2010

Πτυχιακή Εργασία

Απομακρυσμένη διαχείριση και έλεγχο απλών πληροφοριακών συστημάτων Client-Server τεχνολογίας με την αντικειμενοστραφή γλώσσας προγραμματισμού JAVA

Σπουδαστής: Δροσάκης Ιωάννης
Εισηγητής: Μαυρικάκης Ιωάννης

Ευχαριστίες

Θα ήθελα να ευχαριστήσω σε αυτό το σημείο αρχικά την οικογένεια μου που με στήριξε 4 χρόνια στο ηράκλειο και έκαναν μεγάλη υπομονή. Τους φίλους μου Γιώργο, Κώστα, Στέλιο, και την υπόλοιποι παρέα που με άντεξαν και με στήριξαν. Επίσης θέλω να πω ένα μεγάλο ευχάριστο και σε δυο πρόσωπα που σταθήκανε δίπλα μου όσο ήμουν στο ηράκλειο τον Γιάννη και την Μαρία. Δεν θα μπορούσα σε αυτό το σημείο να ξεχάσω και τον κ. Μαυρικάκη που πίστεψε σε εμένα και με βοήθησε στην υλοποίηση της πτυχιακής. Ευχαριστώ πολύ για όλα.

1ο Λειτουργικό Σύστημα
1.1 Τι είναι……6
1.2 Τι παρέχει ένα Λ.Σ……6
1.3 Ταξινόμηση Λ.Σ. ανάλογα με το τι παρέχουν ……7
1.4 Ιστορική αναδρομή……8
1.5 Επίλογος……13

2ο Δίκτυο και Διαδίκτυο
2.1 Τι είναι δίκτυο……14
2.1.1 Κατηγορίες δικτύων ανάλογα με την γεωγραφική κάλυψη……14
2.2 Τι είναι Διαδίκτυο……14
2.2.1 Ιστορική αναδρομή……15
2.2.2 Η σημερινή μορφή του……15
2.2.3 Το WWW……16
2.3 Το πρότυπο TCP\IP……17
2.3.1 Το πρωτόκολλο UDP……17
2.4 Το σύστημα Client – Server……18
2.4.1 Τι είναι Client……19
2.4.2 Τι είναι Server……19
2.4.3 Πως ανατήχθηκε το σύστημα Client – Server……20

3ο Αντικειμενοστραφής προγραμματισμός
3.1 Τι είναι ……22
3.2 Ιστορική αναδρομή ……22
3.3 Το Object Model……23
3.4 Τι είναι:……24
3.4.1 Κλάσεις……25
3.4.2 Ιδιότητες……25
3.4.3 Μέθοδοι……25
3.4.4 Γεγονότα……26
3.4.5 Κληρονομικότητα……26
3.4.6 Ενθυλάκωση……26
3.4.7 Πολυμορφισμός……27
3.5 JAVA……27
3.5.1 Τι είναι……28
3.5.2 Ιστορική αναδρομή ……28
3.5.3 Πλεονεκτήματα……29
3.5.4 Η μορφές της Java……31
3.6 Net Beans……32

3.6.1 Τι είναι……32
3.6.2 Πλεονεκτήματα-Μειονεκτήματα……32

4^{ο} Απομακρυσμένη διαχείριση και έλεγχο απλών πληροφοριακών συστημάτων. Client-Server τεχνολογίες με την αντικειμενοστραφή γλώσσα προγραμματισμού JAVA

4.1 Client
4.2.1 Αποστολή εικόνας.....33
4.2.2 Κινήσεις που γίνονται στον client……37
4.2.3 Κινήσεις που επιτρέπονται στον server……40
4.2.4 Το κυρίως πρόγραμμα του client……41
4.2 Server
4.2.1 Η κλάση που διαχειρίζεται της κινήσεις……46
4.2.2 Αποστολή τον event από τον server……51
4.2.3 Απεικόνιση του Client στον Server……54
4.2.4 Κινήσεις που κάνει ο server στον client……56
4.2.5 Το κυρίως πρόγραμμα του server……57

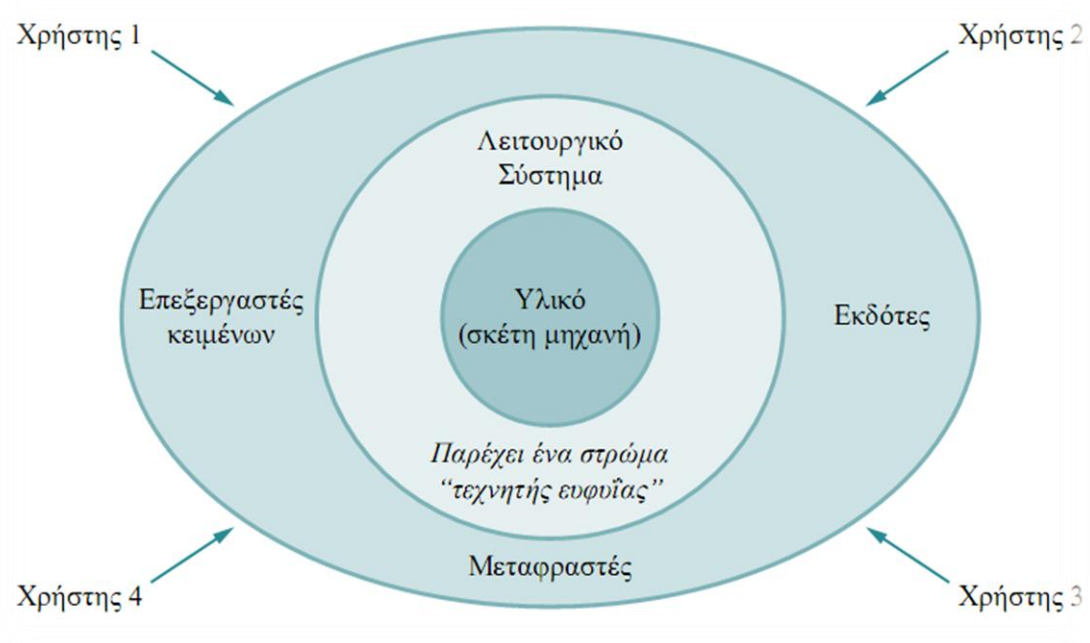
Τι είναι

Το λειτουργικό σύστημα είναι μια συλλογή από προγράμματα τα οποία ενεργούν ως «ενδιάμεσο» μεταξύ των χρηστών (π.χ. προγράμματα, εφαρμογές, συσκευές, άνθρω- ποι) και του Η/Υ. Αυτά τα προγράμματα μπορεί να είναι γραμμένα σε γλώσσα μηχα- νής ή assembly, ή ακόμα και σε μια γλώσσα όπως η C. Τα τελευταία χρόνια παρου- σιάζεται η τάση ορισμένα από αυτά τα προγράμματα να είναι γραμμένα σε μικρο- κώδικα (κώδικα στοιχειωδών εντολών, ενσωματωμένο στο hardware). Χωρίς αμφιβολία, δύο είναι οι βασικοί στόχοι που εξυπηρετεί ένα λειτουργικό σύστημα:

- Να διαχειρίζεται τα δομικά στοιχεία (π.χ. μνήμη, hardware) του υπολογιστή
- Να εξασφαλίζει τη λειτουργία του Η/Υ με αποδοτικό τρόπο (ως προς το ποσοστό χρόνου χρήσης πόρων, τους χρόνους απόκρισης σε εντολές, τη διευκόλυνση του χρήστη κτλ.).

Τι παρέχει ένα Λ.Σ.

1. Παρέχει διευκολύνσεις στο χρήστη για να μπορεί να χρησιμοποιεί τους πόρους του Η/Υ. Με τον όρο «πόροι ενός Η/Υ» εννοούμε:
   - Τα εργαλεία Εισόδου/Εξόδου
   - Τη μνήμη
   - Την κεντρική μονάδα επεξεργασίας
   - Δεδομένα με κάποιας μορφής οργάνωση (π.χ. αρχεία, μηνύματα)



2. Ελέγχει την κατανομή των πόρων ενός Η/Υ στους χρήστες του με κάποια πολιτική κατανομής πόρων κατά τρόπο αποδοτικό, φιλικό και δίκαιο, χωρίς να δημιουργούνται προβλήματα λειτουργίας.

3. Εξασφαλίζει την προστασία των πόρων και των προγραμμάτων των χρηστών από ανεπιθύμητες καταστάσεις (σφάλμα εκτέλεσης προγράμματος, προσπέλαση σε μνήμη που ανήκει σε άλλο χρήστη ή στο σύστημα κτλ.)

Ταξινόμηση Λ.Σ. ανάλογα με το τι παρέχουν

Μια πρώτη ταξινόμηση των λειτουργικών συστημάτων μπορεί να γίνει σε σχέση με το είδος της αλληλεπίδρασης που επιτρέπεται ανάμεσα σε ένα χρήστη και στο πρόγραμμά του, καθώς και στους περιορισμούς που μπαίνουν σε σχέση με την απόκριση του συστήματος. Υπάρχουν τρεις ιδανικές κατηγορίες λειτουργικών συστημάτων με βάση την πιο πάνω ταξινόμηση:

1. Λειτουργικό σύστημα ομαδικής επεξεργασίας. Είναι ένα λειτουργικό σύστημα στο οποίο οι εργασίες (τα προγράμματα) των χρηστών υποβάλλονται στον Η/Υ σε ομάδες. Σε αυτού του είδους τα λειτουργικά συστήματα, δεν είναι δυνατή καμιά αλληλεπίδραση ανάμεσα σε ένα χρήστη και στο πρόγραμμά του κατά τη διάρκεια της επεξεργασίας του προγράμματος. Ο χρόνος απόκρισης του Η/Υ για κάθε χρήστη είναι το χρονικό διάστημα από την υποβολή ως την παραλαβή της εργασίας του χρήστη. Το λειτουργικό σύστημα έχει τη δυνατότητα να κάνει καλή διανομή πόρων και να εφαρμόσει καλή πολιτική ως προς το χρόνο και τη σειρά εκτέλεσης των εργασιών.

2. Λειτουργικό σύστημα με μοίρασμα χρόνου. Εδώ το λειτουργικό σύστημα παρέχει τις υπολογιστικές του υπηρεσίες σε πολλούς χρήστες παράλληλα, επιτρέποντας στον καθέναν από αυτούς να αλληλεπιδρά με το πρόγραμμά του. Το φαινόμενο της «ταυτόχρονης» πρόσβασης στον Η/Υ είναι φυσικά απατηλό. Μοιράζοντας όμως το χρήσιμο χρόνο της ΚΜΕ (Κεντρική Μονάδα Επεξεργασίας) και άλλους πόρους του Η/Υ ανάμεσα σε πολλούς χρήστες, το λειτουργικό σύστημα δημιουργεί την ψευδαίσθηση στους χρήστες ότι ο καθένας από αυτούς έχει το σύνολο των πόρων του Η/Υ στη διάθεσή του. Το λειτουργικό σύστημα κατορθώνει να δίνει πάντοτε κάποια απόκριση σε οποιαδήποτε απαίτηση του χρήστη μέσα σε λίγο χρόνο. Στην πραγματικότητα, ο Η/Υ στρέφει την προσοχή του στην κάθε εργασία για ένα μικρό κομμάτι του χρόνου. Αν η εργασία δεν τελειώσει στο τέλος αυτού του χρόνου, τότε διακόπτεται προσωρινά και «αποθηκεύεται» σε μια ουρά αναμονής. Αυτό επιτρέπει σε κάποια άλλη εργασία να πάρει χρόνο στον Η/Υ κ.ο.κ.

3. Λειτουργικό σύστημα πραγματικού χρόνου. Το λειτουργικό αυτό σύστημα εξυπηρετεί τις εργασίες που του υποβάλλονται μέσα σε αυστηρούς χρονικούς περιορισμούς. Δηλαδή ο χρόνος απόκρισης του Η/Υ είναι αυστηρά ορισμένος. Τέτοια λειτουργικά σύστημα συναντάμε, για παράδειγμα, σε αεροδρόμια για τον έλεγχο της εναέριας κυκλοφορίας. Σήματα διακοπής από τις εργασίες καλούν τον Η/Υ να διαθέσει υπολογιστικούς πόρους σε αυτές. Αν αυτά τα σήματα δεν πάρουν απάντηση σύντομα ή αν η επιθυμητή λειτουργία του Η/Υ αργήσει, τότε η εργασία ίσως αποτύχει ως προς το σκοπό της ή δεχτεί λάθος απαντήσεις.

Ένα συγκεκριμένο λειτουργικό σύστημα μπορεί να ανήκει σε περισσότερες από μία από τις πιο πάνω εξιδανικευμένες κατηγορίες. Είναι συνηθισμένο φαινόμενο τα λειτουργικά συστήματα με μοίρασμα χρόνου και αυτά του πραγματικού χρόνου να έχουν τη δυνατότητα να συμπεριφέρονται και ως λειτουργικά συστήματα ομαδικής επεξεργασίας. Αυτό το κατορθώνουν με διαφανή προς τους χρήστες τρόπο, όταν δεν υπάρχει στο προσκήνιο καμιά εργασία.

Ιστορική αναδρομή

Τα λειτουργικά συστήματα, όπως και οι Η/Υ, είχαν μια εξέλιξη που χαρακτηρίστηκε από σημαντικές αλλαγές στον τρόπο υλοποίησής τους, στους υπολογιστικούς πόρους στους οποίους δίνουν έμφαση και στην αντιμετώπιση του τελικού χρήστη. Αυτές οι αλλαγές δημιούργησαν τις εξής γενιές λειτουργικών συστημάτων:

Πρώτη γενιά 1940

Τα πρώτα υπολογιστικά συστήματα δεν είχαν λειτουργικό σύστημα. Οι χρήστες είχαν άμεση προσπέλαση στη γλώσσα μηχανής και προγραμμάτιζαν τα πάντα κυριολεκτικά «με το χέρι».

Δεύτερη γενιά 1950 – 1960

Από το 1949 μέχρι το 1956 η βασική οργάνωση και ο τρόπος λειτουργίας του Η/Υ έμεινε σχετικά στο ίδιο σημείο. Η κλασική αρχιτεκτονική von Neumann των τότε Η/Υ συμπληρωνόταν με μια αυστηρά ακολουθιακή εκτέλεση εντολών, συμπεριλαμβάνοντας εντολές εισόδου – εξόδου. Ακόμα και στη διάρκεια του φορτώματος(loading) και του τρεξίματος προγραμμάτων, οι χρήστες δούλευαν πάνω στην «κονσόλα», μεταβάλλοντας το περιεχόμενο των καταχωρητών, εκτελώντας εντολές βήμα βήμα, εξετάζοντας θέσεις μνήμης και, γενικά, αλληλεπιδρώντας με τον Η/Υ στο χαμηλότερο δυνατό επίπεδο (μηχανής). Τα προγράμματα γράφονταν σε απόλυτη γλώσσα μηχανής (π.χ. δεκαδικό ή οκταδικό σύστημα) και για τη φόρτωσή τους χρησιμοποιούνταν ένας «απόλυτος» φορτωτής προγραμμάτων (loader), που ήταν σε καθορισμένη θέση της μνήμης και που φόρτωνε το πρόγραμμα σε κάποιο καθορισμένο σύνολο διαδοχικών θέσεων μνήμης. Σε εκείνα τα «άγρια χρόνια», το λεγόμενο «υποβοηθητικό λογικό» (programming aids) βασικά δεν υπήρχε. Σιγά σιγά, όμως, καθώς γινόταν αντιληπτή η σημασία του συμβολικού προγραμματισμού και καθώς οι μεταφραστές σε «γλώσσες μηχανής» (assemblers) άρχισαν να εμφανίζονται, άρχισε να δημιουργείται μια «προκαθορισμένη» σειρά λειτουργιών: Ένας φορτωτής φόρτωνε έναν assembler στο σύστημα. Ο assembler μετάφραζε (σε απόλυτο κώδικα μηχανής) μερικούς σωρούς από κάρτες με προγράμματα χρηστών και «λογικό βιβλιοθήκης» (library routines). Ο απόλυτος κώδικας που έβγαινε από αυτή τη φράση γραφόταν σε ταινία ή κάρτες και μετά ένας φορτωτής ξαναχρησιμοποιούνταν για να φορτώσει τον κώδικα (ως ένα ενιαίο σύνολο πια) στην κύρια μνήμη του (απόλυτου) προγράμματος.

Οι παραπάνω αδυναμίες, μαζί με μια σειρά άλλους λόγους (το κόστος της συνεχούς ανθρώπινης παρέμβασης στη διαδικασία, τη διαθεσιμότητα της FORTRAN και άλλων γλωσσών, την ανάπτυξη κώδικα βιβλιοθήκης και κώδικα λειτουργιών εισόδου – εξόδου), έγιναν πιεστικοί λόγοι για την εξέλιξη των ΛΣ σε ΛΣ «πρώτης γενιάς». Τα πρώτα συστήματα batch αυτοματοποίησαν την ακολουθία: φόρτωμα – μετάφραση – φόρτωμα – εκτέλεση, χρησιμοποιώντας ένα κεντρικό πρόγραμμα ελέγχου που ανακαλούσε και φόρτωνε «προγράμματα συστήματος» (system programms) (π.χ. assembler, compiler, φορτωτή ή κώδικες βιβλιοθήκης) και που αναλάμβανε τη μετάβαση από εργασία σε εργασία. Οι μεταφραστές γλωσσών ξαναγράφτηκαν, ώστε να παράγουν «μετατοπιζόμενο» (relocatable) κώδικα αντί για απόλυτο κώδικα. Οι «συνδετικοί φορτωτές» (linking loaders) άρχισαν να εμφανίζονται. Χάρη σε αυτούς, επιτράπηκε η ανάμειξη τμημάτων (ομάδων από κάρτες) «κώδικα πηγής» (source code) και τμημάτων μετατοπιζόμενου «κώδικα

μηχανής αντικειμένου» (object code). Επίσης, οι κώδικες βιβλιοθήκης μπορούσαν πια να αποθηκεύονται σε μορφή μετατοπιζόμενου κώδικα.

Σε εκείνα τα ΛΣ, τα θέματα προστασίας του συστήματος ήταν τα πιο δύσκολα προβλήματα. Ήταν σχετικά εύκολο για το σύστημα να καταστρέψει τον εαυτό του ή να καταστραφεί από κάποιο χρήστη ή ήταν πιθανό ένας χρήστης να «διαβάσει» πέρα από τη δική του εργασία, επεμβαίνοντας (ίσως άθελά του) στην επόμενη εργασία. Η κατανομή των πόρων (κύριας μνήμης και συσκευών Εισόδου/Εξόδου) ήταν δουλειά του χρήστη και όχι του ΛΣ.

Τρίτη γενιά 1959 – 1965

Γύρω στα 1959 – 63 μερικές σημαντικές ανακαλύψεις στα συστήματα υλικού έδωσαν ώθηση στην παραπέρα εξέλιξη των ΛΣ. Ίσως η πιο σημαντική hardware, εκείνη την εποχή, ήταν το λεγόμενο «κανάλι δεδομένων» (data channel), δηλαδή ένας «πρωτόγονος» Η/Υ και τα εργαλεία εισόδου/εξόδου. Μόλις το κανάλι πάρει μια «εντολή αίτησης για I/O» από τον επεξεργαστή (ΚΜΕ), το κανάλι αρχίζει να εκτελεί και να ελέγχει την I/O εργασία ασύγχρονα και παράλληλα με την (συνεχιζόμενη εν τω μεταξύ) εκτέλεση εντολών από την ΚΜΕ. Δηλαδή η επικάλυψη των εργασιών της ΚΜΕ και των συστημάτων I/O (χρονικά) είναι πια γεγονός. Το κανάλι και η ΚΜΕ μοιράζονται την κύρια μνήμη, που περιέχει προγράμματα και δεδομένα και για τους δύο. Αρχικά, μόνο η ΚΜΕ μπορούσε να ρωτήσει ποια είναι η κατάσταση του καναλιού ανά πάσα στιγμή. Αργότερα όμως έγινε φανερό ότι το όλο σύστημα θα εργαζόταν πιο αποδοτικά εάν το κανάλι μπορούσε να διακόψει την εργασία της ΚΜΕ για να παραδώσει ένα μήνυμα, που συνήθως ήταν η λήξη μιας I/O εργασίας.

Αμέσως άρχισαν να γράφονται πολύπλοκα συστήματα λογικού που μπορούσαν να εκμεταλλευτούν τις πιθανές χρήσεις της νέας αρχιτεκτονικής. Τα συστήματα αυτά περιλάμβαναν διαδικασίες «λογικής μόνωσης» (software buffering), που επέτρεπαν, για παράδειγμα, «στοίβαγμα» (queuing) αποτελεσμάτων λόγω καθυστέρησης γραψίματος της εισόδου κτλ. Ακόμα, περιλάμβαναν ρουτίνες «χειρισμού των σημάτων διακοπής» (interrupt handling), για να γίνεται δυνατή η διαδικασία της απάντησης (από την ΚΜΕ) σε περίπτωση I/O interrupt και η διαδικασία επιστροφής του ελέγχου στη διαδικασία που διακόπηκε, μετά την εξυπηρέτηση του σήματος διακοπής.

Τέταρτη γενιά 1965 – 1980

Η (φαινομενική) ασυμβατότητα ανάμεσα σε υπολογιστές για αριθμητικούς υπολογισμούς μεγάλης κλίμακας και σε εμπορικής χρήσης υπολογιστές αντιμετωπίστηκε πρώτα από την IBM με το σύστημα 360. Η γραμμή παραγωγής 360 ήταν μια σειρά από συμβατούς, όσον αφορά το λογισμικό, υπολογιστές. Το

λειτουργικό τους σύστημα (OS 360), παρ' ότι μεγάλο και δύσχρηστο, εντούτοις εισάγει σημαντικότατες έννοιες, που δεν υπάρχουν στη δεύτερη γενεά.

Μεταξύ αυτών, τα πλέον σημαντικά είναι αυτά του «πολυπρογραμματισμού» (multi-programming). Η ιδέα χρησιμοποιεί διαχωρισμούς της μνήμης σε διάφορα «μέρη» (partitions), έτσι ώστε διάφοροι υπολογισμοί (εργασίες) να εξυπηρετούνται «ταυτόχρονα». Με τον τρόπο αυτό κατέστη δυνατή η συνύπαρξη και η συνεκτέλεση αριθμητικών υπολογισμών μεγάλης κλίμακας και εμπορικών (εντατικών σε I/O) υπολογισμών για πρώτη φορά.

Άλλη σημαντική εξέλιξη των συστημάτων της τρίτης γενεάς υπήρξε η ικανότητα φόρτωσης πολλαπλών εργασιών (από κάρτες) στο σύστημα, ώστε να μην καθυστερεί η εκτέλεσή τους. Η ιδιότητα αυτή ονομάστηκε Spooling (από το Simultaneous Peripheral Operations on Line). Με τη χρήση του Spooling περιορίστηκε σοβαρά η χρήση μαγνητικών ταινιών.

Η ανάγκη για γρήγορο χρόνο απόκρισης οδήγησε στην έννοια της «διαμοίρασης χρόνου» (time sharing), μια παραλλαγή του πολυπρογραμματισμού, όπου ο κάθε χρήστης έχει το δικό του τερματικό και το κεντρικό σύστημα μπορεί να δώσει «άλλη λεπιδραστική» (interactive) εξυπηρέτηση σε όλους τους χρήστες χάρη στο μοίρασμα του χρόνου της CPU. Το πλέον επιτυχημένο time–sharing λειτουργικό σύστημα της εποχής αυτής υπήρξε το MULTICS (διάδοχος του συστήματος CTSS, από το MIT, τα Bell Labs και τη General Electric και τα Unix), ικανό να εξυπηρετήσει εκατοντάδες χρήστες ταυτόχρονα.

Οι ιδέες του MULTICS οδήγησαν το σχεδιαστή λειτουργικών συστημάτων Ken Thomson (των Bell Labs) στη δημιουργία του πρώτου UNIX συστήματος (μιας μικρής κλίμακας ειδικής των MULTICS για τη σειρά PDP). Το αρχικό του όνομα ήταν UNICS (Uniplexed Information and Computing Service), αλλά ο συν–σχεδιαστής του, ο B. Kernigham, το μετέτρεψε σε UNIX.

Στην ομάδα προσετέθη και άλλος επιστήμονας των Bell Labs, ο D. Ritchie, και το UNIX ξαναγράφτηκε στη νέα (τότε) γλώσσα C (που σχεδιάστηκε και υλοποιήθηκε από τον Ritchie). Τα εργαστήρια Bell έδωσαν άδεια χρήσης και ανάπτυξης του UNIX στα πανεπιστήμια δωρεάν και το σύστημα γρήγορα μετακόμισε σε πλειάδα αρχιτεκτονικών (π.χ. VAX, Motorola κτλ.). Η χρήση του ακόμα και σήμερα είναι ταχύτατα ανοδική.

Πέμπτη γενιά 1980 – 1990

Η τέταρτη γενεά λειτουργικών συστημάτων χαρακτηρίστηκε από την ταυτόχρονη εμφάνιση και κυριαρχία των προσωπικών υπολογιστών (PC's). Αυτή η πραγματικότητα οδήγησε στην εμφάνιση λειτουργικών συστημάτων «φιλικών προς το χρήστη» (user friendly) και ευέλικτων (π.χ. XENIX, DOS κτλ.) και αργότερα, λόγω της εμφάνισης τοπικών δικτύων, σε ευέλικτα λειτουργικά συστήματα με αντιλήψεις κατανεμημένου υπολογισμού και πολλαπλών καθηκόντων (multi – tasking OS).

Η σημερινή μορφή των Λ.Σ.

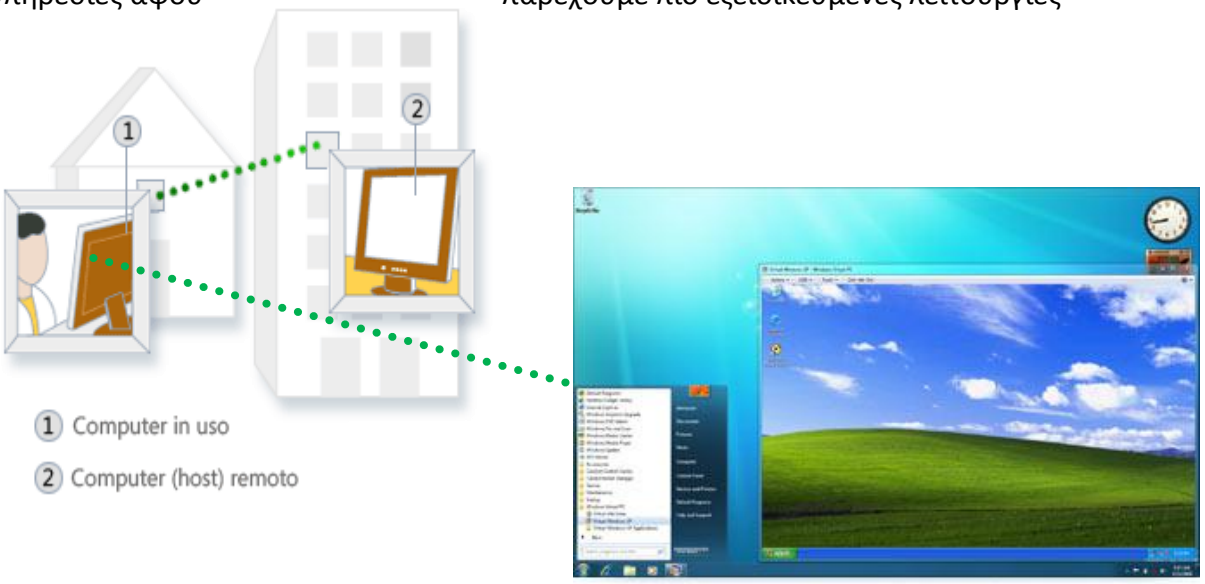
Η σημερινή πραγματικότητα χαρακτηρίζεται από την κυριαρχία του Διαδικτύου και του Παγκόσμιου Ιστού (Internet και Web) και την ύπαρξη γρήγορων τοπικών και μη τοπικών δικτύων. Τα σύγχρονα λειτουργικά συστήματα τροποποίησαν τις παλαιότερες αρχές σχεδιασμού, ώστε να ικανοποιούν τη διαχείριση των δικτυακών πόρων. Έτσι, έχουμε την κυρίαρχη σχεδιαστική αντίληψη των «processes» (διεργασιών ή διαδικασιών), καθώς και τη σχεδιαστική αρχή της επικοινωνίας των διεργασιών με τη μέθοδο «Client – Server» (Πελάτη – Εξυπηρετητή) και άλλες μεθόδους επικοινωνίας (π.χ. sockets), που συμβαδίζουν με τις τεχνολογικές εξελίξεις στα δίκτυα IP. Τα λειτουργικά συστήματα που κυριαρχούν χαρακτηρίζονται από την ανεξαρτησία τους από δεσμεύσεις του hardware, καθώς και από τη φιλικότητα προς το χρήστη και τις επικοινωνιακές τους ικανότητες. Στο επίπεδο των PC's έχουμε την κυριαρχία των Windows, ενώ στο υψηλότερο επίπεδο των multi – tasking συστημάτων εμφανίζονται οι μετεξελίξεις και οι οικογένειες των UNIX (π.χ. Solaris, POSIX κτλ.), καθώς και τα Windows NT για μικρότερα multi – tasking συστήματα. Σήμερα, οι περισσότεροι προσωπικοί υπολογιστές έχουν το λειτουργικό σύστημα ενός χρήστη Windows (π.χ. των ετών 1998 ή 2000). Αυτό το λειτουργικό σύστημα είναι ιδιαίτερα φιλικό προς τον τελικό χρήστη (άνθρωπο), ενώ είναι εφοδιασμένο με πολλά προγράμματα / διευκολύνσεις για σύνδεση με το Διαδίκτυο, με περιφερειακές συσκευές και άλλες διευκολύνσεις για τη χρήση πολυμέσων [ψηφιακών δίσκων (CDs), ψηφιακού video (DVD), ήχου και εικόνας]. Μολονότι το λειτουργικό αυτό σύστημα δεν είναι σύστημα πολλών διεργασιών από την πλευρά του χρήστη, εντούτοις διαχειρίζεται με περίπλοκο τρόπο τη μνήμη και την ποικιλία των τερματικών ή επικοινωνιακών διατάξεων που χρειάζονται σήμερα τα συστήματα εφαρμογών πολυμέσων.

Στις μηχανές «υψηλής απόδοσης» (υπερυπολογισμού – high end) υπάρχει αρκετή αβεβαιότητα για το πλέον κατάλληλο λειτουργικό σύστημα, ενώ το UNIX απλώνεται και προς την κατεύθυνση αυτή.

Είναι επίσης χαρακτηριστική η πρώτη εμφάνιση στοιχείων τεχνητής νοημοσύνης και βάσεων δεδομένων / γνώσεων στο σχεδιασμό τέτοιων συστημάτων, ενώ η αλληλεπίδρασή τους (interfaces) με το Web φαίνεται καθοριστικό στοιχείο για την αποδοχή τους. Οι κατανεμημένες εφαρμογές είναι πλέον πραγματικότητα. Αντίθετα, υπάρχουν ιδιαίτερες δυσκολίες στην αποδοχή λειτουργικών συστημάτων για ευρυζωνικά δίκτυα (WAN's), παρά τις αξιόλογες εξελίξεις στο θέμα της διαχείρισης πόρων των δικτύων (NMS).

Απομακρυσμένη διαχείριση Λ.Σ.(επίλογος 1ου κεφαλαίου)

Η χρήση τον υπολογιστών σε καθημερινές ανάγκες και προβλήματα σε συνάρτηση με την εξέλιξη της τεχνολογίας (τον λειτουργικών συστημάτων και τον δικτύων) μας οδήγησαν σε βελτίωση και τον υπηρεσιών που παρέχουν σε θέματα υποστήριξης. Στο επόμενο κεφάλαιο θα δούμε πώς με την χρήση δικτύων μπορούμε να βελτιώσουμε την χρήση των Η/Υ και να λύσουμε προβλήματα ακόμη και σε απόσταση εξοικονομώντας έτσι χρόνο και χρήμα, βελτιώνοντας παράλληλα και της υπηρεσίες αφού παρέχουμε πιο εξειδικευμένες λειτουργίες



Τι είναι δίκτυο

Δίκτυο είναι το σύνολο από υπολογιστές και συσκευές που ομοιάζουν με υπολογιστή, τα οποία επικοινωνούν μεταξύ τους μέσω ενός μέσου μεταφοράς των δεδομένων. Αυτό το μέσο μεταφοράς μπορεί να είναι καλώδιο, η τηλεφωνική μας γραμμή ή κάποιο άλλο π.χ. κάποιος ασύρματος μεταδότης.
Σε ένα δίκτυο ο κάθε υπολογιστής πρέπει να είναι ικανός να επικοινωνεί με τους υπόλοιπους και μάλιστα αμφίδρομα, δηλαδή να δύναται να διοχετεύσει πληροφορίες αλλά και να λάβει. Για να γίνει τώρα αυτό θα πρέπει να οριστούν εξ αρχής ο τρόπος επικοινωνίας του με το υπόλοιπο δίκτυο. Θα πρέπει δηλαδή να οριστούν κάποιοι κανόνες για την ασφαλή και απρόσκοπτη μεταφορά των δεδομένων.

Κατηγορίες δικτύων ανάλογα με την γεωγραφική κάλυψη

Τα τοπικά δίκτυα (LAN) είναι μικρά εταιρικά δίκτυα. Η κάλυψη που προσφέρουν δεν υπερβαίνει τα όρια ενός κτηρίου. Τα μητροπολιτικά δίκτυα (MAN) συνδέουν μικρότερα δίκτυα, καλύπτοντας ολόκληρες πόλεις. Τα δίκτυα ευρείας περιοχής (WAN) δεν υπόκεινται σε γεωγραφικούς περιορισμούς.
Τοπολογία είναι ο τρόπος σύνδεσης των υπολογιστών στο δίκτυο. Στα τοπικά δίκτυα, οι τρεις βασικές τοπολογίες είναι ο δίαυλος, ο αστέρας και ο δακτύλιος. Για να σχηματιστεί ένα δίκτυο απαιτούνται διάφορες συσκευές και εξαρτήματα: κάρτες δικτύου, καλωδίωση, διανομείς, διακόπτες, γέφυρες, πύλες, επαναλήπτες, δρομολογητές, firewalls. Σε ένα δίκτυο, υπεύθυνο για την απρόσκοπτη επικοινωνία των υπολογιστών είναι το λειτουργικό σύστημα δικτύου. Δημοφιλή λειτουργικά συστήματα είναι το Novell Netware, τα Windows NT/2000/XP και το UNIX.
Πρωτόκολλο επικοινωνίας είναι το σύνολο των κανόνων που ρυθμίζουν την ανταλλαγή δεδομένων στο δίκτυο. Το πρωτόκολλο επικοινωνίας ενσωματώνεται στο λειτουργικό σύστημα δικτύου. Το δημοφιλέστερο σήμερα πρωτόκολλο επικοινωνίας στους κανόνες του οποίου στηρίζεται το Διαδίκτυο, είναι το TCP/IP.

Τι είναι Διαδίκτυο

Διαδίκτυο είναι ένα δίκτυο ηλεκτρονικών υπολογιστών που (δια)συνδέει άλλα δίκτυα. Ο αντίστοιχος αγγλικός όρος internet προκύπτει από τη σύνθεση λέξεων inter-network. Στην πιο εξειδικευμένη και περισσότερο χρησιμοποιούμενη μορφή, με τους όρους Διαδίκτυο, Ιντερνέτ περιγράφεται το παγκόσμιο πλέγμα διασυνδεδεμένων υπολογιστών περιλαμβανομένων και των υπηρεσιών και πληροφοριών που παρέχει στους χρήστες του. Το Διαδίκτυο χρησιμοποιεί την μεταγωγική μετάδοση πακέτων (packet switching) και το πρωτόκολλο επικοινωνίας

TCP/IP. Έτσι ο όρος διαδίκτυο κατέληξε να αναφέρεται στο παγκόσμιο αυτό δίκτυο. Η τεχνική της σύνδεσης δικτύων με αυτό τον τρόπο ονομάζεται internet working.

Η ιστορία του διαδικτύου

Ο πυρήνας του Διαδικτύου ξεκίνησε το 1969 με την ονομασία ARPANET στις Ηνωμένες Πολιτείες στην Υπηρεσία Προηγμένων Αμυντικών Ερευνών (Defense Advanced Research Projects Agency, DARPA) του υπουργείου Άμυνας των ΗΠΑ. Η αρχική έρευνα που συνέβαλε στο ARPANET περιλάμβανε εργασίες στα αποκεντρωμένα δίκτυα, queueing theory και ανταλλαγή πακέτων packet switching. Το 1983 το ARPANET άλλαξε το βασικό του δικτυακό πρωτόκολλο επικοινωνίας από το NCP στο TCP/IP ξεκινώντας έτσι το Διαδίκτυο όπως το γνωρίζουμε σήμερα.

Ένα σημαντικό βήμα στην ανάπτυξη έκανε το Εθνικό Ίδρυμα Επιστημών (National Science Foundation, NSF) των ΗΠΑ το οποίο έχτισε την πανεπιστημιακή ραχοκοκαλιά, τοNSFNet, το 1986. Σημαντικά διαφορετικά δίκτυα που έχουν επιτυχώς ενσωματωθεί στο Διαδίκτυο είναι μεταξύ των άλλων το Usenet, τοFidonet και το Bitnet. Στη δεκαετία του1990 το Διαδίκτυο προσάρμοσε επιτυχώς την πλειοψηφία των παλιότερων δικτύων υπολογιστών.

Το διαδίκτυο σήμερα

Το Διαδίκτυο συγκροτείται από αμφίπλευρα ή πολύπλευρα εμπορικά συμβόλαια (π.χ. ομότιμες συμφωνίες) και από τεχνικές προδιαγραφές ή πρωτόκολλα που περιγράφουν την ανταλλαγή δεδομένων στο δίκτυο. Τα πρωτόκολλα αυτά μορφοποιούνται με συζητήσεις μέσα στο Internet Engineering Task Force (IETF) και τις ομάδες εργασίας του, οι οποίες είναι ανοιχτές για δημόσια συμμετοχή και κριτική. Αυτές οι επιτροπές παράγουν κείμενα που είναι γνωστά ως Αιτήματα για Σχολιασμό (ΑΓΣ). Ορισμένα ΑΓΣ εγείρονται από την κατάσταση του Προτύπου Διαδικτύου από το Συμβούλιο Αρχιτεκτονικής του Διαδικτύου (IAB).

Μερικά από τα πιο γνωστά πρωτόκολλα Internet protocol suite είναι το IP, TCP, το UDP, το DNS, το PPP, το SLIP, το ICMP, το POP3, IMAP, το SMTP, το HTTP, το HTTPS, το SSH, το Telnet, το FTP, το LDAP και το SSL. Μερικές από τις πιο γνωστές υπηρεσίες του internet που χρησιμοποιούν αυτά τα πρωτόκολλα είναι το ηλεκτρονικό ταχυδρομείο (e-mail), τα Usenet news groups, η διαμοίραση αρχείων (file sharing), ο Παγκόσμιος Ιστός (World Wide Web), το Gopher, το session access, το WAIS, το finger, το IRC, το MUD και το MUSH. Από αυτές, το ηλεκτρονικό ταχυδρομείο και ο Παγκόσμιος Ιστός είναι οι πιο ευρέως χρησιμοποιούμενες, ενώ πολλές άλλες υπηρεσίες έχουν βασιστεί πάνω σε αυτές, όπως ταχυδρομικές λίστες και αρχεία καταγραφής ιστού. Το Διαδίκτυο καθιστά δυνατή τη διάθεση υπηρεσιών

σε πραγματικό χρόνο όπως ραδιόφωνο μέσω Ιστού και προβλέψεις μέσω Ιστού που μπορούν να προσπελαστούν από οπουδήποτε στον κόσμο.

Το Ίντερνετ επίσης έχει μία μεγάλη επίδραση στην γνώση και τη διαμόρφωση απόψεων. Μέσα από την αναζήτηση λέξεων-κλειδιών (key words) μέσω της χρήσης μηχανών αναζήτησης, όπως το Google, εκατομμύρια άνθρωποι έχουν εύκολη και άμεση πρόσβαση σε ένα τεράστιο, παγκόσμιο και ποικίλο όγκο πληροφοριών. Συγκρινόμενο με τις έντυπες εγκυκλοπαίδειες και τις παραδοσιακές βιβλιοθήκες, το Ίντερνετ αντιπροσωπεύει μία ξαφνική και απότομη αποκέντρωση των πληροφοριών και των δεδομένων. Η γλώσσα που χρησιμοποιείται περισσότερο για την επικοινωνία στο Ίντερνετ είναι η Αγγλική κυρίως λόγω της καταγωγής του Ίντερνετ, της χρήσης της Αγγλικής στον προγραμματισμό λογισμικού και στην αδυναμία των πρώτων γενιών υπολογιστών να χρησιμοποιήσουν άλλους χαρακτήρες πέραν του λατινικού αλφάβητου. Το δίκτυο μεγάλωσε αρκετά τα τελευταία χρόνια και επαρκές περιεχόμενο είναι πλέον διαθέσιμο στις γλώσσες των περισσότερο ανεπτυγμένων χωρών

Ο Παγκόσμιος Ιστός (World Wide Web)

Το 1993, το εργαστήριο CERN στην Ελβετία παρουσιάζει τον Παγκόσμιο Ιστό (World Wide Web - WWW) του Tim Berners-Lee.[Lee00]. Πρόκειται για ένα σύστημα διασύνδεσης πληροφοριών multimedia και παρουσίασής τους σε ηλεκτρονικές σελίδες. Το γραφικό αυτό περιβάλλον κάνει την εξερεύνηση του Internet πιο προσιτή στον απλό χρήστη. Παράλληλα, εμφανίζονται διάφορα εμπορικά δίκτυα που ανήκουν σε εταιρίες παροχής υπηρεσιών Διαδικτύου (Internet Service Providers -ISP) και προσφέρουν πρόσβαση σε όλους. Οποιοσδήποτε διαθέτει PC και modem μπορεί να συνδεθείμετο Internet. Το 1995, το NSFnet καταργείται πλέον επίσημα και το φορτίο του μεταφέρεται σε εμπορικά δίκτυα.

Το Διαδίκτυο, από το 1995 και εφεξής, άρχισε να λαμβάνει τη μορφή με την οποία μας είναι γνωστό σήμερα. Πλέον το μεγαλύτερο μέρος του πληθυσμού του πλανήτη ζει σε χώρες συνδεδεμένες στο Internet. Καθημερινά άνθρωποι από όλες τις γωνιές της γης συνδέονται στο Internet προκειμένου να εργαστούν, να ενημερωθούν, να επικοινωνήσουν μεταξύ τους στέλνοντας ηλεκτρονική αλληλογραφία, να ανταλλάξουν αρχεία, να κάνουν τις αγορές τους ή απλά να διασκεδάσουν.

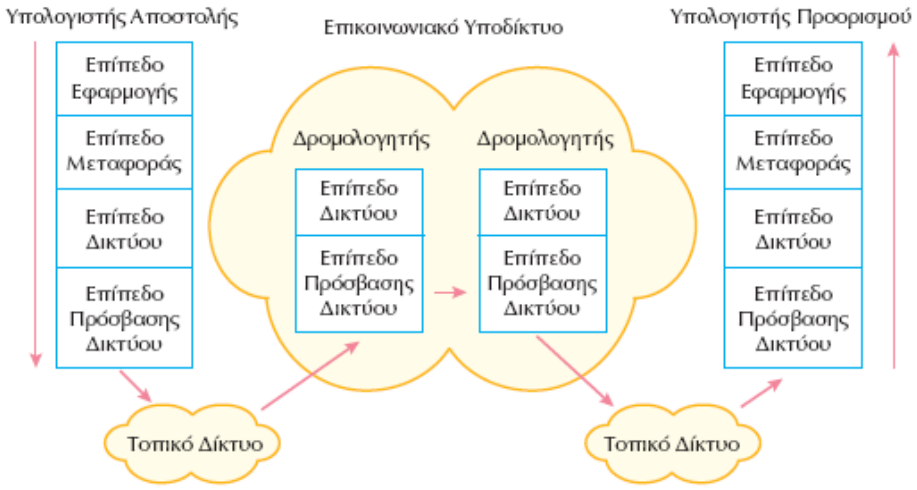
Το πρότυπο TCP/IP

Οι εφαρμογές, που βασίζονται στα πρωτόκολλα TCP/IP, χρησιμοποιούν τέσσερα επίπεδα

1. Πρωτόκολλα εφαρμογής
2. Πρωτόκολλο επιπέδου μεταφοράς (TCP/UDP)
3. Πρωτόκολλο επιπέδου δικτύου (IP)
4. Πρωτόκολλα για διαχείριση φυσικού μέσου (Ethernet)

Η τεχνολογία TCP/IP

- Βασίζεται σε ένα μοντέλο που θεωρεί ότι μεγάλος αριθμός δικτύων διασυνδέονται μεταξύ τους μέσω δρομολογητών.
- Τα πακέτα για να φτάσουν στο προορισμό τους περνάνε μέσα από μεγάλο αριθμό δικτύων.
- Η αποστολή δε γίνεται αντιληπτή από το χρήστη
- Ο χ΄ρηστης γνωρίζει μόνο IP ή όνομα.
- Έχει χτιστή με βάση τη τεχνολογία χωρίς σύνδεση
- Πληροφορία μεταφέρεται με τη μοπρφή παλκέτων



Πρωτόκολλο UDP

- Το **Πρωτόκολλο αυτοδύναμων πακέτων χρήστη** (**User Datagram Protocol, UDP)** είναι ένα από τα βασικά πρωτόκολλα που χρησιμοποιούνται στο Διαδίκτυο.
- Οι εφαρμογές audio και video streaming χρησιμοποιούν κατά κόρον πακέτα UDP. Για τις εφαρμογές αυτές είναι πολύ σημαντικό τα πακέτα να παραδοθούν στον παραλήπτη σε σύντομο χρονικό διάστημα ούτως ώστε να μην υπάρχει διακοπή στην ροή του ήχου ή της εικόνας. Κατά συνέπεια προτιμάται το πρωτόκολλο UDP διότι είναι αρκετά γρήγορο, παρόλο που υπάρχει η πιθανότητα μερικά πακέτα UDP να χαθούν.
- Σε αντίθεση με το πρωτόκολλο TCP, το UDP υποστηρίζει broadcasting, δηλαδή την αποστολή ενός πακέτου σε όλους τους υπολογιστές ενός δικτύου, και

multicasting, δηλαδή την αποστολή ενός πακέτου σε κάποιους συγκεκριμένους υπολογιστές ενός δικτύου.

- Το UDP βρίσκεται ανάμεσα στο επίπεδο δικτύου (network layer) και στο επίπεδο εφαρμογών (application layer)
- Κάθε πακέτο UDP έχει μία κεφαλίδα (header) που αναφέρει τα χαρακτηριστικά του.
- Το UDP είναι ένα πιο απλό και ελαφρύ πρωτόκολλο, στο οποίο δεν υπάρχει η έννοια της σύνδεσης. Κάθε πακέτο UDP διανύει το δίκτυο σαν μία ξεχωριστή αυτόνομη μονάδα και όχι σαν μία σειρά πακέτων σε μία σύνδεση, όπως στο TCP.

Το σύστημα Client – Server

Τι είναι το client-server computing;

Γενικά, το client-server computing αναφέρεται σε μια βασική αλλαγή στο στυλ των υπολογιστών, την αλλαγή από τα συστήματα που βασίζονται στα μηχανήματα στα συστήματα που βασίζονται στον χρήστη.

Ειδικότερα, ένα σύστημα client-server είναι ένα σύστημα στο οποίο το δίκτυο ενώνει διάφορους υπολογιστικούς πόρους, ώστε οι clients (ή αλλιώς front end) να μπορούν να ζητούν υπηρεσίες από έναν server (ή αλλιώς back end), ο οποίος προσφέρει πληροφορίες ή επιπρόσθετη υπολογιστική ισχύ.

Με άλλα λόγια, στο client-server μοντέλο, ο client θέτει μια αίτηση και ο server επιστρέφει μια ανταπόκριση ή κάνει μια σειρά από ενέργειες. Ο server μπορεί να ενεργοποιείται άμεσα για την αίτηση αυτή ή να προσθέτει την αίτηση σε μια ουρά. Η άμεση ενεργοποίηση για την αίτηση μπορεί, για παράδειγμα, να σημαίνει ότι ο server υπολογίζει έναν αριθμό και τον επιστρέφει αμέσως στον client. Η τοποθέτηση της αίτησης σε μια ουρά μπορεί να σημαίνει ότι η αίτηση πρέπει να τεθεί σε αναμονή για να εξυπηρετηθεί. Ένα καλό παράδειγμα για αυτό είναι όταν εκτυπώνουμε ένα κείμενο σε ένα εκτυπωτή δικτύου. Ο server τοποθετεί την αίτηση σε μια ουρά μαζί με αιτήσεις εκτυπώσεων και από άλλους clients. Μετά επεξεργάζεται την αίτηση με βάση την σειρά προτεραιότητας, η οποία , σε αυτή την περίπτωση, καθορίζεται από τη σειρά με την οποία ο server παρέλαβε την απαίτηση.

Το client-server computing είναι πολύ σημαντικό, διότι επιτυγχάνει τα εξής:

- ✓ Αποτελεσματική χρήση της υπολογιστικής ισχύος.
- ✓ Μείωση του κόστους συντήρησης, δημιουργώντας συστήματα client-server που απαιτούν λιγότερη συντήρηση και κοστίζουν λιγότερο στην αναβάθμιση.
- ✓ Αύξηση της παραγωγικότητας, προσφέροντας στους χρήστες ξεκάθαρη πρόσβαση στις αναγκαίες πληροφορίες μέσω σταθερών και εύκολων στην χρήση διασυνδέσεων.
- ✓ Αύξηση της ευελιξίας και της δυνατότητας δημιουργίας συστημάτων που υποστηρίζουν πολλά περιβάλλοντα.

Τι είναι Client

Ο client είναι ο αιτών των υπηρεσιών. Ο client δεν μπορεί παρά να είναι ένας υπολογιστής. Οι υπηρεσίες που ζητούνται από τον client μπορεί να υπάρχουν στους ίδιους σταθμούς εργασίας ή σε απομακρυσμένους σταθμούς εργασίας που συνδέονται μεταξύ τους μέσω ενός δικτύου. Ο client ξεκινάει πάντα την επικοινωνία.

Τα συστατικά του client είναι πολύ απλά. Μια client μηχανή πρέπει να μπορεί να κάνει τα ακόλουθα:

- ✓ Να τρέχει το λογισμικό των γραφικών διεπαφών χρηστών (GUIs).
- ✓ Να δημιουργεί τις αιτήσεις για πληροφορίες και να τις στέλνει στον server.
- ✓ Να αποθηκεύει τις επιστρεφόμενες πληροφορίες.

Αυτές οι αιτήσεις καθορίζουν πόση μνήμη χρειάζεται, ποια ταχύτητα επεξεργασίας θα μπορούσε να βελτιώσει τον χρόνο ανταπόκρισης, και πόση χωρητικότητα αποθήκευσης απαιτείται.

Τι είναι Server

Ο server απαντάει στις αιτήσεις που γίνονται από τους clients. Ένας client μπορεί να ενεργεί ως server εάν λαμβάνει και επεξεργάζεται αιτήσεις όπως ακριβώς και τις στέλνει (για παράδειγμα, ένας σταθμός εργασίας που χρησιμοποιείται και ως server εκτυπώσεων από άλλους). Οι server δεν ξεκινάνε τις επικοινωνίες -περιμένουν τις αιτήσεις των clients.

Επιστρέφοντας στο παράδειγμα του server εκτυπώσεων ενός δικτύου, ο client ζητάει από τον server να εκτυπώσει ένα κείμενο σε έναν συγκεκριμένο εκτυπωτή και ο server προσθέτει την εκτύπωση σε μια ουρά και ενημερώνει τον client όταν το κείμενο εκτυπωθεί επιτυχημένα. Η διαδικασία του client μπορεί να ανήκει φυσικά στον ίδιο σταθμό εργασίας με την διαδικασία του server. Στο παράδειγμα εδώ, μια εντολή εκτύπωσης μπορεί να εκδίδεται στον server του σταθμού εργασίας του δικτύου, χρησιμοποιώντας την διαδικασία του server εκτυπώσεωνσε αυτόν τον σταθμό εργασίας.

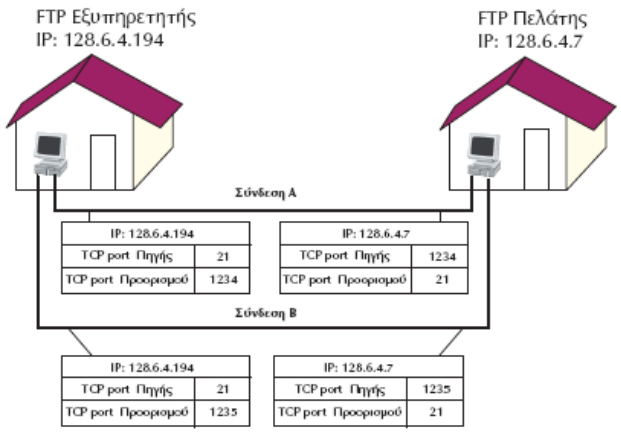
Τα συστατικά του server είναι πολύ απλά. Μια server μηχανή πρέπει να μπορεί να κάνει τα ακόλουθα :

- ✓ Να αποθηκεύει, να ανακτά και να προστατεύει πληροφορίες.
- ✓ Να επιθεωρεί τις αιτήσεις των clients.
- ✓ Να δημιουργεί εφαρμογές διαχείρισης πληροφοριών, όπως δημιουργία αντιγράφων, ασφάλεια κτλ.
- ✓ Να διαχειρίζεται πληροφορίες.

Πώς αναπτύχθηκε η client-server τεχνολογία;

Η τεχνολογία των υπολογιστών αναπτύχθηκε βαθμιαία, με τέτοιο τρόπο που κάθε καινούργια αρχιτεκτονική έπαιρνε τα πλεονεκτήματα από τις τεχνικές που ήδη υπήρχαν, ώστε να εκμεταλλεύεται όλες τις δυνατότητες των υπολογιστών. Σήμερα οι υπολογιστές είναι μικρότεροι, γρηγορότεροι και φθηνότεροι από ότι παλιότερα. Σαν αποτέλεσμα, η γενική κατεύθυνση είναι η διανομή της επεξεργασίας της πληροφορίας αλλά και της ίδιας της πληροφορίας σε ένα πλήθος αυτών των νέων υπολογιστών.

Ο όρος αρχιτεκτονική συνήθως χρησιμοποιείται για να περιγράφει συστήματα διαχείρισης βάσεων δεδομένων, λειτουργικά συστήματα και άλλους υπολογιστικούς μηχανισμούς λογισμικού και υλικού. Οι αρχιτεκτονικές περιγράφουν πως οι συσκευές και τα λογισμικά πακέτα ταιριάζουν για να φτιάξουν είναι εύκολο στην χρήση και στην διαχείριση σύνολο.



Η κλασσική αρχιτεκτονική αποτελείται από έναν υπολογιστή μεγάλης ισχύος, (που παίζει το ρόλο του οικοδεσπότη) με ένα ή περισσότερα απλά τερματικά. Οι εφαρμογές ελέγχονται και διανέμονται από τον υπολογιστή-«οικοδεσπότη». Σε αυτόν πραγματοποιούνται όλες οι διαχειρίσεις πληροφοριών, η λογική των εφαρμογών και η μορφοποίηση της εμφάνισης τους. Οι χρήστες αλληλεπιδρούν με το κεντρικό σύστημα μέσω των τερματικών, τα οποία εμφανίζουν μόνο πληροφορίες. Αυτή είναι η πιο συνηθισμένη αρχιτεκτονική σήμερα.

Ένα καλά οργανωμένο σύστημα που χρησιμοποιεί αυτήν την κλασσική αρχιτεκτονική προσφέρει τις ακόλουθες δυνατότητες:

- ✓ Ένα υψηλό επίπεδο αξιοπιστίας .
- ✓ Κεντρικό έλεγχο και κεντρική διαχείριση των πληροφοριών.
- ✓ Ισχυρή διαχείριση των πληροφοριών και δυνατότητα αποθηκεύσεων .

Πάντως, οι κλασικές εφαρμογές περιορίζουν την ευελιξία των τελικών χρηστών. Η διασύνδεση των χρηστών δεν είναι γραφική, κάτι που κάνει το σύστημα δυσκολότερο στη χρήση και σημαίνει ότι ο χρήστης πρέπει να μάθει πως να χρησιμοποιήσει την γλώσσα του οικοδεσπότη. Επίσης, οι εφαρμογές εξαρτώνται από μια πλατφόρμα, που σημαίνει ότι εάν κάτι συμβεί στον υπολογιστή-«οικοδεσπότη», ο χρήστης δεν μπορεί να χρησιμοποιήσει το σύστημα, έως ότου το σύστημα αρχίσει να επαναλειτουργεί.

Στην client-server αρχιτεκτονική, η client εφαρμογή τρέχει σε έναν πλήρη σταθμό εργασίας. Αυτός ο σταθμός μπορεί να είναι ένας προσωπικός υπολογιστής, ένας UNIX σταθμός εργασίας ή ένας Mac. Η client εφαρμογή βασίζεται στις υπηρεσίες που προσφέρει ο server και επικοινωνούν μέσω πρωτοκόλλων, όπως το πρωτόκολλο του Internet (TCP/IP) ή του Novell (IPX/SPX).

Το περιβάλλον του client-server έχει πολλά πλεονεκτήματα σε σχέση με τις κλασσικές αρχιτεκτονικές. Η διαχείριση της διασύνδεσης των χρηστών και άλλες επεξεργασίες είναι αποφορτισμένα από τον «οικοδεσπότη», ενώ ο server ακόμη προσφέρει συγκεντρωμένο έλεγχο των κοινών πόρων. Επειδή ο client επικοινωνεί με τον server μέσω ενός καθορισμένου συστήματος διασύνδεσης, δεν χρειάζεται να γνωρίζει που ανήκει ο server ή πως ενεργεί. Ο σταθμός εργασίας τρέχει την εφαρμογή και εμφανίζει τις πληροφορίες στον χρήστη. Μόνο όταν ο client προσπελάζει πληροφορίες, τότε εγκαθίσταται επικοινωνία με τον server. Ο φόρτος εργασίας μειώνεται δραματικά στον υπολογιστή-«οικοδεσπότη» όσο αυξάνεται η ισχύς κάθε σταθμού εργασίας.

Οι οργανισμοί έχουν να κάνουν με συνεχώς περισσότερα δεδομένα, τα οποία πρέπει να τα διαχειρίζονται και να τα εκμεταλλεύονται στις εργασίες τους. Η αύξηση του όγκου των δεδομένων, σε συνδυασμό με την προσπάθεια των οργανισμών να μειώσουν το κόστος, να αυξήσουν την παραγωγικότητα και να βελτιώσουν τις υπηρεσίες των πελατών (με καλύτερη χρήση πληροφοριών και ταχύτερο χρόνο ανταπόκρισης στους πελάτες ταυτόχρονα), έχουν συμβάλει σε μια ώθηση για δημιουργία και χρήση client-server εφαρμογών.

Σε συνδυασμό με τις μεταβαλλόμενες απαιτήσεις των επιχειρήσεων, η ανάπτυξη της τεχνολογίας των client-server έχουν οδηγήσει στα ακόλουθα:

- ✓ Πρόοδο στο υλικό.
- ✓ Πρόοδο στο λογισμικό.
- ✓ Πρόοδο στο δίκτυο.

Τι είναι αντικειμενοστραφής προγραμματισμός

Ο αντικειμενοστραφής προγραμματισμός (oop, object-oriented programming) είναι ένα μοντέλο γλώσσας προγραμματισμού οργανωμένο περισσότερο γύρω από τα δεδομένα παρά γύρω από τις ενέργειες και τη λογική του προγραμματισμού. εμφανίστηκε το 1960 στην επιστήμη των υπολογιστών, καθιερώθηκε το 1990 και αντικατέστησε σε μεγάλο βαθμό το δομημένο προγραμματισμό. Είναι ένα πρακτικό αποτέλεσμα της προσπάθειας να προσομοιωθούν πολύπλοκες δραστηριότητες του ανθρώπινου μυαλού, αντιμετωπίζοντας τα δεδομένα ως αντικείμενα όπως συμβαίνει δηλαδή και στον πραγματικό κόσμο.

Παραδοσιακά, ένα πρόγραμμα αντιμετωπίζεται ως μια λογική διαδικασία που λαμβάνει δεδομένα εισόδου, τα επεξεργάζεται και παράγει δεδομένα εξόδου. Η πρόκληση του προγραμματισμού ήταν το πώς θα αποτυπωθεί η λογική και όχι το πώς θα οριστούν τα δεδομένα. Αντίθετα ο αντικειμενοστραφής προγραμματισμός εστιάζει στα αντικείμενα που χρειάζεται να διαχειριστεί και όχι στη λογική της διαχείρισης τους.

Ιστορική αναδρομή

Η πρώτη γλώσσα προγραμματισμού που περιείχε αντικειμενοστραφής έννοιες ήταν η γλώσσα προγραμματισμού Simula 67, από την οποία επηρεάστηκε τη δεκαετία του 70’ η ανάπτυξη της γλώσσας προγραμματισμού Smalltalk που εισήγαγε τον όρο αντικειμενοστραφής προγραμματισμός. Η Smalltalk αναπτύχθηκε από τον Άλαν Κέι της εταιρείας Xerox στο πλαίσιο μίας εργασίας με στόχο τη δημιουργία ενός χρήσιμου, αλλά και εύχρηστου, προσωπικού υπολογιστή. Το 1980 η τελική έκδοση της Smaltak έγινε διαθέσιμη σηματοδοτώντας ότι η έρευνα για την αντικατάσταση του δομημένου προγραμματισμού με ένα πιο σύγχρονο υπόδειγμα απέδιδε αποτελέσματα. Στη γλώσσα αυτή καταργήθηκαν οι παραδοσιακές δομές δεδομένων από τις κλάσεις και έτσι όλα ήταν αντικείμενα.

Περίπου την ίδια εποχή και επίσης με επιρροές από την Simula, ολοκληρωνόταν η ανάπτυξη της C++ που ήταν η επέκταση της γλώσσας προγραμματισμού C με αντικειμενοστραφή χαρακτηριστικά. Με την επίδραση της C++ κατά την δεκαετία του 80’ έβγαλαν πολλές γλώσσες της εποχής εκείνης αντικειμενοστρεφής εκδόσεις. Το προγραμματιστικό υπόδειγμα του

αντικειμενοστραφή προγραμματισμού επικράτησε το πρώτο ήμισυ της δεκαετίας του 1990 επειδή ήταν καταλληλότερο στην ανάπτυξη της τότε καθιέρωσης των γραφικών διασυνδέσεων χρήστη στους μικροϋπολογιστές. Το 1995 εμφανίζεται η Java ως μια πλήρως αντικειμενοστραφή γλώσσα που έμοιαζε στο συντακτικό της με την C++, προσέφερε πρωτοποριακές δυνατότητες για την εποχή και έδωσε νέα ώθηση στον αντικειμενοστραφή προγραμματισμό. Ταυτόχρονα εμφανίστηκαν ποικίλες άτυπες βελτιώσεις στο βασικό προγραμματιστικό υπόδειγμα, όπως οι αντικειμενοστραφείς γλώσσες μοντελοποίησης λογισμικού και τα σχεδιαστικά πρότυπα. Το 2001 η Microsoft εστίασε την προσοχή της στην πλατφόρμα .NET, μία ανταγωνιστική της Java πλατφόρμα ανάπτυξης και εκτέλεσης λογισμικού η οποία ήταν εξολοκλήρου προσανατολισμένη στην αντικειμενοστρέφεια.

Αυτή τη στιγμή η C++ και η Java είναι οι πιο δημοφιλείς αντικειμενοστραφείς γλώσσες. Η Java συγκεκριμένα έχει σχεδιαστεί ειδικά για χρήση σε κατανεμημένες εφαρμογές μέσω δικτύου και διαδικτύου.

Το Object Model

Το Object Model που χρησιμοποιείται στη C# είναι παρόμοιο με άλλες αντικειμενοστραφείς γλώσσες (Object Oriented Languages). Η αντικειμενοστρέφεια είναι νεώτερη από το Δομημένο Προγραμματισμό, αλλά είναι πλέον πολύ σημαντική για τους προγραμματιστές σήμερα. Ο Object Oriented προγραμματισμός παρέχει μια τυποποιημένη γλώσσα και ένα πλαίσιο που είναι σημαντικό για να κάνει την εργασία τμημάτων λογισμικού (objects) ενωμένα μαζί σε ένα σύστημα. Οι Object Oriented έννοιες που μαθαίνονται για τη Visual Basic ισχύουν για τις περισσότερες γλώσσες που ένας προγραμματιστής είναι πιθανό να αντιμετωπίσει σήμερα. Μια γλώσσα βασισμένη σε αντικείμενα έχει τις περισσότερες από τις ιδιότητες μιας αντικειμενοστραφούς γλώσσας, αλλά μπορεί να στερείται μερικών. Παραδείγματος χάριν η Visual Basic δεν έχει κληρονομικότητα, ενώ μια γλώσσα βασισμένη σε πρωτότυπα στηρίζεται σε αυτά αντί στις κλάσεις για να δημιουργήσει τα αντικείμενα.

Η αντικειμενοστρέφεια (Object Orientation) έχει αλλάξει τον τρόπο που οι βιομηχανίες λογισμικού προσεγγίζουν την ανάπτυξη λογισμικού, και ιδιαίτερα, πώς οι τεχνικοί προσεγγίζουν την ανάλυση, το σχεδιασμό, την επαναχρησιμοποίηση κώδικα, και την υλοποίηση.

Ο Object Oriented προγραμματισμός είναι μια προσέγγιση (όχι ένα συγκεκριμένο εργαλείο) που οργανώνεται γύρω από τα αντικείμενα παρά από τις εφαρμογές, τα δεδομένα ή τη λογική.

Object

Ως αντικειμενοστραφής γλώσσα η C# , οργανώνεται γύρω από τα αντικείμενα. Τα αντικείμενα είναι προγραμματισμένοι κατασκευαστές οι οποίοι έχουν δεδομένα, ταυτότητα και συμπεριφορά. Τα δεδομένα των αντικειμένων περιλαμβάνονται στα πεδία τους, στις ιδιότητες και στα γεγονότα (events), ενώ οι συμπεριφορές των αντικείμενων ορίζονται από τις μεθόδους και το interface τους. Τα αντικείμενα έχουν ταυτότητα δηλαδή δύο αντικείμενα με το ίδιο set δεδομένων δεν είναι απαραίτητα το ίδιο αντικείμενο.

Τα αντικείμενα έχουν τις εξής ιδιότητες:
•Οτιδήποτε και αν χρησιμοποιείται στην C# είναι αντικείμενο, ακόμα και οι φόρμες των Windows και τα controls.
•Τα αντικείμενα δημιουργούνται από τα templates που ορίζονται από κλάσεις και structs.
•Τα αντικείμενα χρησιμοποιούν τις Ιδιότητές τους για να λειτουργήσουν και να αλλάξουν τις πληροφορίες τις οποίες εμπεριέχουν.
•Συχνά τα αντικείμενα έχουν μεθόδους και γεγονότα τα οποία τους επιτρέπουν να κάνουν δραστηριότητες (actions).
•Όλα τα αντικείμενα στην C# ιεραρχοποιούνται από το Αντικείμενο.

Βλέποντας την ανάλυση και το σχεδιασμό του λογισμικού ως αντικείμενα είναι πολύ σημαντικό επειδή και οι άνθρωποι σκέφτονται σε σχέση με τα αντικείμενα. Όλα γύρω μας είναι αντικείμενα.

Classes (Κλάσεις)

Μια κλάση είναι ένα σύνολο παρόμοιων ενεργειών και ιδιοτήτων, και ένα αντικείμενο είναι ένα instance (μια περίπτωση) μιας κλάσης. Μπορούμε να σκεφτούμε μια κλάση ως σχεδιασμό ενός αντικειμένου που θα δημιουργηθεί στον κώδικά μας. Και οι κλάσεις και τα αντικείμενα έχουν ταυτότητα. Μια κλάση καθορίζει τις ιδιότητες και τις μεθόδους, ενώ ένα αντικείμενο τις χρησιμοποιεί.

Μια κλάση καθορίζει τα στοιχεία των δεδομένων και τη μορφή των δεδομένων που συνδυάζονται για να απεικονίσουν το τι ένα αντικείμενο «γνωρίζει». Παραδείγματος χάριν, μπορούμε να δημιουργήσουμε ένα αντικείμενο αποκαλούμενο 'Oscar' από μια κλάση που ονομάζεται Dog. Η κλάση Dog καθορίζει τι πρόκειται να είναι το αντικείμενο 'Oscar', και όλα τα «dog-related» μηνύματα που ένα αντικείμενο Oscar μπορεί να ενεργήσει επάνω τους. Όλες οι αντικειμενοστραφείς γλώσσες έχουν κάποιες έννοιες. Καλούν μερικές φορές ένα «εργοστάσιο», να κατασκευάσει τα instances ενός αντικειμένου από τον καθορισμό μιας κλάσης. Μπορούμε να κάνουμε περισσότερα από ένα αντικείμενα αυτής της κλάση, και να τα καλέσουμε Spot, Fido, Rover κ.λπ. Η κλάση Dog καθορίζει τα

μηνύματα που τα αντικείμενα Dog καταλαβαίνουν, όπως «bark», «fetch», και «roll-over».

Οι κλάσεις και τα αντικείμενα έχουν χαρακτηριστικά. Παραδείγματα χαρακτηριστικών τους για ένα άτομο είναι η ηλικία, το ύψος, το βάρος, και δείκτης νοημοσύνης. Αυτά τα χαρακτηριστικά αναφέρονται ως χαρακτηριστικά κλάσης και ιδιότητες αντικειμένου. Τα χαρακτηριστικά της κλάσης και οι ιδιότητες του αντικειμένου είναι επίσης γνωστές ως member fields. Επειδή ένα αντικείμενο έχει τη δυνατότητα να θέσει τις ιδιότητες, τα αντικείμενα λέγονται επίσης ότι έχουν την κυριότητα. Οι τρέχουσες τιμές των χαρακτηριστικών ενός αντικειμένου είναι η επικρατούσα κατάστασή τους.

Properties (Ιδιότητες)

Τα Properties (ιδιότητες) λένε για το αντικείμενο, πληροφορίες όπως είναι το όνομά του, η θέση του στον υπολογιστή, εάν είναι ορατό, ενεργό, ή το χρώμα του. Οι ιδιότητες είναι όπως τα επίθετα που περιγράφουν τα αντικείμενα (ουσιαστικά). Οι ιδιότητες στην .NET δηλώνονται με τη χρησιμοποίηση του ονόματος αντικειμένου, μιας περιόδου (.), και την επιθυμητή ιδιότητα. Για παράδειγμα αν θέλουμε να δηλώσουμε ότι ένα αντικείμενο είναι ορατό δηλαδή Visible, τότε θα χρησιμοποιήσουμε public namespase Visible { get; set; } και οι τιμές που παίρνει είναι true ή false. Η σημαντικότερη ιδιότητα κάθε αντικειμένου είναι γενικά το όνομά της. Το όνομα είναι αυτό που δένει το αντικείμενο σε ένα script και το πώς ένα αντικείμενο δηλώνεται από οποιοδήποτε άλλο αντικείμενο. Η κωδικοποίηση της κλάσης πρέπει να λαμβάνεται υπόψη κατά την ονομασία των αντικειμένων.

Methods (Μέθοδοι)

Οι κλάσεις και τα αντικείμενα έχουν ενέργειες. Αυτές οι ενέργειες αναφέρονται ως λειτουργίες κλάσης και μέθοδοι αντικειμένου. Τις αναφέρουμε γενικά ως μεθόδους (methods). Μέθοδος είναι το πώς ο κώδικας μπορεί να χρησιμοποιήσει ένα αντικείμενο κάποιας κλάσης. Οι μέθοδοι μπορούν να διαιρεθούν σε queries (ερωτήσεις) επιστρέφοντας την επικρατούσα κατάσταση και σε commands (εντολές) που την αλλάζουν (υπορουτίνα). Μερικές φορές η πρόσβαση στα δεδομένα ενός αντικειμένου είναι περιορισμένη στις μεθόδους της κλάσης του. Μια μέθοδος είναι ουσιαστικά η εφαρμογή μιας υπηρεσίας αντικειμένου η οποία είναι απλά η δράση που ένα μήνυμα μεταφέρει. Είναι ο κώδικας, ο οποίος εκτελείται όταν το μήνυμα στέλνεται σε ένα συγκεκριμένο αντικείμενο. Ορίσματα παρέχονται συχνά ως τμήμα ενός μηνύματος. Στον Object Oriented προγραμματισμό στέλνουμε ένα μήνυμα από ένα αντικείμενο σε ένα άλλο. Οι μέθοδοι και οι ιδιότητες είναι

κλήσεις μηνυμάτων. Οι παράμετροι μιας μεθόδου είναι το περιεχόμενο των μηνυμάτων.

Events (Γεγονότα)

Τα γεγονότα (Events) είναι το πως τα αντικείμενα πρέπει να ανταποκρίνονται. Μπορούν να είναι ερεθίσματα για μεθόδους αντικειμένων και αναφέρονται ως Object.Events. Για παράδειγμα όταν σε μία εφαρμογή ο χρήστης θα πρέπει να πατήσει ένα κουμπί τότε ενεργοποιείτε το γεγονός (event) OnClick();.Άλλα γεγονότα είναι παραδείγματος χάριν:
event EventHandler DoubleClick
event EventHandler MouseEnter
event MouseEventHandler MouseMove
και φυσικά πολλά πολλά άλλα.

Inheritance (Κληρονομικότητα)

Η αρχή μιας κλάσης καθιστά πιθανό να καθοριστούν οι υποκλάσεις που μοιράζουν μερικά ή όλα τα κύρια χαρακτηριστικά της κλάσης. Αυτό καλείται κληρονομικότητα (inheritance). Η κληρονομικότητα μας επιτρέπει επίσης να επαναχρησιμοποιήσουμε τον κώδικα αποτελεσματικότερα. Ένας μηχανισμός για τις υποκλάσεις και παρέχει έναν τρόπο να καθοριστεί μια υποκλάση ως εξειδίκευση ή υποκατηγορία ή επέκταση μιας γενικότερης κλάσης. Μια υποκλάση κληρονομεί όλα τα μέλη (members) των superclass (υπερκλάσεων) της, αλλά μπορεί να επεκτείνει τη «συμπεριφορά» τους και να προσθέσει τα νέα μέλη.

Encapsulation (ενθυλάκωση)

Το encapsulation (ενθυλάκωση) είναι το κρύψιμο των στοιχείων και του κώδικα και καλείται συχνά ως «black box» προσέγγιση, δεδομένου ότι οι χρήστες μιας κλάσης δεν μπορούν να δουν μέσα στην κλάση (μπορούν να δουν μόνο το public interface της). Εξασφαλίζει επίσης ότι ο κώδικας εξωτερικά από μια κλάση βλέπει μόνο τις λειτουργικές λεπτομέρειες εκείνης της κλάσης, αλλά όχι τις λεπτομέρειες της υλοποίησης. Το Encapsulation επιτυγχάνεται διευκρινίζοντας ποιες κλάσεις μπορούν να χρησιμοποιήσουν τα μέλη ενός αντικειμένου. Το αποτέλεσμα είναι ότι κάθε αντικείμενο εκθέτει σε οποιαδήποτε κλάση ένα συγκεκριμένο interface. Τα μέλη καθορίζονται συχνά ως public, protected και private. Αυτό καθορίζεται από το εάν είναι διαθέσιμα σε όλες τις κλάσεις, τις υποκλάσεις ή μόνο την καθορισμένη κλάση. Μερικές γλώσσες επεκτείνονται περαιτέρω: Η Java χρησιμοποιεί protected λέξη κλειδί για να περιορίσει την πρόσβαση. Η C# και η VB.NET υποκαθιστούν μερικά μέλη σε κλάσεις χρησιμοποιώντας τις λέξεις κλειδιά ως internal (C#) ή ως Friend (VB.NET). Τα αντικείμενα αλληλεπιδρούν το ένα με το άλλο μέσω μηνυμάτων. Το μόνο πράγμα που ένα αντικείμενο γνωρίζει για ένα άλλο είναι το

interface του αντικειμένου. Τα δεδομένα και η λογική κάθε αντικειμένου είναι κρυμμένα από άλλα αντικείμενα. Αυτό επιτρέπει στον αναλυτή να χωρίσει την υλοποίηση ενός αντικειμένου από το interface του. Εφ' όσον το interfaceπαραμένει ίδιο, οποιεσδήποτε αλλαγές στην εσωτερική υλοποίηση είναι προφανείς στο χρήστη.

Polymorphism (Πολυμορφισμός)

Ένα άλλο όφελος που προκύπτει από το διαχωρισμό της υλοποίησης από τη συμπεριφορά είναι ο πολυμορφισμός (polymorphism). Ο πολυμορφισμός είναι συμπεριφορά που ποικίλλει ανάλογα με την κλάση στην οποία η συμπεριφορά επιδρά, δηλαδή δύο ή περισσότερες κλάσεις μπορούν να αντιδράσουν διαφορετικά στο ίδιο μήνυμα. Ο πολυμορφισμός επιτρέπει σε δύο ή περισσότερα αντικείμενα για να αποκριθούν στο ίδιο μήνυμα. Με άλλα λόγια, ο πολυμορφισμός επιτρέπει σε οποιαδήποτε απόγονο κλάση για να επαναπροσδιορίσει οποιαδήποτε μέθοδο που κληρονομείται από την κλάση γονέων του. Η επίδραση είναι ότι ο πολυμορφισμός επιτρέπει σε ένα σταλμένο αντικείμενο να επικοινωνήσει με τα διαφορετικά αντικείμενα κατά σύμφωνο τρόπο χωρίς ανησυχία για το πόσες διαφορετικές υλοποιήσεις ενός μηνύματος υπάρχουν.

JAVA

Η Java είναι μια νέα γλωσσά προγραμματισμού, η οποία σε αντίθεση με άλλες γλώσσες γνωστές ήδη από δεκαετίες, εμπεριέχει όλα τα χαρακτηριστικά της εξέλιξης της επιστήμης των υπολογιστών και είναι ιδιαίτερα γνωστή όχι μόνο από τους υπολογιστές αλλά και από τα κινητά και πολλές άλλες συσκευές πολυμέσων καθώς και από το διαδίκτυο.

Πρόκειται για μία αντικειμενοστραφής γλώσσα, που επιτρέπει σε οποιονδήποτε υπολογιστή όπου κι αν βρίσκεται αυτός, να έχει πρόσβαση και να χρησιμοποιεί μια εφαρμογή εγκατεστημένη σε κάποιο δίκτυο. Χρησιμοποιείται τόσο σε εφαρμογές διαδικτύου όσο και σε αυτόνομες εφαρμογές.

Αν τη συγκρίνουμε από πλευράς δομής με τις άλλες γλώσσες προγραμματισμού, η Java μοιάζει περισσότερο με τη C. Της μοιάζει αρκετά στον τρόπο σύνταξης και τη φιλοσοφία αλλά δεν είναι C.

Ιστορική αναδρομή

Η δημιουργία της Java ξεκίνησε το 1990 στην εταιρεία Sun Microsystems από μια ομάδα ανθρώπων με επικεφαλή τον James Gosling. Ο σκοπός της ομάδας αυτής ήταν να δημιουργήσουν προγράμματα τα οποία θα έκαναν πιο ''επαναστατική'' τη χρήση και τη λειτουργία των οικιακών συσκευών(βίντεο, τηλεόραση, στερεοφωνικά κτλ). Στην αρχή χρησιμοποίησαν την C++ λόγο ταχύτητας αλλά ο Gossling, σύντομα διαπίστωσε ότι η C++ δεν ήταν η κατάλληλη γλώσσα προγραμματισμού για τα προγράμματα που ήθελαν να φτιάξουν. Οπότε με τη βοήθεια ενός άλλου προγραμματιστή, του Patrick Naughton τον Αύγουστο του 1991 δημιούργησε μια καινούργια γλώσσα την οποία ονόμασαν OAK (βελανιδιά, από μια βελανιδιά που υπήρχε στον κήπο του κτηρίου που ήταν το γραφείο τους). Κατάφεραν τελικά να φτιάξουν έναν υπολογιστή χειρός που έμοιαζε με τηλεχειριστήριο το οποίο όμως δεν παρουσίασε την αναμενόμενη εμπορική επιτυχία παρόλο που κατασκευαστικά πέτυχε. Το 1993 η ομάδα προσπάθησε να προωθήσει την OAK στον τομέα Video On Demand (VOD) κάνοντας σχετική πρόταση στην εταιρεία Time Warner, η οποία ενδιαφερόταν για τέτοια έργα την εποχή εκείνη .Επειδή ο τομέας αυτός όμως κρίθηκε εμπορικά μη αξιοποιήσιμος η πρόταση απορρίφθηκε. Την εποχή οπού το internet ξεφεύγει από την κατάσταση απλού κειμένου και αρχίζουν να εμφανίζονται τα γραφικά , ένας από τους ιθύνοντες της Sun εκείνη την περίοδο, διέβλεψε ότι η OAK θα μπορούσε να χρησιμοποιηθεί στον παγκόσμιο ιστό. Έτσι η ομάδα το αντιλαμβάνεται και αλλάζει κατεύθυνση του έργου για τρίτη φορά, η φιλοσοφία της εταιρείας ήταν να διαθέτει τη γλώσσα δωρεάν ώστε να τυποποιηθεί. Κατόπιν μετονομάζεται σε Java τον Ιανουάριο του 1995, και λίγο αργότερα παρουσιάζεται επίσημα απο την Sun ως εργαλείο ανάπτυξης εφαρμογών για το Internet. Η διάδοση της αρχίζει από τα προγράμματα πλοήγησης όπως Netscape Navigator (2.0), Internet Explorer(3.0) με υποστήριξη Java.

Όπως αναφέραμε και παραπάνω η Java από πλευράς δομής μοιάζει περισσότερο με την C/C++. Μοιάζει αρκετά στον τρόπο σύνταξης και χρησιμοποιεί πολλά στοιχεία της, δεν είναι όμως C/C++. Αποφεύγει πολλά χαρακτηριστικά της C/C++ για λόγους απλότητας και ασφάλειας, ενώ σε άλλα επεκτείνει τις λειτουργίες τους για να γίνει πιο ευέλικτη και πιο σύγχρονη γλώσσα. Έχει πάρει επίσης και στοιχεία από την γλώσσα προγραμματισμού Smalltalk όπως τον πηγαίο κώδικα (Byte code) και το Garbage Collection(αυτόματος καθαρισμός της μνήμης).

Πλεονεκτήματα

Μια από τις ιδιότητες που κάνουν την Java να ξεχωρίζει από άλλες γλώσσες προγραμματισμού, είναι τα applets (μικροεφαρμογές), ειδικά προγράμματα που το σημαντικό τους χαρακτηριστικό είναι η ασφάλεια που διαθέτουν από ιούς και προγράμματα με κακό σκοπό. Αυτό συμβαίνει διότι τα applets δεν μπορούν να γράψουν στο σκληρό δίσκο αν δεν πάρουν άδεια. Για να τρέξει ένα applet

χρειάζεται ένα πρόγραμμα πλοήγησης (browser), όπως ο Internet Explorer, το Netscape Communicator κτλ., που να έχει σύνδεση στο διαδίκτυο ώστε να μπορούμε να το κατεβάσουμε. Όταν κάποιος κατεβάζει αρχεία από το διαδίκτυο, αντιμετωπίζει άμεσα τον κίνδυνο να "εισπράξει" κάποιον ιό η άλλο πρόγραμμα, το οποίο πιθανόν να προκαλέσει ζημίες στον σκληρό δίσκο του υπολογιστή του. Η Java λύνει αυτό το πρόβλημα περιορίζοντας αυστηρά το τι μπορεί να κάνει ένα applet. Ένα applet λ.χ. δε μπορεί να γράψει στο σκληρό μας δίσκο αν δε μας ζητήσει πρώτα την άδεια, ούτε και να μπλοκάρει τον υπολογιστή μας.

Η Java εκτός από τα applet μπορεί να χρησιμοποιηθεί και για την δημιουργία παντός φύσεως προγραμμάτων τα οποία καλούνται εφαρμογές (applications).

Σήμερα αποτελεί μια ιδιάιτερα δημοφιλής γλώσσα που αξιοπιείται με ποικίλους τρόπους και για διάφορα πεδία εφαρμογών. Αυτό οφείλεται σε ορισμένα ιδιαίτερα χαρακτηριστικά που καθιστούν τη γλώσσα δημοφιλή, όπως τα ακόλουθα:

•Είναι (σχετικά) απλή.

Οι δημιουργοί της Java άφησαν έξω από τη γλώσσα "δύσκολα" χαρακτηριστικά που συναντάμε σε άλλες γλώσσες, όπως λ.χ. οι δείκτες (pointers) ή οι δομές και οι ενώσεις (structures and unions) της C.

•Είναι αντικειμενοστραφής (object oriented).

Αντικειμενοστρέφεια είναι μία προσέγγιση στην ανάπτυξη λογισμικού που οργανώνει τόσο το πρόβλημα όσο και τη λύση του ως μία συλλογή από διακριτά αντικείμενα. Η Java όπως και η C, χρησιμοποιεί τις κλάσεις για να οργανώσει τον κώδικα σε λογικές ενότητες. Κατά το χρόνο εκτέλεσης, το πρόγραμμα δημιουργεί από τις κλάσεις αντικείμενα. Τα αντικείμενα αυτά έχουν δύο συνιστώσες: τα πεδία και τις μεθόδους. Τα πεδία περιγράφουν τι είναι το αντικείμενο οι μέθοδοι περιγράφουν τι κάνει το αντικείμενο. Οι κλάσεις μπορούν να κληρονομήσουν ιδιότητες από άλλες κλάσεις. Εκείνο που δεν επιτρέπεται είναι η πολλαπλή κληρονομικότητα (multiple inheritance), όπου μια κλάση έχει τη δυνατότητα να κληρονομήσει πεδία και μεθόδους από περισσότερες από μια άλλες κλάσεις.

•Επιτρέπει την επαναχρησιμοποίηση κώδικα.

Η επαναχρησιμοποίηση κώδικα επιτυγχάνεται με τη χρησιμοποίηση εισαγωγή ήδη υπαρχόντων προγραμμάτων των λεγόμενων κλάσεων οι οποίες επιτρέπουν στον προγραμματιστή να φτάσει γρήγορα στο επιθυμητό τελικό αποτέλεσμα.

•Είναι μεταγλωττιζόμενη αλλά μπορεί να χαρακτηριστεί και ως διερμηνευόμενη.

Η μεταγλώττιση του πηγαίου προγράμματος έχει σαν αποτέλεσμα την παραγωγή ενός ειδικού κώδικα, ο οποίος ονομάζεται κώδικας byte (bytecode). Ο κώδικας αυτός, ενώ μοιάζει με τον κώδικα σε γλώσσα μηχανής μπορεί να εκτελεσθεί από

οποιοδήποτε λειτουργικό σύστημα (Windows, UNIX, Mac OS κτλ) που διαθέτει διερμηνευτή της C. Εδώ έχουμε τη λύση σχετικά με την τοποθέτηση προγραμμάτων σε σελίδες του παγκόσμιου ιστού του Internet. Τα προγράμματα των υπολογιστών είναι κατά κύριο λόγο προσανατολισμένα στο υλικό (hardware) του υπολογιστή και στο λειτουργικό του σύστημα. Ένα πρόγραμμα γραμμένο για τα Windows δε μπορεί να τρέξει στον Macintosh, ούτε μια εφαρμογή για Macintosh μπορεί να εκτελεστεί σε ένα σταθμό εργασίας UNIX. Η Java λύνει αυτό το πρόβλημα δημιουργώντας εφαρμογές με ανεξαρτησία πλατφόρμας (platform independence). Αυτό γίνεται διότι ο μεταγλωττιστής της Java δε δημιουργεί γηγενή κώδικα (native code) προσαρμοσμένο στο συγκεκριμένο τύπο υπολογιστή(όπως κάνουν άλλες γλώσσες), αλλά δημιουργεί κώδικα Byte.

•Είναι ασφαλής.

Η Java έχει σχεδιαστεί από την αρχή με τέτοιον τρόπο, ώστε να παρέχει ασφάλεια εκτέλεσης του κώδικα σε δίκτυο. Αυτό είχε ως αποτέλεσμα τον περιορισμό αρκετών από τα χαρακτηριστικά που έχουν η C και η C++. Έτσι, δεν υπάρχουν δείκτες (pointers),ούτε μπορεί να γίνει αυθαίρετη προσπέλαση διευθύνσεων της μνήμης. Τα μέτρα αυτά παρέχουν προστασία από τη δράση των ιών. Η Java χρησιμοποιεί επίσης έναν ισχυρό μηχανισμό για τον έλεγχο αναμενόμενων και μη αναμενόμενων σφαλμάτων (handling).

•Υποστηρίζει πολυνημάτωση (multithreading).

Ένα πρόγραμμα Java μπορεί να περιλαμβάνει πολλές ξεχωριστές διαδικασίες, οι οποίες να εκτελούνται συνεχώς και ανεξάρτητα η μία από την άλλη. Μπορεί λόγου χάρη από το πρόγραμμα να μεταδίδεται μια εικόνα, και συγχρόνως ο χρήστης να εισαγάγει στοιχεία απο το πληκτρολόγιο.

•Κάνει συλλογή αχρήστων (garbage collection).

Τα προγράμματα της Java κάνουν από μόνα τους συλλογή αχρήστων. Αυτό σημαίνει ότι ο προγραμματιστής δε χρειάζεται να φροντίζει πλέον ο ίδιος για τη διαγραφή αχρήστων δεδομένων από τη μνήμη.

•Μπορεί να δημιουργήσει γρήγορο κώδικα.

Αυτό οφείλεται στην χρήση γρήγορων μεταγλωττιστών των λεγόμενων “just in time compilers” (JIT).

•Παρέχει βιβλιοθήκες κώδικα για διάφορες χρήσεις, όπως δημιουργία γραφικών, χειρισμό αλφαριθμητικών, μαθηματικές πράξεις, προσπέλαση αρχείων στο Διαδύκτιο, χειρισμό σχεσιακών βάσεων δεδομένων, δημιουργία εφαρμογών πελάτη-εξυπηρετητή, κλήση απομακρυσμένων αντικειμένων κ.τ.λ

• Με την Java μπορούμε να δημιουργήσουμε δυναμικές ιστοσελίδες (dynamic web pages), οι οποίες χρησιμοποιώντας το πρωτόκολλο μεταφοράς HTTP και την γλώσσα υπερκειμένου HTML προσφέρουν στους χρήστες του διαδικτύου πλείστες δυνατότητες. Μεταξύ άλλων, σε μια ιστοσελίδα μπορούμε:

- Να " παίξουμε" ήχο ή βίντεο.
- Να χρησιμοποιήσουμε διανυσματικά γραφικά αντί ψηφιογραφικών (bitmap) εικόνων.
- Να δημιουργήσουμε κίνηση σε πραγματικό χρόνο.
- Να δημιουργήσουμε φόρμες εισαγωγής στοιχείων από τους χρήστες.
- Να φτιάξουμε παιχνίδια και εφαρμογές πραγματικού χρόνου, στα οποία να συμμετέχουν ταυτόχρονα πολλοί παίκτες.

Η μορφές της Java

Η εταιρία Sun προσφέρει την Java σε τρεις μορφές, κάθε μια από τις οποίες στοχεύει σε διαφορετικό τομέα τεχνολογίας. Οι μορφές αυτές είναι:

• J2SE (Java 2 platform Standard Edition). Είναι το στάνταρτ πακέτο της Java το οποίο χρησιμοποιείται τόσο για εφαρμογές του Παγκόσμιου Ιστού όσο και για κανονικές προγραμματιστικές εφαρμογές.

•J2EE (Java 2 platform Enterprise Edition). Το πακέτο περιλαμβάνει μια σειρά από τεχνολογίες κάτω από ενιαία αρχιτεκτονική και στοχεύει στη δημιουργία επιχειρηματικών εφαρμογών ηλεκτρονικού εμπορίου βασισμένων στον Παγκόσμιο Ιστό.

•J2ME (Java 2 platform Micro Edition).Το πακέτο αποτελεί μια "ελαφρύ" έκδοση για αποκλειστική χρήση σε εμπορικές συσκευές και εργαλεία, όπως έξυπνες πιστωτικές κάρτες, κινητά τηλέφωνα, εικονοτηλέφωνα, συστήματα πλοήγησης αυτοκινήτων, χειριστήρια ασυρμάτου ελέγχου συσκευών κ.τ.λ.

NETBEANS

Το NetBeans IDE είναι ένα περιβαλλοντικό ανάπτυγμα IDE ένα εργαλείο στη διάθεση των προγραμματιστών για να γράψουν, να κάνουν compile, debug και να αναπτύξουν προγράμματα. Χρησιμοποιείται κυρίως για σχεδίαση Java εφαρμογών και είναι γραμμένο σε Java, αλλά μπορεί να υποστηρίξει ένα μεγάλο εύρος γλωσσών προγραμματισμού, όπως Java, C/C++ και άλλες. Υπάρχει επίσης ένας μεγάλος

αριθμός υπομονάδων (modules) που βοηθάνε στην επέκταση της λειτουργικότητας του NetBeans IDE.

Πλεονεκτήματα

- GNU license
- Υποστηρίζει πολλές γλώσσες προγραμματισμού
- Μεταφερσιμότητα λόγω του ότι είναι γραμμένο σε Java
- Debugger
- Ευκολία συγγραφής κώδικα
- Ευκολία δημιουργίας GUI
- Code completion

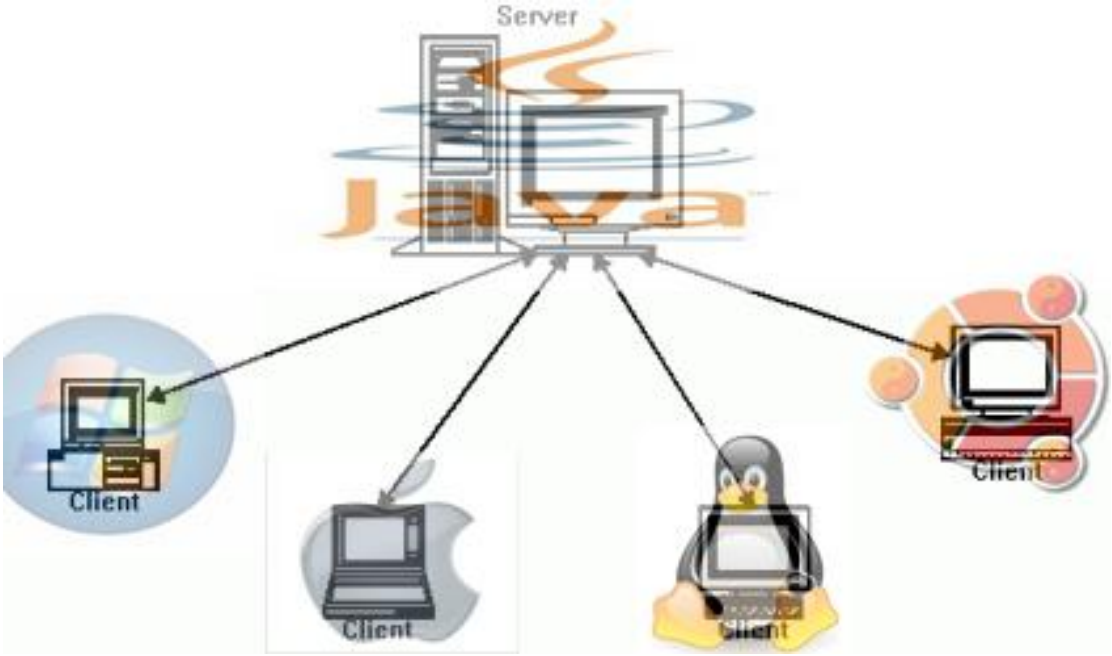
Μειονεκτήματα

- Αργός , επειδή τρέχει σε εικονική μηχανή

Παράλληλα, η έκδοση 6.5 έχει σημαντικές καινοτομίες σε σχέση με την προηγούμενη και η πιο σημαντική βρίσκεται στο γραφικό περιβάλλον το οποίο είναι πλέον πολύ εύκολο στην χρήση. Ταυτόχρονα, βοηθάει με την αυτόματη δημιουργία κώδικα έτσι ώστε να γλιτώνουμε πολύ χρόνο στην υλοποίηση της εφαρμογής. Επίσης, μέσω του γραφικού περιβάλλοντος μπορούμε και εντοπίζουμε τα λάθη και τις ανωμαλίες στον κώδικα που γράφουμε πολύ γρήγορα. Στην εφαρμογή έχει χρησιμοποιηθεί η έκδοση 6.5.

Απομακρυσμένη διαχείριση και έλεγχο απλών πληροφοριακών συστημάτων. Client-Server τεχνολογίες με την αντικειμενοστραφή γλώσσα προγραμματισμού JAVA

Η πτυχιακή έχει να κάνει με την διαχείριση απομακρυσμένων λειτουργικών συστημάτων. Με την βοήθεια της java κατέφερα να ενόσω και να κανό διαχείριση το διάφορων πληροφοριακών συστημάτων από απόσταση. Η τεχνολογία που χρησιμοποίησα είναι 2-tier, δηλαδή χρήση ενός client που ελέγχετε από ένα server. Στον server μπορούν να φιλοξενηθούν πολύ client με κόστος βέβαια την ταχύτητα λήψης εικόνας αφού μέσο διαδικτύου η ταχύτητες είναι ακόμη χαμηλές σε αντίθεση σε τοπικό δίκτυο το αποτέλεσμα είναι πολύ καλό.



Client

Αποστολή εικόνας

Στον παρακάτω κώδικα παρατηρούμε την διαδικασία που κάνει ο client έτσι ώστε ο server να έχει την εικόνα του υπολογιστή του. Όλο αυτό το πέτυχε με την βιβλιοθήκη Robot. Όμως χριζόμαστε και αλλά πράγματα για να ατιμάσουμε την διαδικασία της μετάδοσης όπως την βιβλιοθήκη του Socket και ObjectOutputStream. Στην ουσία η διαδικασία που κάνει η κλάση αυτή είναι να

τραβήξει φωτογραφίες (screenshot) της οθόνης να της αποθηκεύει και στην συνέχεια να της ετοιμάζει για μετάδοση μέσω δικτύου με χρήση του πρωτοκόλλου UDP.

```
package remoteclient;


import java.awt.Rectangle;

import java.awt.Robot;

import java.awt.image.BufferedImage;

import java.io.IOException;

import java.io.ObjectOutputStream;

import java.net.Socket;

import javax.swing.ImageIcon;


class ScreenSpyer extends Thread {


   Socket socket = null;

   Robot robot = null; // Used to capture screen

   Rectangle rectangle = null; //Used to represent screen dimensions

   boolean continueLoop = true; //Used to exit the program

      public ScreenSpyer(Socket socket, Robot robot,Rectangle rect) {

      this.socket = socket;

      this.robot = robot;

      rectangle = rect;

//Με το start() αρχίζει η εκτέλεση του thread

      start();

   }
```

//Κάποια βασικά πράγματα που πρέπει να πούμε για τα thread είναι ότι ένα thread είναι εκτελέσιμο τμήμα μιας διεργασίας. Σε ένα πρόγραμμα μπορούμε να έχουμε πολλά threads. Ο βασικός λόγος που χρησιμοποιούμε εδώ thread είναι γιατί μπορούμε σε ένα server να έχουμε παράλληλα πολλούς client.

```
public void run(){
    ObjectOutputStream oos = null;
    //Used to write an object to the stream
    try{
       //Prepare ObjectOutputStream
       oos = new ObjectOutputStream(socket.getOutputStream());
       oos.writeObject(rectangle);
    }catch(IOException ex){
       ex.printStackTrace();
    }
   while(continueLoop){
```

// Capture screen κατή πολύ χρήσιμο στα λειτουργικά συστήματα με αυτό τον τρόπο μπορούμε να κρατήσουμε κάτι που βλέπουμε στην οθόνη μας. Μια σειρά από screen capture μας δίνει την ψευδαίσθηση ότι έχουμε κινούμενη εικόνα. Με αυτό το τέχνασμα καταφέρνουμε να βλέπουμε το monitor του client

```
       BufferedImage image = robot.createScreenCapture(rectangle);
       /* I have to wrap BufferedImage with ImageIcon because BufferedImage class
```

```
         * does not implement Serializable interface
         */
        ImageIcon imageIcon = new ImageIcon(image);
        //Send captured screen to the server
        try {
           System.out.println("before sending image");
           oos.writeObject(imageIcon);
           oos.reset();
           //Clear ObjectOutputStream cache
           System.out.println("New screenshot sent");
        } catch (IOException ex) {
          ex.printStackTrace();
        }
```

//Ο παρακάτω κώδικας προστέθηκε για να εξοικονομούμε bandwidth. Δηλαδή σε κάθε print screen το thread μένη ανενεργό για 100ms. Σε περιπτοση που έχουμε πολλούς client συνδεδεμένους σε ένα server δεν θα προλάβαινε ο server να τους δει όλους.

```
        try{
           Thread.sleep(100);
        }catch(InterruptedException e){
           e.printStackTrace();
        }
      }

   }

}
```

Υπεύθυνο να βλέπει της κινήσεις που γίνονται στον client

Στην παρακάτω κλάση την έχουμε δημιουργήσει για να πιάνει της κινήσεις που γίνονται από τον server στον client.

```
package remoteclient;


import java.awt.Robot;

import java.io.IOException;

import java.net.Socket;

import java.util.Scanner;


/*

 * Used to recieve server commands then execute them at the client side

 */

class ServerDelegate extends Thread {


   Socket socket = null;

   Robot robot = null;

   boolean continueLoop = true;


   public ServerDelegate(Socket socket, Robot robot) {

      this.socket = socket;

      this.robot = robot;

      start(); //Start the thread and hence calling run method

   }


   public void run(){
```

```
Scanner scanner = null;
try {
   //prepare Scanner object
   System.out.println("Preparing InputStream");
   scanner = new Scanner(socket.getInputStream());

   while(continueLoop){
      //recieve commands and respond accordingly
      System.out.println("Waiting for command");
      int command = scanner.nextInt();
      System.out.println("New command: " + command);
```

//Στον παρακάτω κώδικα χωρίζουμε την κίνηση που έγινε. Αν ο server έστειλε το -1 τότε πατήθηκε το κουμπί του ποντικιού αν στήλη το -2 τότε στο κουμπί του ποντικιού έχουμε επαναφορά , σε περίπτωση που έχουμε το -3 και -4 τότε έχουμε κίνηση (πατήσαμε κάποιο κουμπί από το πληκτρολόγιο και το αφήσαμε αντίστοιχα) από το πληκτρολόγιο και με το -5 έχουμε κίνηση του ποντικιού δηλαδή αλλάζει θέση ο δείκτης του ποντικιού. Όλα αυτά τα ορίσαμε σε μια άλλη κλάση εδώ απλός σκανάρουμε την κίνηση που γίνετε και την αποτυπώνουμε στην οθόνη του client .

```
      switch(command){
         case -1:
            robot.mousePress(scanner.nextInt());
         break;
         case -2:
            robot.mouseRelease(scanner.nextInt());
         break;
         case -3:
            robot.keyPress(scanner.nextInt());
         break;
```

```
                case -4:
                    robot.keyRelease(scanner.nextInt());
                break;
                case -5:
                    robot.mouseMove(scanner.nextInt(), scanner.nextInt());
                break;
            }
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
  }

}
```

Κινήσεις που επιτρέπονται να γίνουν από τον server

Σε αυτή την κλάση ορίζω της κινήσεις και με την παραπάνω κλάση απλός ελέγχω την κίνηση που γίνετε από τον server. Η κινήσεις που έχω βάλει είναι κλικ του ποντικιού, χρήση του πληκτρολογίου και κίνηση του δείκτη του ποντικιού.

```
package remoteclient;


public enum EnumCommands {
   PRESS_MOUSE(-1),
   RELEASE_MOUSE(-2),
   PRESS_KEY(-3),
   RELEASE_KEY(-4),
   MOVE_MOUSE(-5);

   private int abbrev;

   EnumCommands(int abbrev){
      this.abbrev = abbrev;
   }

  public int getAbbrev(){
      return abbrev;
   }
}
```

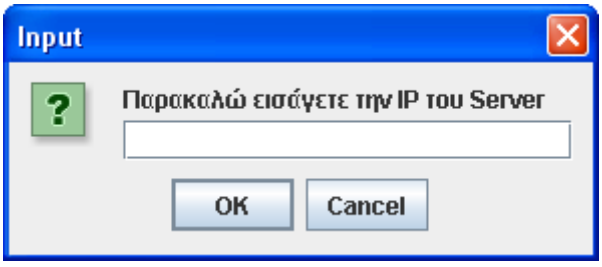
Το κυρίως πρόγραμμα που κάνει χρήση τον παραπάνω κλάσεων

Η παρακατο κλαση δημηουργηθηκε για να ενοσουμε της πιο πανο κλασης και να δημηουργησουμε το γραφικο περιβαλον του χρηστη (client).

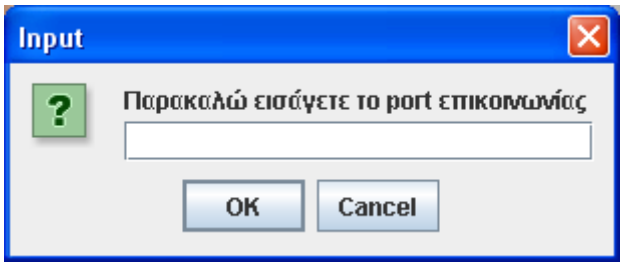
```
package remoteclient;


import java.awt.AWTException;

import java.awt.Dimension;

import java.awt.GraphicsDevice;

import java.awt.GraphicsEnvironment;

import java.awt.Rectangle;

import java.awt.Robot;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.IOException;

import java.net.Socket;

import java.net.UnknownHostException;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JOptionPane;


public class ClientInitiator {


    Socket socket = null;
```

//Η λέξη main (main είναι το όνομα της μεθόδου) αναφέρεται σε μια μέθοδο που σαν σκοπό της έχει να εκτελεί οτιδήποτε κώδικα προσθέτουμε μέσα στα άγκιστρα που καθορίζουν το εύρος δράσης της. Με άλλα λόγια, για να είναι μια κλάση εκτελέσιμη ή ακόμα μια εφαρμογή, είναι απαραίτητη προϋπόθεση να υπάρχει μια main μέθοδο μέσα στην κλάση μας. Ανεξαρτήτου πόσο πολύπλοκη ή μεγάλη σε όγκο προγραμματισμού είναι η εφαρμογή μας, μόνο μια main μέθοδο απαιτείται για την εκτέλεσή της. Κάθε μέθοδος ακολουθείται από μια παρένθεση μέσα στην οποία ορίζονται οι μεταβλητές τις οποίες οι μέθοδος δέχεται σαν input και θα επεξεργαστεί τις τιμές τους. Στην δική μας main η παρένθεση περιέχει τον πίνακα (array) με το όνομα args ο οποίος δέχεται σαν input μόνο String δεδομένα. Εάν και ο ορισμός της main δεν πρόκειται να αλλάξει ποτέ, τον args πίνακα τον χρειαζόμαστε αποκλειστικά και μόνο όταν τρέχουμε java εφαρμογή μέσα από command-line και ορίζουμε κάποιες αρχικές τιμές μεταβλητών πριν την εκτέλεση του προγράμματος. Μπροστά από το όνομα της μεθόδου υπάρχει η λέξη void. Η συγκεκριμένη λέξη κλειδί μας ενημερώνει ότι δεν αναμένουμε καμία τιμή να επιστρέψει στο κύριο πρόγραμμα σαν αποτέλεσμα της εκτέλεσης της μεθόδου. Η main απλά εκτελεί τον κώδικα που τις έχουμε ορίσει με μετά το πέρας της ενέργειας αυτής, ολοκληρώνεται και το κύριο πρόγραμμα.Τέλος, η λέξη public σημαίνει ότι μπορεί οποιοσδήποτε από την ίδια εφαρμογή, project, πακέτο ή κλάση να ενεργοποιήσει την εκτέλεση της main, ενώ η λέξη static σημαίνει ότι η μέθοδο ανήκει στην κλάση και για να ενεργοποιηθεί η εκτέλεση της δεν είναι αναγκαίο να δημιουργήσουμε αντικείμενο. Εάν αυτοί οι όροι κάπως σας έχουν συγχύσει μην ανησυχείτε γιατί ακόμα δεν έχει έρθει η σωστή χρονική στιγμή για την ανάλυσή τους.

```
public static void main(String[] args){

    String ip = JOptionPane.showInputDialog("Παρακαλώ εισάγετε την IP του
Server");
```



String port = JOptionPane.showInputDialog("Παρακαλώ εισάγετε το port επικοινωνίας ");



new ClientInitiator().initialize(ip, Integer.parseInt(port));

}

public void initialize(String ip, int port){

Robot robot = null; //Used to capture the screen

Rectangle rectangle = null; //Used to represent screen dimensions

try {

System.out.println("Connecting to server .....");

socket = new Socket(ip, port);

System.out.println("Connection Established.");

```
            //Get default screen device
            GraphicsEnvironment
gEnv=GraphicsEnvironment.getLocalGraphicsEnvironment();
            GraphicsDevice gDev=gEnv.getDefaultScreenDevice();

            //Με της παρακάτω εντολές αποθηκεύουμε (pixel) της διαστάσεις της οθόνης
για να μπορούμε στη συνέχεια να της προσαρμόσουμε στο παράθυρο του server
            Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
            rectangle = new Rectangle(dim);

            //Prepare Robot object
            robot = new Robot(gDev);

            //Στο παρακάτω σημείο κώδικα καλούμε την κλάση drawGUI
            drawGUI();
            //ScreenSpyer sends screenshots of the client screen
            new ScreenSpyer(socket,robot,rectangle);
            //ServerDelegate recieves server commands and execute them
            new ServerDelegate(socket,robot);
        } catch (UnknownHostException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (AWTException ex) {
                ex.printStackTrace();
        }
    }
```

//Το παρακάτω κομμάτι κώδικα αφού δημιουργήσουμε τη σύνδεση μεταξύ client και server δημιουργεί ένα κουμπί που τερματίζει την σύνδεση μεταξύ τους. Απλός κώδικας δημιουργούμε ένα frame που έχει ένα κουμπί με όνομα ‘’Τερματισμός’’. Όταν πατήσουμε το κουμπί τότε παύει να υπάρχει και η σύνδεση. Το κουμπί μπορεί να το πατήσει και ο server και ο client έχοντας το ίδιο αποτέλεσμα και στης δυο περιπτώσεις.

```
   private void drawGUI() {

      JFrame frame = new JFrame("Remote Admin");

      JButton button= new JButton("Τερματισμός");


      frame.setBounds(100,100,150,150);

      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.add(button);

      button.addActionListener( new ActionListener() {


         public void actionPerformed(ActionEvent e) {

            System.exit(0);

         }

      }

     );

     frame.setVisible(true);

   }

}
```

Server

Η κλάση που διαχειρίζεται της κινήσεις και ενημερώνει τον client

Η παρακάτω κλάση είναι υπεύθυνη να βλέπει της κινήσεις που κάνη ο server για λογαριασμό του client. Για να πιάνει ο Η/Υ της κινήσεις στην java έχουμε τα Action Listener. Εδώ χρησιμοποιούμε τον Key Listener και το Mouse Listener.

```
package remoteserver;


import java.awt.Rectangle;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseMotionListener;

import java.io.IOException;

import java.io.PrintWriter;

import java.net.Socket;

import javax.swing.JPanel;


class ClientCommandsSender implements KeyListener,

      MouseMotionListener,MouseListener {


   private Socket cSocket = null;

   private JPanel cPanel = null;

   private PrintWriter writer = null;

   private Rectangle clientScreenDim = null;


   ClientCommandsSender(Socket s, JPanel p, Rectangle r) {
```

```
        cSocket = s;
        cPanel = p;
        clientScreenDim = r;
        //Στης παρακάτω τρεις εντολές οριοθετούμε που θα δουλέψουν η Listener.
```
Δηλαδή όταν γίνει η σύνδεση δημιουργείτε ένα panel που έχουμε την εικόνα από τον client και ότι κίνηση γίνετε από τον server τα actions Listener την πιάνουν και κάνουν την αντίστοιχη κίνηση και στον client
```
        cPanel.addKeyListener(this);
        cPanel.addMouseListener(this);
        cPanel.addMouseMotionListener(this);
        try {
            //Prepare PrintWriter which will be used to send commands to
            //the client
           writer = new PrintWriter(cSocket.getOutputStream());
        } catch (IOException ex) {
           ex.printStackTrace();
        }

    }

    //Not implemeted yet
    public void mouseDragged(MouseEvent e) {
    }

    public void mouseMoved(MouseEvent e) {
        double xScale = clientScreenDim.getWidth()/cPanel.getWidth();
        System.out.println("xScale: " + xScale);
```

```
        double yScale = clientScreenDim.getHeight()/cPanel.getHeight();
        System.out.println("yScale: " + yScale);
        System.out.println("Mouse Moved");
        writer.println(EnumCommands.MOVE_MOUSE.getAbbrev());
        writer.println((int)(e.getX() * xScale));
        writer.println((int)(e.getY() * yScale));
        writer.flush();
    }

    //Στον παρακάτω κώδικα έχουμε της μεθόδους που υλοποιούνται όταν ένας Listener αρχίσει να πιάνει κάποιες κινήσεις. Στης συγκεκριμένες μεθόδους έχουμε την κίνηση του ποντικιού και τον αν έγινε κάποιο κλικ από το ποντίκι
    public void mouseClicked(MouseEvent e) {
    }

    public void mousePressed(MouseEvent e) {
        System.out.println("Mouse Pressed");
        writer.println(EnumCommands.PRESS_MOUSE.getAbbrev());
        int button = e.getButton();
        int xButton = 16;
        if (button == 3) {
            xButton = 4;
        }
        writer.println(xButton);
        writer.flush();
    }
```

```
    public void mouseReleased(MouseEvent e) {
        System.out.println("Mouse Released");
        writer.println(EnumCommands.RELEASE_MOUSE.getAbbrev());
        int button = e.getButton();
        int xButton = 16;
        if (button == 3) {
            xButton = 4;
        }
        writer.println(xButton);
        writer.flush();
    }

    //Η παρακάτω μέθοδοι δεν της υλοποιούμε αλλά όταν κάνουμε implement σε μια κλάση μπαίνουν όλες οι μέθοδοι ακόμα και αν δεν της υλοποιούμε
    public void mouseEntered(MouseEvent e) {
    }

    public void mouseExited(MouseEvent e) {

    }

    public void keyTyped(KeyEvent e) {
    }
```

```
    //Στης παρακάτω μεθόδους έχουμε υλοποίηση τον listener από το πληκτρολόγιο
    public void keyPressed(KeyEvent e) {
        System.out.println("Key Pressed");
        writer.println(EnumCommands.PRESS_KEY.getAbbrev());
        writer.println(e.getKeyCode());
        writer.flush();
    }

    public void keyReleased(KeyEvent e) {
        System.out.println("Key Released");
        writer.println(EnumCommands.RELEASE_KEY.getAbbrev());
        writer.println(e.getKeyCode());
        writer.flush();
    }

}
```

Αποστολή τον event από τον server και λήψη εικόνας από τον client

Στην παραπάνω κλάση πιάναμε της κινήσεις που έκανε ο server στην οθόνη του client εδώ ενημερώνουμε τον client για το τη κινήσεις που έγιναν και ο client μας ενημερώνει για την κατάσταση και της αλλαγές στην οθόνη του.

```
package remoteserver;


import java.awt.BorderLayout;

import java.awt.Rectangle;

import java.beans.PropertyVetoException;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.net.Socket;

import javax.swing.JDesktopPane;

import javax.swing.JInternalFrame;

import javax.swing.JPanel;


class ClientHandler extends Thread {


   private JDesktopPane desktop = null;

   private Socket cSocket = null;

   private JInternalFrame interFrame = new JInternalFrame("Client Οθονη",

                                       true, true, true);

   private JPanel cPanel = new JPanel();


   public ClientHandler(Socket cSocket, JDesktopPane desktop) {

      this.cSocket = cSocket;

      this.desktop = desktop;
```

```
    start();
}

public void drawGUI(){
    interFrame.setLayout(new BorderLayout());
    interFrame.getContentPane().add(cPanel,BorderLayout.CENTER);
    interFrame.setSize(100,100);
    desktop.add(interFrame);
    try {

        interFrame.setMaximum(true);
    } catch (PropertyVetoException ex) {
        ex.printStackTrace();
    }

    cPanel.setFocusable(true);
    interFrame.setVisible(true);
}

public void run(){

    Rectangle clientScreenDim = null;
```

```
        ObjectInputStream ois = null;


        drawGUI();


        try{
           //Παρακάτω ακούμε για την άφιξη της εικόνας από τον client στον server
           ois = new ObjectInputStream(cSocket.getInputStream());
           clientScreenDim =(Rectangle) ois.readObject();
        }catch(IOException ex){
           ex.printStackTrace();
        }catch(ClassNotFoundException ex){
           ex.printStackTrace();
        }
        //Τοποθετούμε την εικόνα στο panel από το screen capture του client
        new ClientScreenReciever(ois,cPanel);
        //Στέλνουμε τα event από τον server στον client
        new ClientCommandsSender(cSocket,cPanel,clientScreenDim);
     }


}
```

Απεικόνιση του Client στον Server

Η παρακάτω κλάση είναι υπεύθυνη για την απεικόνιση της οθόνης του client στον server. Η μέθοδος ClientScreenReciever() έχει δυο ορίσματα το ένα είναι η εικόνα του client και η άλλη είναι το μέγεθος του panel του server που πρέπει να τοποθετηθεί η εικόνα. Τα δυο αυτά ορίσματα μας τα δίνει η παραπάνω κλάση που λαμβάνει την εικόνα και ορίζει και το panel.

```
package remoteserver;


import java.awt.Graphics;

import java.awt.Image;

import java.io.IOException;

import java.io.ObjectInputStream;

import javax.swing.ImageIcon;

import javax.swing.JPanel;


class ClientScreenReciever extends Thread {


   private ObjectInputStream cObjectInputStream = null;

   private JPanel cPanel = null;

   private boolean continueLoop = true;


   public ClientScreenReciever(ObjectInputStream ois, JPanel p) {

      cObjectInputStream = ois;

      cPanel = p;

      //Εδώ καλούμε το thread

      start();
```

```
    }

    public void run(){

        try {

            //Διαβάζουμε τα screenshot του client και τα τοποθετούμε στον παρακάτω
κώδικα
            while(continueLoop){

                ImageIcon imageIcon = (ImageIcon) cObjectInputStream.readObject();
                System.out.println("New image recieved");
                Image image = imageIcon.getImage();
                image = image.getScaledInstance(cPanel.getWidth(),cPanel.getHeight()
                                        ,Image.SCALE_FAST);
                //Τοποθετούμε την εικόνα στο panel
                Graphics graphics = cPanel.getGraphics();
                graphics.drawImage(image, 0, 0,
cPanel.getWidth(),cPanel.getHeight(),cPanel);
            }
         } catch (IOException ex) {
            ex.printStackTrace();
        } catch(ClassNotFoundException ex){
           ex.printStackTrace();
        }
    }
}
```

Κινήσεις που κάνει ο server στον client

Η κινήσεις που έχω βάλει να κάνει ο server στον client κίνηση του ποντικιού, κλικ του ποντικιού και χρήση του πληκτρολογίου.

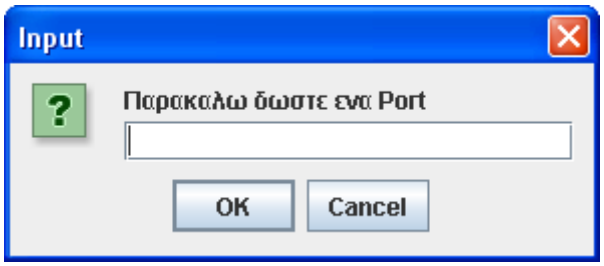
```
package remoteserver;




public enum EnumCommands {
   PRESS_MOUSE(-1),
   RELEASE_MOUSE(-2),
   PRESS_KEY(-3),
   RELEASE_KEY(-4),
   MOVE_MOUSE(-5);

   private int abbrev;

   EnumCommands(int abbrev){
      this.abbrev = abbrev;
   }

   public int getAbbrev(){
      return abbrev;
   }
}
```

Το κυρίως πρόγραμμα του server

Σε αυτή την κλάση έχουμε το εκτελέσιμο κομμάτι του server. Ενώνει της παραπάνω κλάσης πετυχενοντας την σύνδεση με τους clients στον server μπορούν να συνδεθούν παραπάνω από ένα client σε τοπικό δίκτυο η ποιότητα της υπηρεσίας είναι καλή αλλά μέσο διαδικτύου η ποιότητα μειώνετε.

```
package remoteserver;


import java.awt.BorderLayout;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import javax.swing.JDesktopPane;

import javax.swing.JFrame;

import javax.swing.JOptionPane;


/**

 * Το κυρίως πρόγραμμα του server

 */

public class ServerInitiator {

   //εδώ δημιουργούμε το frame του server

   private JFrame frame = new JFrame();



   private JDesktopPane desktop = new JDesktopPane();


   public static void main(String args[]){

      String port = JOptionPane.showInputDialog("Παρακαλω δωστε ενα Port");
```

```
    new ServerInitiator().initialize(Integer.parseInt(port));
}
```




```
public void initialize(int port){

    try {
        ServerSocket sc = new ServerSocket(port);
        //Καλούμε την κλάση drawGUI()
        drawGUI();
        //Ο server περιμένει τον client για να συνδεθη
        while(true){
            Socket client = sc.accept();
            System.out.println("New client Connected to the server");

            new ClientHandler(client,desktop);
        }
    } catch (IOException ex) {
        ex.printStackTrace();
```

```
        }
    }


    public void drawGUI(){
            frame.add(desktop,BorderLayout.CENTER);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            //Show the frame in a maximized state
frame.setExtendedState(frame.getExtendedState()|JFrame.MAXIMIZED_BOTH);
            frame.setVisible(true);
    }
}
```