Νικόλαος Δημητρακόπουλος

Πανεπιστήμιο Πελοποννήσου

Σχολή Θετικών Επιστημών & Τεχνολογίας

Τμήμα Επιστήμης & Τεχνολογίας Τηλεπικοινωνιών

Α.Μ.: 2024200200010

Μελέτη για το μάθημα του Αντικειμενοστρεφούς Προγραμματισμού:

«Πολλαπλή Κληρονομικότητα»

Ακαδημαϊκό έτος 2002-2003

Κεφάλαιο 1

Εισαγωγή στον Αντικειμενοστρεφή Προγραμματισμό (Α.Π.)

1.1 Κλάσεις

Οι κλάσεις είναι ίσως το πιο σημαντικό κομμάτι του αντικειμενοστρεφούς προγραμματισμού. Είναι ίσως αυτό που κάνει το αντικειμενοστρεφές μοντέλο να ξεχωρίζει από τον πατροπαράδοτο συναρτησιακό προγραμματισμό αλλά και τα άλλα στυλ προγραμματισμού. Ο ρόλος της είναι να κατηγοριοποιεί και να ομαδοποιεί κάποια κοινά χαρακτηριστικά (πεδία) ενός αντικειμένου, αλλά και να συμπεριλαμβάνει τις δυνατές ,επάνω σε αυτά, πράξεις (μέθοδοι).

1.2 Οριοθέτηση

Ένα άλλο σημαντικό στοιχείο είναι η οριοθέτηση. Με την οριοθέτηση εννοούμε το διαχωρισμό των πεδίων και των μεθόδων μιας κλάσης σε ιδωτικά και κοινόχρηστα (ή δημόσια). Οι δύο αυτές έννοιες έχουν να κάνουνε με το κατά πόσο είναι προσβάσημα τα χαρακτηριστικά μίας κλάσης από κάποια άλλη. Έτσι με τα ιδωτικά (private) οι υπόλοιπες κλάσεις σημαίνει πως δεν έχουνε πρόσβαση σε αυτά ενώ αντίθετα με τα δημόσια (public) οι υπόλοιπες κλάσεις έχουν πλήρη πρόσβαση. Σε ορισμένες γλώσσες υπάρχουν και άλλες υλοποιήσεις περισσότερων διαχωρισμών.

1.3 Κληρονομικότητα

Κληρονομικότητα είναι ο μηχανισμός με τον οποίο τα χαρακτηριστικά (πεδία – μέθοδοι) μιας κλάσης περνάνε σε μία άλλη. Έτσι μπορούμε να παρομοιάσουμε την μία κλάση σαν τον γονιό που κληροδοτεί τα χαρακτηριστικά του και την άλλη σαν το παιδί που κληρονομεί. Θα πρέπει να επισυμανθεί πως δεν κληρονομούνται όλα τα χαρακτηριστικά αλλά μόνο αυτά που δεν είναι ιδιωτικά.

Σκοπός της κληρονομικότητας δεν είναι να αντιγραφεί μία κλάση αλλά να δημιουργηθεί μία νέα που θα είναι η εκλεύπτυνση της υπάρχουσας, δηλαδή κάτι πιο συγκεκριμένο από την ήδη υπάρχουσα παραμένοντας όμως ταυτόχρονα και ότι ήταν. Για παράδειγμα έστω η κλάση Α με όνομα «δένδρο» και κάποια χ χαρακτηριστικά. Έστω τώρα ότι θέλουμε να ορίσουμε ένα συγκεκριμένο είδος δένδρου όπως το πλατάνι. Η νέα κλάση «πλατάνι» είναι ένα «δένδρο» αλλά και κάτι περισσότερο από αυτό. Έτσι έχει τα χ χαρακτηριστικά του «γονέα» αλλά και κάποια άλλα ψ, δικά του, που το κάνουν να διαφέρει από τα υπόλοιπα δένδρα.

1.5 Πολλαπλή κληρονομικότητα.

Πολλαπλή κληρονομικότητα ονομάζεται, όπως είναι άλλωστε προφανές, η περίπτωση όπου δεν έχουμε μόνο μία υπερκλάση (γονέα), αλλά περισσότερες. Στην περίπτωση αυτή η υποκλάση κληρονομεί χαρακτηριστικά από όλες τις υπερκλάσεις της, κάτι που κάνει ένα πρόγραμμα πολύ πιο ευέλικτο αλλά παράλληλα δημιουργούνται πολλά προβλήματα, όπως θα δούμε παρακάτω πιο αναλυτικά.

Κεφάλαιο 2

Εισαγωγή στην κληρονομικότητα

Για να εμβαθύνουμε στο θέμα της πολλαπλής κληρονομικότητας πρέπει πρώτα να αναλύσουμε ως ένα βαθμό την κληρονομικότητα γενικά, αφού από εκεί ξεκινάνε όλα.

2.1 Χαρακτηριστικά της κληρονομικότητας

2.1.1. Εισαγωγή – Βασικά χαρακτηριστικά

Όπως αναφέρθηκε και πιο πριν η κληρονομικότητα είναι ένας από τους μηχανισμούς του αντικειμενοστρεφούς προγραμματισμού για την συσχέτηση αντικειμένων μεταξύ τους. Πιο συγκεκριμένα μέσω αυτής περνάμε χαρακτηριστικά από μία «μητέρα» κλάση σε ένα «παιδί».

Όπως είναι λογικό μία κλάση δεν περιορίζεται στο πόσες υποκλάσεις μπορεί να έχει. Αυτό είναι λογικό αφού αν λάβουμε υπόψιν μας το προηγούμενο παράδειγμα όπου έχουμε την υπερκλάση «δένδρο», δεν είναι μόνο ο πλάτανος υποκλάση της αλλά και όλα τα άλλα δένδρα όπως ιτιά, ελιά κ.λ.π.

Επίσης μία κλάση που κληροδοτεί μπορεί ταυτόχρονα και αυτή να κληρονομεί από μία άλλη υπερκλάση. Αυτό δείχνει τη δυναμικότητα που έχει το μοντέλο αυτό καθώς και το πόσο πολύ λεπτομέρεια μπορούμε να έχουμε κατά τον σχεδιασμό.

2.1.2. Σχέση κληρονομικότητας και οριοθέτησης

Υπάρχει άμεση συσχέτηση μεταξύ της κληρονομικότητας και της οριοθέτησης. Αυτό γίνεται γιατί , όπως συμβαίνει γενικότερα με κάποια χαρακτηριστικά μίας κλάσεις που δεν θέλουμε να είναι προσβάσημα από ξένες προς αυτή κλάσεις, κάποια πεδία ή μέθοδοί της , θέλουμε να παραμένουνε «προσωπικά» και μη προσβάσημα ούτε από τις υποκλάσεις τις. Πρακτικά αυτό σημαίνει ότι όταν γίνεται η οριοθέτηση σε μία κλάση αυτό έχει άμεση συνέπια και ως προς τα χαρακτηριστικά που θα κληροδοτηθούνε.

2.1.3. Πολυμορφισμός

Όταν έχουμε ένα στιγμιότυπο μίας υποκλάσης μπορεί να θέλουμε να το δούμε σαν στιγμιότυπο της υπερκλάσης. Το γιατί αυτό μπορεί να είναι χρήσιμο γίνεται αντιλιπτό στην επόμενη παράγραφο. Σε αυτή την περίπτωση λέμε ότι έχουμε πολυμορφισμό ενός αντικειμένου.

Σημαντικό είναι να ξεκαθαρίσουμε ότι πολυμορφισμό μπορούμε να έχουμε μόνο από κάτω προς τα πάνω, δηλαδή από μία υποκλάση σε μία υπερκλάση. Επίσης πρέπει να επισημάνουμε ότι ο πολυμορφισμός είναι εφικτός λόγο του ότι στις γλώσσες προγραμματισμού που υποστηρίζεται χρησιμοποιείται late binding και όχι early binding όπως συνηθιζόταν στις προγενέστερες γλώσσες. Ο λόγος είναι η αυξημένη πολυπλοκότητα για τον σαφή διαχωρισμό των πολυμορφισμένων αντικειμένων, κάτι που συνεπάγεται και μείωση της ταχύτητας έναντι των πατροπαράδωτων γλωσσών προγραμματισμού που υλοποιούν το διαδικαστικό

μοντέλο. Δεν θα επεκταθούμε περισσότερο στο late binding, για περισσότερες πληροφορίες μπορούν να βρεθούν σε σχετική βιβλιογραφεία.

2.1.4. Υπέρβαση

Πολλές φορές όταν κληρονομούνται κάποια χαρακτηριστικά, είτε αυτά είναι μέθοδοι είτε πεδία ενός αντικειμένου, μπορεί να θέλουμε να τα «επαναπροσδιορίσουμε». Αυτό μπορεί να συμβαίνει γιατί μαζί με την εκλέπτυνση ενός αντικειμένου, όπου προσθέτονται νέα χαρακτηριστικά, αρκετές φορές πρέπει να εκλεπτυνθούν και τα χαρακτηριστικά του.

Όταν γίνεται αυτό λέμε ότι έχουμε «υπερφόρτωση» (override) . Αυτό πρακτικά συμαίνει ότι από την στιγμή που το αντικείμενο θεωρείται στυγμιότυπο της κλάσης που έχει γίνει η υπέρβαση, δεν χρησιμοποιείται η μέθοδος ή το πεδίο της υπερκλάσης αλλά ο επαναπροσδιορισμός τους που βρίσκεται στην υποκλάση. Σε αυτό ακριβώς το σημείο έρχεται και εμπλέκεται ο πολυμορφισμός και άρα η χρήση του late binding που καλείται να διαλέξει ποιάς κλάσης είναι το στιγμιότυπο.

Κεφάλαιο 3

Πολλαπλή Κληρονομικότητα (Π.Κ.)

3.1 Εισαγωγή στην Π.Κ.

Όπως είδαμε προηγουμένος μία κλάση μπορεί να έχει πολλαπλούς απογόνους ή / και να είναι και αυτή απόγονος μία άλλης υπερκλάσης. Μερικές γλώσσες όμως υποστηρίζουν την «πολλαπλή κληρονομικότητα». Τι ακριβώς είναι αυτό; Είναι η δυνατότητα που δίνεται ώστε μία κλάση να κληρονομεί από περισσότερους από έναν προγόνους, δηλαδή να έχει περισσότερες από μία υπερκλάσεις.

Αυτό δίνει πραγματικά πολύ περισσότερες δυνατότητες κατά την υλοποίηση ενός προγράμματος αφού γίνεται άμεση επαναχρησιμοποίηση κώδικα. Παρόλα αυτά μπορεί να δημιουργήσει και πολλά προβλήματα κατά τον σχεδιασμό ενός συστήματος αφού το σχεδιάγραμμα των κλάσεων αρχίζει και περιπλέκεται αρκετά. Επίσης υπάρχουν και πολλά άλλα προβλήματα που δημιουργούνται κάνωντας χρήση της και έτσι πολλοί ερευνητές έχουν διαφωνήσει στο αν η Π.Κ. είναι «καλή» ή «κακή».

Η ΠΚ είναι ένα πολύ δυνατό χαρακτιριστικό στον αντικειμενοστρεφή προγραμματισμό. Έχει πολύ μεγάλο ενδιαφέρον στον τομέα της τεχνιτής νοημοσύνης κυρίως για την δύναμη που δίνει στον τομέα της ιεαρχείας των κλάσσεων. Επίσης έχει πολλά πλεονεκτήματα κυρίως στον τομέα της τμηματοποίησης, της επαναχρησιμοποίησης και του αυξητικού σχεδιασμού.

3.2 Η Π.Κ. μέσα από την ζωή

Ο αντικειμενοστρεφής προγραμματισμός γενικότερα, αλλά και πιο συγκεκριμένα η κληρονομικότητα σε αυτόν , σχεδιάστηκαν πέρνοντας σαν παράδειγμα το πως βλέπουμε τα πράγματα γύρω μας (με το παράδειγμα του κεφαλαίου 1 & 2 μπορεί αυτό να γίνει κατανοητό). Όμως στην πραγματικότητα ένα οποιοδήποτε πράγμα (αντικείμενο) σπάνια έχει μόνο μία πλευρά. Συνήθως έχει από πολλές μέχρι άπειρες σχέσεις με το υπόλοιπο κόσμο, απλά μας αρκεί το από ποια μεριά θα το δούμε (άλλωστε όπως είχε πει και ο Αινστάιν «τα πάντα είναι σχετικά»). Με αυτό το σκεπτικό εισήχθηκε στο αντικειμενοστρεφές μοντέλο και η έννοια της πολλαπλής κληρονομικότητας. Για να γίνει πιο κατανοητό αυτό μπορούμε να δούμε ένα παράδειγμα.

Έστω η κλάση άνθρωπος. Η κλάση αυτή έχει σαν υποκλάσεις τις εξής υποκλάσεις: φοιτητής, μαθητής, εργαζόμενος, άνεργος, έγγαμος, άγαμος, ορφανός κ.λ.π. Αμέσως γίνεται κατανοητό ότι κάποιος άνθρωπος μπορεί να είναι ταυτόχρονα φοιτητής, εργαζόμενος και έγγαμος (πω, πω, αυτός έχει μεγάλο πρόβλημα ☺ !!!) . Είναι άρα πολύ λογικό να θέλουμε να συνδιάσουμε περισσότερες από μία κλάσεις ταυτόχρονα.

3.3 Θετικές επιπτώσεις από την χρήση της Π.Κ.

Βλέποντας το προηγούμενο παραδείγμα κατανοούμε το που μπορεί να χρησιμεύσει η Π.Κ. Τα πλεονεκτήματα μπορεί να είναι ανεκτίμητα σε αυτές τις περιπτώσεις. Καταρχήν, όπως αναφέρθηκε και προηγουμένως γίνεται άμεση επαναχρησιμοποίηση κώδικα καθώς και συνδιασμός του με ακόμα καλύτερη «περιγραφή» των αντικειμένων που θέλουμε. Σε αντίθετη περίπτωση θα έπρεπε να χρησιμοποιήσουμε μεθόδους που ταιριάζουν καλύτερα στο διαδικαστικό μοντέλο (Άμεσο παράδειγμα είναι η $1^{η}$ προγραμματιστική άσκηση, η οποία υλοποιείται στην γλώσσα Java η οποία δεν υποστηρίζει Π.Κ. Για την υλοποίηση του συστήματος αυτού χρησιμοποιήθηκαν αρκετές Boolean μεταβλητές και αρκετές συνθήκες if που, εν μέρη, καταστρατιγούσαν το ιδεαλιστικό διαδικαστικό μοντέλο και περιορίζανε την αφαιρετικότητα και την γενικότητα του προγράμματος – περισσότερη αναφορά στην υποστήριξη της Π.Κ. από αντικειμενοστρεφείς γλώσσες προγραμματισμού γίνεται σε επόμενο κεφάλαιο).

3.4 Αρνητικές επιπτώσεις από την χρήση της Π.Κ.

3.4.1 Το πρόβλημα των ασύμβατων χαρακτηριστικών.

Η Π.Κ. δεν είναι όμως τόσο καλή όσο φαίνεται. Υπάρχουν αρκετά προβήματα από την χρησιμοποίησή της. Το πρώτο πρόβλημα είναι κυρίως νοηματικό και ενδέχεται να μην δημιουργεί άμεσα πρόβλημα σε κάποια υλοποίηση ενός συστήματος. Αυτό ξεκινάει από το γεγονός ότι όταν κληρονομούμε από δύο ή περισσότερες κλάσεις η υποκλάση είναι υποχρεωμένη να πάρει όλα τα χαρακτηριστικά των υπερκλάσεων της χωρίς καμία εξαίρεση (εκτός βέβαια από τις περιπτώσεις όπου μεσολαβεί η οριοθέτηση) .

Στο προηγούμενο παράδειγμα με τον άτυχο κύριο συνδιάσαμε τις κλάσεις φοιτητής, εργαζόμενος και έγγαμος. Κάθε μία από αυτές τις κλάσεις έχει κάποιες

ιδιαίτερες μεθόδους (απ'τη στηγμή που η κληρονομικότητα χρησιμοποιείται για εκλέπτυνση και συγκεκριμενοποίηση) που δεν τις έχουν οι άλλες. Για παράδειγμα ο φοιτητής παίζει τάβλι και σαχλαμαρίζει με κοπελίτσες, ο εργαζόμενος ξεκουράζεται και ο έγγαμος πηγαίνει για καφέ με την γυναίκα του κ.λ.π. Στην προκειμένη περίπτωση αυτού του συνδιασμού είναι ευκόλως εννοούμενο ότι ,στον πραγματικό κόσμο τουλάχιστον, αυτά δεν μπορούν να συνυπάρξουν...

Αν το δούμε από μία πιο «προγραμματιστική» σκοπιά μπορούμε να δούμε ότι αυτό το νοηματικό πρόβλημα μπορεί όντως να δημιουργήσει προβλήματα. Σαν παράδειγμα μπορούμε να πάρουμε αυτό του Meyer στο οποίο δημιουργεί μία υποκλάση των εξής κλάσεων : πίνακας και στοίβα. Όπως είναι αναγκαστικό στην Π.Κ. η νέα κλάση αυτή θα πάρει τα χαρακτηριστικά και των δύο της προγόνων. Όμως δημιουργείται το πρόβλημα ότι σε μία στοίβα δεν μπορούμε να έχουμε πράξεις όπως οι δείκτες των στοιχείων που έχουμε σε έναν πίνακα.

3.4.2 Το πρόβλημα των ομόνυμων κληροδοτούμενων χαρακτηριστικών

Ένα πολύ σημαντικό πρόβλημα που μπορεί να προκύψει από την χρήση της Π.Κ. είναι στην περίπτωση όπου δύο οι περισσότερες υπερκλάσεις έχουν ένα πεδίο ή μία μέθοδο με το ίδιο όνομα και πόσο μάλλον στην περίπτωση όπου τα χαρακτηριστικά αυτά είναι ασύμβατα μεταξύ τους. Το πρόβλημα αυτό θα κάνει την εμφάνισή του την στιγμή που κληθεί ένα από αυτά τα χαρακτηριστικά καθώς το σύστημα δεν θα γνωρίζει σε ποια από όλες τις κλάσεις που το εμπεριέχουν αναφέρεται.

3.4.3 Δυσκολία σχεδιασμού

Τέλος ένα άλλο μειονέκτημα του αντικειμενοστρεφούς μοντέλου που εντείνεται με την Π.Κ. είναι η δυσκολία στο σχεδιασμό. Όσο οι κλάσεις αλλά και οι μεταξύ τους σχέσεις αυξάνουν η δυσκολία αλλά και ο χρόνος που απαιτείται για τον σχεδιασμό ενός συστήματος πολλαπλασιάζεται. Παρόλα αυτά αυτό συνήθως αντισταθμίζεται από το γεγονός ότι η μετέπειτα υλοποίηση είναι πολύ πιο εύκολη αλλά και από το ότι μπορεί πολύ εύκολα το σύστημα μας να εξελιχθεί και να αυξηθεί ανάλογα με τι εκάστοτε ανάγκες. Αυτό στον χώρο τον υπολογιστών είναι ένα αρκετά σημαντικό σημείο και γι'αυτόν τον λόγο τελικά καταλίγει ο αντικειμενοστρεφής προγραμματισμός να υπερέχει σε ορισμένα σημεία έναντι των άλλων μοντέλων.

3.5 Σύκγριση θετικών και αρνητικών στοιχείων

Κάποιος βλέποντας όλα τα παραπάνω θα μπορούσε εύκολα να σκευτεί ότι η Π.Κ. καταλήγει να έχει περισσότερες αρνητικές επιπτώσεις απ'ότι θετικές. Παρόλα αυτά η σύγκριση αυτή δεν πρέπει να γίνεται έτσι επιπόλαια. Δύο είναι οι κύριες παράμετροι που παίζουνε ρόλο, πρώτον η ορθή χρήση του χαρακτηριστικού αυτού από τον προγραμματιστή αλλά κυρίως και από την εκάστοτε γλώσσα προγραμματισμού. Όπως θα δούμε παρακάτω η κάθε μία γλώσσα προσφέρει (ή δεν προσφέρει) συγκεκριμένο τρόπο υποστήριξης και διασφάλησης όσο το δυνατό λιγότερων προβλημάτων.

Η πολλαπλή κληρονομικότητα είναι καλή αλλά δεν υπάρχει σωστός τρόπος για την χρησιμοποίησή της. ***Alan Snyder***

Κεφάλαιο 4

Υποστήριξη Π.Κ. από γλώσσες προγραμματισμού και τρόποι επίλυσης των προβλημάτων της

4.1 Μερικές γλώσσες που υποστηρίζουν Π.Κ.

Eiffel	**C++**	**Java**	**Smalltalk**
Πλήρης υποστήριξη πολλαπλής κληρονομικότητας	Πλήρης υποστήριξη πολλαπλής κληρονομικότητας αλλά με προβλήματα	Απλή κληρονομικότητα αλλά υποστήριξη πολλαπλών interfaces	Απλή κληρονομικό-τητα

4.2 Υλοποιήσεις

Όπως αναφέρθηκε και πιο πριν η εκάστοτε γλώσσα προσφέρει διαφορετική προσέγκιση στο θέμα της πολλαπλής κληρονομικότητας. Επίσης είδαμε ότι το κύριο πρόβλημα της Π.Κ. είναι το πρόβλημα των ομόνυμων χαρακτηριστικών. Στις επόμενες παραγράφους θα δούμε πως κάθε γλώσσα προσπαθεί να επιλύσει το πρόβλημα αυτό.

4.2.1 Eiffel

Η προσέγκιση της Eiffel είναι μία από τις πιο προσεγμένες αλλά και ολοκληρωμένες στο θέμα αυτό. Για να μπορέσει να αντιμετωπίσει αυτό το πρόβλημα υποστιρίζει την «μετονομασία κληροδωτούμενων μεθόδων» (inherited method renaming). Χαρακτηριστικό είναι ότι στην ιστοσελίδα της γλώσσας αυτής αναφέρεται το εξής: *«Μην ακούτε αυτούς που λένε ότι η πολλαπλή κληρονομικότητα είναι κακή. Απλά δεν την υποστηρίζει η γλώσσα τους ή την υποστηρίζει ελειπώς».*

4.2.2 C++

Στην C++ για να ξεπεραστεί το πρόβλημα αυτό απαιτείται reimplementation. Για αυτόν το λόγο θεωρείται ανεπαρκής ή ελλειπής η υποστήριξη της Π.Κ. καθώς επίσης αυξάνεται και η πολυπλοκότητα της γλώσσας.

4.2.3 Java

Η Java δεν υποστηρίζει την πολλαπλή κληρονομικότητα. Ο μόνος τρόπος για να πετύχεις κάτι αντίστοιχο είναι η χρήση interfaces. Παράδειγμα σε αυτό μπορεί να είναι η κλάση Thread με interface το Runnable. Σε περίπτωση που θέλουμε η κλάση μας να έχει πιο άμεση σχέση με την Thread την κάνουμε extend. Όμως αν θέλουμε παράλληλα να κληρονομεί και από κάποια άλλη οποιαδήποτε κλάση τότε αναγκαστικά χρησιμοποιούμε το interface της Thread.

Η Sun (δημιουργός της Java) αναφέρει ενδεικτικά τα εξής για την μη υποστήριξη της Π.Κ.:

Η ομάδα που σχεδίασε την Java, ήθελε αυτή να είναι:

- *Απλή, αντικειμενοστρεφής και γνωστή*
- *Συμπαγής και ασφαλής*
- *Ουδέτερη σε σχέση με την πλατφόρμα και φορητή (μεταφέρσιμη)*
- *Υψηλής απόδοσης*
- *Δυναμική και με νήματα*

Οι λόγοι που τους οδήγησαν στο να μην υποστηρίξουν την Π.Κ. ήταν κυρίως ότι ήθελαν την γλώσσα απλή και εύκολα χρησιμοποιήσημη χωρίς ιδιαίτερη εκπαίδευση. Επίσης θέλανε να είναι «γνωστή» (familiar) έτσι την έκαναν να μοιάζει με την C++ χωρίς όμως την επιπρόσθετη πολυπλοκότητά της.

Κατά την γνώμη των δημιουργών, η Π.Κ. δημιουργεί περισσότερα προβλήματα και σύγχηση, απ'όσα λύνει. Έτσι «έκοψαν» την Π.Κ.(όπως επίσης και την υπερφόρτωση τελεστών). Η εμπειρία τους στην C++ τους δίδαξε ότι η Π.Κ. δεν άξιζε τον πονοκέφαλο.

Έτσι αποφάσησαν να την επιτρέψουν μόνο μέσω των interfaces, μία ιδέα δανεισμένη απο την Objective C.

Βιβλιογραφεία

- On the notion of inheritance [Antero Taivalsaari]
- Αυξητικές μεταβολές στον Α.Π. [Δημήτριος Θεοτόκης]
- Is Multiple Inheritance Essential to OOP? (PANEL) [Yen-Ping Shan, Tom Cargill, Brad Cox, William Cook, Mary Loomis, Alan Snyder]