τον κώδικα της κλάσης `Palette`.
- Ποια τεχνική χρησιμοποιήσατε;

Brick.html

Palette.html

Ιωάννης Γαβιώτης
gaviotis@aegean.gr
http://www.syros.aegean.gr/users/gaviotis/tmp2

Όροι Ενότητας 6

8/5/2006

- Συντακτικά σφάλματα
- Λογικά σφάλματα
- Έλεγχος μονάδας
- Έλεγχος εφαρμογής
- Προγραμματισμός-Με-Συμβόλαιο
- Έλεγχος ορίων
- Παλινδρομικός έλεγχος
- Θετικός & αρνητικός έλεγχος
- Ισχυρισμοί
- Αρθρώματα
- Διελεύσεις κώδικα
- Επιθεώρηση κατάστασης αντικειμένου
- Εκτύπωση κατά συνθήκη

Τεκμήριο Ενότητας

- `assert`

178