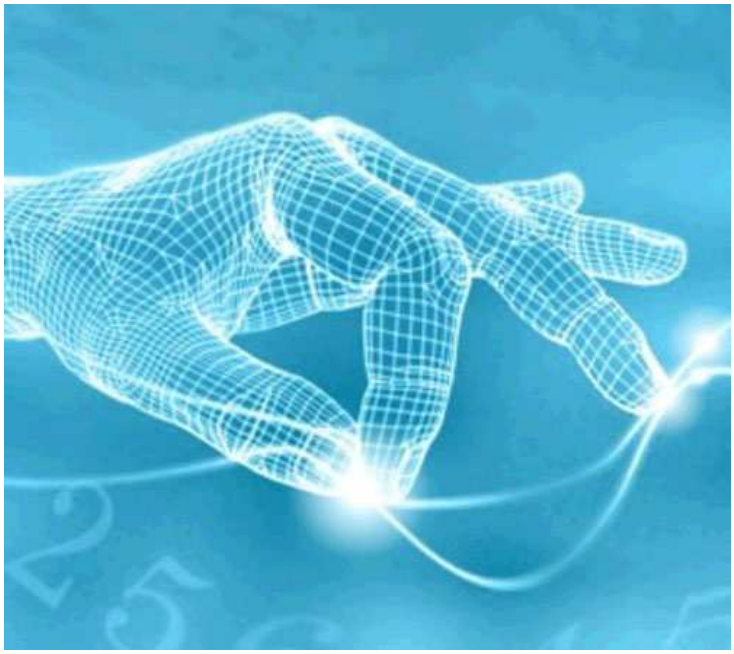
ΤΕΧΝΟΛΟΓΙΚΟ
ΕΚΠΑΙΔΕΥΤΙΚΟ
ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Δημιουργία προγράμματος υπολογισμού υγρομόνωσης με την δημιουργία αντικειμενοστραφούς κώδικα



Σπουδαστής:
Γεώργιος Μιχάλης
17/11/2011

Επιβλέπων
Δρ. Νικόλαος Παπαδάκης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΜΙΧΑΛΗΣ ΓΕΩΡΓΙΟΣ

ΕΙΣΗΓΗΤΗΣ: ΝΙΚΟΛΑΟΣ ΠΑΠΑΔΑΚΗΣ

ΠΕΡΙΕΧΟΜΕΝΑ

Εισαγωγή

Δομή Εργασίας

Η εργασία αποτελείται από τα παρακάτω κεφάλαια. Στο πρώτο κεφάλαιο γίνεται αναφορά στο πρόβλημα της υγρομόνωσης, αναφέροντας κάποιες βασικές έννοες, όπως τους μηχανισμούς διάχυσης υδρατμών, και τα υδατοπερατά και υδατοστεγή υλικά.

Στο δεύτερο κεφάλαιο γίνεται μια αναφορά στις κατηγορίες προγραμματισμού και τον λόγο που χρησιμοποιήσαμε την γλωσσά java για την περάτωση της εργασίας μας.

Στο τρίτο κεφάλαιο παρουσιάζονται πιο λεπτομερώς τα στοιχεία σε μια γλώσσα προγραμματισμού όπως οι τύποι μεταβλητών κλπ.

Στο τέταρτο κεφάλαιο παρουσιάζεται μια τυπική διαδικασία αντιμετώπισης ενός προβλήματος από έναν προγραμματιστή καθώς και τα βήματα που πρέπει να ακολουθούνται για να σχεδιαστεί ένα πρόγραμμα με μηχανολογική προσέγγιση.

Στο πέμπτο κεφάλαιο παρουσιάζεται αναλυτικά η διαδικασία που ακολουθήθηκε για τον σχεδιασμό του προγράμματος, καθώς και λεπτομερής ανάλυση της λογικής που χρησιμοποιήθηκε σε κάποια κομμάτια του κώδικα.

Στο έκτο κεφάλαιο βρίσκεται το εγχειρίδιο χρήσης του προγράμματος.

Στο έβδομο κεφάλαιο αναφέρονται κάποια χρήσιμα στοιχεία που πρέπει να γνωρίζει κάθε προγραμματιστής, όπως το πώς να κάνει το πρόγραμμα του να διαβάζει αρχεία, η μορφή αρχείων xml, κλπ.

1 Υγρασία και υγρομόνωση

1.1 Γενικά

Το παρακάτω κεφάλαιο είναι από τις σημειώσεις του κ. Μ. Κτενιαδάκη, Σεμινάριο ΕΝΕΡ4-ΙΚΕΜΤΕΕ 2009 κεφ 1 υγρομόνωση, σελ 1-4.

Ο ατμοσφαιρικός αέρας είναι μίγμα διαφόρων αερίων, κυρίως N_2 και O_2 αλλά και CO_2, H_2, ευγενών αερίων και υδρατμών.

Αέρας απαλλαγμένος υδρατμών χαρακτηρίζεται ως ξηρός αέρας, ενώ αέρας σε συνήθη κατάσταση, που περιέχει νερό σε μορφή υδρατμών, χαρακτηρίζεται ως υγρός αέρας (αν δεν αναφέρεται χαρακτηριστικό, εννοείται υγρός αέρας). Ως απόλυτη υγρασία του αέρα **(α)** ορίζεται η μάζα των υδρατμών (M_u), η οποία περιέχεται σε 1 m^3 αέρα. Μετράται σε g/m^3 ή σε kg/m^3. Από τον ορισμό καταλαβαίνουμε ότι πρόκειται για την πυκνότητα του αέρα σε υδρατμούς. Υπολογίζεται από τον τύπο:

$$a = m_{\text{υδρατμών}} / V_{\text{αέρα}}$$

Ως ειδική υγρασία του αέρα, (**w**), ορίζεται η μάζα των υδρατμών (M_u), η οποία περιέχεται σε 1 kg ξηρού αέρα. Μετράται σε g/$kg_{\xi\alpha}$ ή kg/$kg_{\xi\alpha}$. Υπολογίζεται από τον τύπο:

$$w = m_{\text{υδρατμών}} / V_{\text{ξηρού αέρα}}$$

Για μια ορισμένη θερμοκρασία, η περιεκτικότητα του ατμοσφαιρικού αέρα σε υδρατμούς δεν είναι δυνατόν να ξεπεράσει μια καθορισμένη τιμή. Ο αέρας για μια καθορισμένη θερμοκρασία, μπορεί να περιέχει μία μέγιστη δυνατή ποσότητα υδρατμών (M_{uk}), οπότε ονομάζεται κορεσμένος.

Όσο μικρότερη είναι η θερμοκρασία του αέρα, τόσο μικρότερη ποσότητα υδρατμών M_{uk} μπορεί να συγκρατήσει ο αέρας για να γίνει κορεσμένος. Συνεπώς, αέρας θερμοκρασίας Θ_a και ειδικής υγρασίας W_a, και που περιέχει δεδομένη μάζα υδρατμών $\mathbf{m_u}$, μπορεί να γίνει κορεσμένος αν ψυχθεί σε κάποια θερμοκρασία Θ_k. Αυτή η θερμοκρασία είναι γνωστή ως «σημείο δρόσου» του αέρα ή θερμοκρασία κορεσμού.

Αν αυτός ο αέρας ψυχθεί σε ακόμη χαμηλότερη θερμοκρασία, τότε ένα μέρος των περιεχόμενων υδρατμών θα αποβληθεί από την μάζα του αέρα, σε υγρή μορφή (συμπυκνώματα), έτσι ώστε το προκύπτον νέο μίγμα να είναι κορεσμένο σ' αυτή τη νέα χαμηλότερη θερμοκρασία. Προφανώς, στη νέα αυτή κατάσταση ο αέρας θα έχει άλλη, μικρότερη ειδική υγρασία w, αφού μειώνεται η περιεχόμενη μέγιστη δυνατή ποσότητα υδρατμών M_u. Θεωρώντας τον ατμοσφαιρικό αέρα ως μίγμα τελείων αερίων, η πίεση P του (υγρού) αέρα, δηλ. η ατμοσφαιρική πίεση, ισούται με το άθροισμα των μερικών πιέσεων των συστατικών του (νόμος των μερικών πιέσεων του Dalton). Αν θεωρηθεί ότι τα συστατικά του αέρα είναι δύο: ο (ξηρός) αέρας και οι υδρατμοί, με τις μερικές πιέσεις (τάσεις) τους να είναι αντίστοιχα P_a, και P_u , θα ισχύει:

P = P_a + P_u {1}.

Η μερική πίεση των υδρατμών έχει κάποια μέγιστη τιμή P_{uk} όταν αυτός είναι κορεσμένος, σε κάποια θερμοκρασία (οπότε περιέχει μάζα υδρατμών M_{uk} που ονομάζεται πίεση (τάση) κορεσμένων υδρατμών. Ως σχετική υγρασία του αέρα, **(ϕ)**, ορίζεται ο λόγος της μερικής πίεσης των υδρατμών P_u προς την μερική πίεση (τάση) των υδρατμών P_{uk} όταν είναι κορεσμένος, στην ίδια θερμοκρασία. Μετρείται συνήθως επί τοις εκατό (%), οπότε 0 ≤ ϕ ≤ 100%.

$$\phi=\frac{P_u}{P_{uk}} \quad \{2\}.$$

Μπορεί να αποδειχθεί ότι, για θερμοκρασίες αέρα μικρότερες των 65°C και σε κανονική ατμοσφαιρική πίεση, η σχετική υγρασία του αέρα ισούται, κατά προσέγγιση, με την αναλογία της περιεχόμενης μάζας υδρατμών $M_υ$ προς τη μέγιστη δυνατή περιεκτικότητα του αέρα σε υδρατμούς M_{uk} (όταν είναι κορεσμένος) στην ίδια θερμοκρασία, δηλαδή:

$$\phi=\frac{m_u}{mP_{uk}}=\frac{w_u}{W_{uk}} \quad \{3\}.$$

Σύμφωνα με την παραπάνω σχέση, για κάποια ορισμένη θερμοκρασία Θ_a του αέρα, όσο μεγαλύτερη είναι η σχετική υγρασία του τόσο η περιεχόμενη μάζα υδρατμών πλησιάζει την μάζα κορεσμού και, επομένως, αρκεί μικρή προσθήκη υδρατμών ή μικρή πτώση της θερμοκρασίας για να γίνει κορεσμένος. Άρα υπάρχει συσχέτιση μεταξύ της θερμοκρασίας και της σχετικής υγρασίας του αέρα με το σημείο δρόσου του Θ_k. Η πίεση κορεσμού των υδρατμών P_k, που εμφανίζεται όταν ο αέρας είναι κορεσμένος σε ορισμένη θερμοκρασία, είναι η μέγιστη δυνατή. Αν η σχετική υγρασία του ατμοσφαιρικού αέρα είναι ϕ, τότε η μερική πίεση των υδρατμών P υπολογίζεται από τη σχέση {2}.

1.2 Συμπύκνωση πάνω σε δομικές επιφάνειες

1.2.1 Σχηματισμός συμπυκνώσεων υδρατμών και αίτια

Υγροποίηση / συμπύκνωση των υδρατμών στην εσωτερική επιφάνεια κατασκευής συμβαίνει όταν η θερμοκρασία πάνω στην εσωτερική επιφάνειας των μελών της κατασκευής είναι μικρότερη από το σημείο δρόσου του αέρα στον εσωτερικό χώρο. Η υγρασία, η οποία αποτίθεται από τον αέρα στα μέλη της κατασκευής με μορφή σταγόνων ή μεμβράνης, ονομάζεται νερό συμπύκνωσης. Συμβαίνει συνήθως κατά τη χειμερινή περίοδο, σε θερμαινόμενους ή όχι χώρους.

Η χαμηλή θερμοκρασία της εσωτερικής επιφάνειας των δομικών στοιχείων, που πιθανόν να οδηγήσει σε εμφάνιση συμπυκνωμάτων, είναι συνάρτηση διαφόρων παραγόντων και μπορεί να οφείλεται σε:

- Χαμηλή εξωτερική θερμοκρασία, οπότε μπορεί να εμφανισθεί αντίστοιχα σχετικά χαμηλή θερμοκρασία και πάνω στην επιφάνεια των δομικών στοιχείων.

- Ανεπαρκή θερμομόνωση, οπότε λόγω μεγάλης θερμορροής δημιουργείται μεγάλη θερμοκρασιακή πτώση από τον αέρα του χώρου στην εσωτερική επιφάνεια του τοιχώματος.
- Αυξημένη υγρασία του αέρα του χώρου, οπότε το σημείο δρόσου του αέρα είναι αυξημένο και η επιφάνεια μπορεί «ευκολότερα» να το φθάσει.
- Ανεπαρκή αερισμό του χώρου, επειδή δεν απομακρύνονται οι εντός του χώρου παραγόμενοι υδρατμοί.
- Μικρή ή καθόλου κυκλοφορία αέρα κοντά στις επιφάνειες, οπότε, λόγω μικρού εσωτερικού συντελεστή θερμικής μετάβασης, η θερμοκρασία τους απέχει αρκετά από του χώρου.
- Θέρμανση ψυχθέντων χώρων, επειδή ο αέρας θερμαίνεται γρηγορότερα από τις επιφάνειες του κελύφους του χώρου και έτσι αυτές μπορούν να βρεθούν, για ορισμένο χρονικό διάστημα, σε θερμοκρασία χαμηλότερη του σημείου δρόσου του θερμού αέρα

Συνήθεις επικίνδυνες θέσεις για να δημιουργηθούν συμπυκνώσεις υδρατμών είναι:

- Γωνίες εξωτερικών τοίχων ή κολονών, ειδικά αν είναι χωρίς μόνωση.
- Χωρίς μόνωση γωνίες δαπέδου – τοίχου
- Σημεία διακοπής υπάρχουσας θερμομόνωσης, σε γωνίες ή σε επίπεδο
- Οροφές, ειδικά κοντά σε μπαλκόνια ή βεράντες χωρίς μόνωση
- Μη αεριζόμενες θέσεις (π.χ. κλειστές ντουλάπες, πίσω από κουρτίνες ή έπιπλα,
- κλειστά υπόγεια κλπ)
- Επιφάνειες κοντά σε θέσεις παραγωγής υδρατμών
  Επιφάνειες γυάλινων ή μεταλλικών ανοιγμάτων

1.2.2 Αποτροπή ή απομάκρυνση συμπυκνωμάτων

Η δημιουργία εσωτερικών συμπυκνώσεων υδρατμών θα αποφευχθεί, αν ληφθούν μέτρα ώστε να μην φθάνει η εσωτερική επιφάνεια των δομικών στοιχείων σε θερμοκρασία χαμηλότερη από το σημείο δρόσου του αέρα, ακόμη και σε δυσμενείς υγροθερμικές συνθήκες του αέρα. Επομένως, η αποτροπή σχηματισμού συμπυκνωμάτων μπορεί να επιτευχθεί με:

- Ισχυρή θερμομόνωση των δομικών στοιχείων, που είναι το πιο αποτελεσματικό μέτρο, αρκεί να γίνει σωστά ώστε να αποφεύγονται θερμογέφυρες.
- Επαρκή φυσικό αερισμό του χώρου, εφόσον η παραγωγή υδρατμών μέσα σ’ αυτόν δεν είναι υπερβολική και υπάρχει η δυνατότητα φυσικού αερισμού, με κατάλληλα ανοίγματα.
- Κατάλληλο τεχνητό εξαερισμό του χώρου, όταν: είτε η παραγωγή υδρατμών είναι αυξημένη λόγω της χρήσης του χώρου (π.χ. κουζίνες, λουτρά, χώροι συγκέντρωσης ατόμων κλπ) είτε δεν είναι εφικτός ή αποτελεσματικός ο φυσικός αερισμός λόγω διαμόρφωσης (π.χ. κλειστά υπόγεια ή WC, χώροι με λίγα ανοίγματα κλπ)

- Διάταξη επίπλων, κουρτινών, ντουλαπιών κλπ, τέτοια ώστε να αποφεύγονται – κατά το δυνατόν – θέσεις στάσιμου αέρα κοντά στα εξωτερικά τοιχώματα του χώρου.
- Αποφυγή απότομης θέρμανσης χώρων που έχουν παραμείνει σχετικά ψυχροί, ειδικά όταν χρησιμοποιείται θέρμανση με θερμό αέρα. Σ' αυτή την περίπτωση, καλό είναι η θέρμανση του ψυχρού χώρου να γίνει σταδιακά.

Αν σε κάποια επιφάνεια σχηματισθούν συμπυκνώματα, αυτά είναι δυνατόν, ανάλογα με την περίπτωση, να απομακρύνονται με:

- Διάχυσή τους στο περιβάλλον ή προς γειτονικούς χώρους, μέσω των δομικών στοιχείων.
- Απορροή τους, κυρίως από επιφάνειες ανοιγμάτων, μέσω κατάλληλων «νεροχυτών» ή συστημάτων απορροής προς τα έξω.
- Αερισμό του χώρου, γενικά ή τοπικά.

1.3 Μηχανισμός διάχυσης υδρατμών και η συμπεριφορά των υλικών

Τα συνήθη οικοδομικά υλικά, όσον αφορά τη συμπεριφορά τους σε σχέση με την υγρασία, μπορούν να χαρακτηρισθούν ως:

1.3.1 Υδατοπερατά

Είναι εκείνα που έχουν αρκετά μεγάλους πόρους ώστε να μπορούν να απορροφήσουν και να συγκρατήσουν μέσα στη μάζα τους ποσότητα νερού ή να μπορεί μέσω αυτών να διέλθει νερό (π.χ. γυψοκονιάματα, ινώδη ορυκτά, ινώδη ξύλα, υαλοβάμβακας κλπ). Όλα τα υλικά αυτά είναι εύκολα διαπερατά από υδρατμούς.

1.3.2 Υδατοστεγή

Υδατοστεγή υλικά είναι εκείνα των οποίων οι πόροι είναι πολύ μικροί ώστε να μην τους διαπερνά κανένα μόριο νερού, όπως π.χ. σκυρόδεμα, κεραμίδια, τούβλα κλπ. Ωστόσο, τα υλικά αυτά δεν είναι κατ' ανάγκην και στεγανά ως προς τους υδρατμούς, των οποίων τα μόρια (ως αέρια) μπορούν να διέλθουν μέσα από αυτούς τους πόρους. Πολλά από τα υλικά αυτά χαρακτηρίζονται και ως «φρένα υδρατμών», εφόσον είναι ελάχιστα διαπερατά από υδρατμούς, όπως π.χ. διάφορα βερνίκια ή χρώματα, πλαστικά, διάφορα ασφαλτόπανα και ασφαλτικές επιστρώσεις κλπ.

1.3.3 Υδρατμοστεγή ή φράγματα υδρατμών

Εκείνα τα υλικά που έχουν μικροσκοπικούς ή καθόλου πόρους και επομένως μέσα από τη μάζα τους δεν μπορεί να περάσουν καθόλου υδρατμοί. Τέτοια υλικά είναι τα τζάμια, το αφρώδες γυαλί, το αλουμίνιο, διάφορα πλαστικά φύλλα και μεμβράνες PVC και πολυαιθυλενίου, καθώς και ορισμένες συνθετικές στεγανωτικές επικαλύψεις.

1.4 Βασικές εξισώσεις και μεγέθη για τη διάχυση των υδρατμών

Οι υδρατμοί διέρχονται / διαχέονται μέσα από τη μάζα των δομικών στοιχείων, αφού τα περισσότερα από τα συνήθη υλικά έχουν μικρούς ή μεγάλους πόρους, μέσω των οποίων μπορούν να διέλθουν οι υδρατμοί. Απαραίτητη προϋπόθεση για να δημιουργηθεί διάχυση μέσα από τα διάφορα υλικά (εκτός από τα φράγματα υδρατμών) είναι να υπάρχει διαφορά πίεσης στις δύο εκτεθειμένες επιφάνειες του στοιχείου, ώστε οι υδρατμοί να μπορούν να μετακινηθούν από το χώρο όπου επικρατεί υψηλή (μερική) πίεση των υδρατμών προς το χώρο χαμηλότερης (μερικής) πίεσης υδρατμών.

Έτσι, η διάχυση (ή διαπίδυση) των υδρατμών είναι φαινόμενο ανάλογο της διάδοσης της θερμότητας μέσα από ένα δομικό τοίχωμα. Το ρόλο της απαιτούμενης «διαφοράς δυναμικού» παίζει η διαφορά πίεσης αντί της διαφοράς θερμοκρασίας και, βέβαια ,αφορά μεταφορά μάζας και όχι ενέργειας. Η κατεύθυνση ροής των υδρατμών δια μέσου ενός δομικού στοιχείου μπορεί, ανάλογα με τη διαφορά της τάσης υδρατμών, να είναι από μέσα προς τα έξω ή αντίστροφα ή από το τοίχωμα προς τα μέσα και προς τα έξω. Το μέγεθος που ενδιαφέρει τους υπολογισμούς είναι η ανά μονάδα χρόνου ροή μάζας υδρατμών m^3 που διαχέεται δια μέσου ενός τοιχώματος, σε $\left[\frac{kg}{h}\right]$. Επίσης, πολύ χρήσιμο μέγεθος είναι ο ανά μονάδα επιφάνειας ρυθμός ροής της μάζας των υδρατμών, δια μέσου ενός επιπέδου τοιχώματος, δηλαδή:

$$g = \frac{m}{A}\ \sigma\varepsilon[\frac{kg}{h * m^2}]$$

(Η ώρα [h] χρησιμοποιείται ως μονάδα χρόνου, επειδή αν χρησιμοποιείτο το δευτερόλεπτο [s] θα προέκυπταν πολύ μικρές αριθμητικά παροχές).

Η βασική σχέση που διέπει τη διάχυση των υδρατμών μέσα από ένα επίπεδο δομικό τοίχωμα (εφόσον αγνοηθούν οι τυχόν τριχοειδείς δυνάμεις που δημιουργούνται), είναι:

$$\text{m}=K_D * A * (P_i - P_e)$$

όπου:

- m: Ο ρυθμός ροής μάζας του υδρατμού, kg/h
- A: Επιφάνεια του δομικού στοιχείου, κάθετα στη ροή των υδρατμών, m^2
- P_i: Η μερική πίεση των υδρατμών εσωτερικά, Pa
- P_e: Η μερική πίεση των υδρατμών εξωτερικά, Pa
- K_D: Συντελεστής διαπερατότητας (ή διόδου) υδρατμών του τοιχώματος, σε $\left[\frac{\text{kg}}{\text{m}^2\text{hPa}}\right]$.

Ανά μονάδα επιφάνειας:

$$g = m/A = K_D * (P_i - P_e)$$

όπου g σε $[\frac{kg}{m^2 h}]$

Ο K_D είναι αντίστοιχος του συντελεστή θερμοπερατότητας του τοιχώματος. Για ένα τοίχωμα που αποτελείται από N στρώσεις, διαφόρων υλικών, προκύπτει από τη σχέση:

$$\frac{1}{K_d} = \frac{1}{\beta_1} + \frac{d_1}{\delta_1} + \frac{d_2}{\delta_2} + \cdots + \frac{d_n}{\delta_n} + \frac{1}{\beta_e}$$

όπου:

β_i και β_e : Συντελεστές υδρατμομετάβασης, στην εσωτερική και την εξωτερική επιφάνεια αντίστοιχα. Οι β_i και β_e είναι πολύ μεγάλοι, οπότε $1/\beta_i = 1/\beta_e \approx 0$)

d_j : Το πάχος της κάθε στρώσης (j), m

δ_j : Συντελεστής υδρατμοαγωγιμότητας του υλικού (j), σε $\left[\frac{\text{kg}}{\text{m.h.Pa}}\right]$.

Έτσι, η σχέση { g = m/A = $K_D * (P_i - P_e)$} απλοποιείται στην :

$$\frac{1}{K_D} = \sum \frac{d_j}{\delta_j}$$

Ο συντελεστής δ_j είναι αντίστοιχος του συντελεστή θερμικής αγωγιμότητας (σε φαινόμενα μετάδοσης θερμότητας) και εξαρτάται από την πίεση και τη θερμοκρασία. Ο συντελεστής αγωγιμότητας των υδρατμών δ_j ορίζεται ως η ποσότητα των υδρατμών σε kg, η οποία διέρχεται, λόγω διαπίδυσης (διάχυσης), σε 1 ώρα μέσα από στρώμα υλικού που έχει εμβαδόν 1 $[m^2]$και πάχος 1 [m], όταν η διαφορά των μερικών πιέσεων των υδρατμών μεταξύ των δύο επιφανειών του είναι 1 [Pa] και το σύστημα βρίσκεται σε μόνιμη κατάσταση, δηλαδή η μερική τάση των υδρατμών τοπικά παραμένει σταθερή με το χρόνο.

Για κάθε στρώση, ο λόγος $[\frac{d_j}{\delta_j}]$ είναι η αντίσταση διάχυσης των υδρατμών του j-στρώματος.

Σε πρακτικές εφαρμογές, αντί του συντελεστή d_j χρησιμοποιείται ο συντελεστής αντίστασης σε διάχυση των υδρατμών, **μ**, που ορίζεται ως ο αριθμός που δείχνει πόσες φορές μεγαλύτερη είναι η αντίσταση κατά τη διαπίδυση των υδρατμών μέσα από ένα στρώμα ομοιογενούς υλικού από την αντίσταση κατά τη διαπίδυση των υδρατμών μέσα από στρώμα αέρα ίσου πάχους, στις ίδιες συνθήκες περιβάλλοντος. Η αντίσταση διαπίδυσης των υδρατμών είναι καθαρός αριθμός. Η αντίσταση διαπίδυσης των υδρατμών, **μ**, συνδέεται με το συντελεστή αγωγιμότητας των υδρατμών, **δ**, με τη σχέση:

$$\delta = \frac{639 * 10^{-9}}{\mu}$$

όπου: ο **δ** σε $\left[\frac{kg}{m.h.Pa}\right]$ σε θερμοκρασία αναφοράς 0°C

ΠΑΡΑΤΗΡΗΣΗ: Συνήθως ο **μ** λαμβάνεται σταθερός, στη θερμοκρασία αναφοράς, ενώ ο **δ** εξαρτάται από τη θερμοκρασία και την πίεση. Το σφάλμα ωστόσο δεν είναι σημαντικό και κυμαίνεται από -6% για -20°C έως +6% για +20°C.

Για κάθε στρώση, λαμβάνοντας υπόψη την σχέση { δ}, η αντίσταση διάχυσης των υδρατμών γίνεται:

$$\frac{d}{\delta} = \frac{d * \mu}{636 * 10^{-9}} = 10^6 * 1.57 * d * \mu$$

Έτσι, η $\{\frac{1}{K_D} = \sum \frac{d_j}{\delta_j}\}$ γράφεται :

$$\frac{1}{K_D} = 10^6 * 1.57 * \sum (d_j * \mu_j)$$

και η ανά μονάδα επιφάνειας ροή των υδρατμών δίδεται από την:

$$\mathrm{g} = \frac{m}{A} = \frac{(P_i - P_e)}{1.57 \cdot 10^6 \cdot \sum (d_j \cdot \mu_j)}$$

1.5 Δημιουργία συμπυκνωμάτων μέσα σε δομικά στοιχεία

Εξετάζοντας την περίπτωση λειτουργίας ενός χώρου το χειμώνα («περίοδος δρόσου»), η διάδοση θερμότητας από μέσα προς τα έξω μέσα από ένα δομικό τοίχωμα συνοδεύεται και από διάχυση υδρατμών κατά την ίδια κατεύθυνση, εάν και εφόσον η (μερική) πίεση των υδρατμών είναι μεγαλύτερη στον θερμό χώρο από την αντίστοιχη (μερική) πίεση των υδρατμών στο κρύο εξωτερικό περιβάλλον. Αυτό πράγματι συμβαίνει σχεδόν πάντα (αν εξαιρέσει κανείς τις μη συνήθεις περιπτώσεις να υπάρχει πολύ χαμηλή σχετική υγρασία εσωτερικά και πολύ υψηλή σχετική υγρασία στο περιβάλλον, οπότε οι τιμές των πιέσεων μπορεί να είναι αντίστροφες). Οι υδρατμοί, «ταξιδεύοντας» στο εσωτερικό του δομικού στοιχείου, συναντούν στρώσεις υλικών διαφορετικής θερμοκρασίας την οποία και θεωρείται ότι αποκτούν.

Όταν στο εσωτερικό της κατασκευής η θερμοκρασία είναι μικρότερη από τη θερμοκρασία κορεσμού των υδρατμών, $[\Theta_k]$, τότε γίνεται υγροποίηση των υδρατμών, με αποτέλεσμα να υγρανθεί η κατασκευή σε κάποια σημεία ή και περιοχές. Η υγρασία (συμπυκνώματα) μέσα σε κάποιο δομικό στοιχείο μειώνει την θερμομονωτική ικανότητά της κατασκευής και μπορεί να προκαλέσει βλάβη ή φθορά στο ίδιο το υλικό ή διαβρώσεις και σκουριές σε υλικά. Ακόμη (σε περίπτωση χαμηλών θερμοκρασιών λειτουργίας) μπορεί το νερό να μετατραπεί σε πάγο, με καταστροφικά πολλές φορές αποτελέσματα λόγω των αναπτυσσόμενων δυνάμεων, αφού το νερό όταν στερεοποιείται διαστέλλεται περίπου κατά 10%.

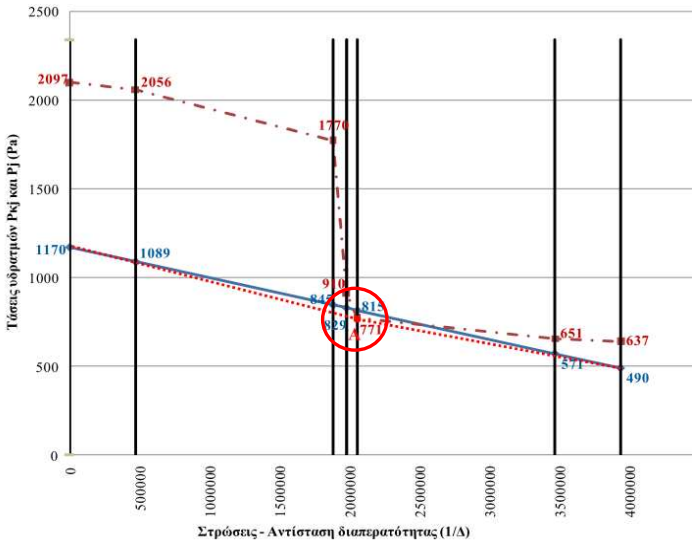
Για τον έλεγχο της υγροποίησης των υδρατμών μέσα στο εσωτερικό της κατασκευής πρέπει να προσδιοριστούν τόσο οι επικρατούσες μερικές τάσεις των υδρατμών, P, όσο και οι αντίστοιχες τάσεις των κορεσμένων υδρατμών, .

Τα κριτήρια, που σχετίζονται με τη δημιουργία συμπυκνωμάτων μέσα σε δομικό στοιχείο, ώστε ή να αποτραπούν τελείως ή να αποφευχθούν ζημιογόνα αποτελέσματα εφόσον δημιουργηθούν, είναι:

- Αποκλεισμός συμπύκνωσης σε ευπαθείς στρώσεις.
- Η μέγιστη επιτρεπόμενη (συγκρατούμενη) υγρασία, σε κάποια στρώση, στο τέλος της χειμερινής περιόδου, δηλ. κατά την «περίοδο δρόσου».
- Πλήρης εξάτμιση της συγκρατούμενης στην «περίοδο δρόσου» υγρασίας κατά τη διάρκεια του καλοκαιριού («περίοδος ξήρανσης»).

Η μελέτη για το αν τηρούνται και σε ποιο βαθμό τα παραπάνω κριτήρια, μπορεί να γίνει, με αρκετά ακριβή αποτελέσματα, χρησιμοποιώντας τη γραφοαναλυτική μέθοδο του Glaser.

Η μέθοδος στηρίζεται στην κατασκευή ενός διαγράμματος (διάγραμμα Glaser), στο οποίο χαράσσονται οι μεταβολές (καμπύλες) της τάσης κορεσμένων υδρατμών, , και της μερικής πίεσης των υδρατμών, , σε όλες τις διαχωριστικές επιφάνειες των στρώσεων του τοιχώματος. Οι στρώσεις του δομικού στοιχείου παριστάνονται στο διάγραμμα, υπό κλίμακα, με την αντίσταση διάχυσης των υδρατμών της κάθε στρώσης, στον οριζόντιο άξονα.

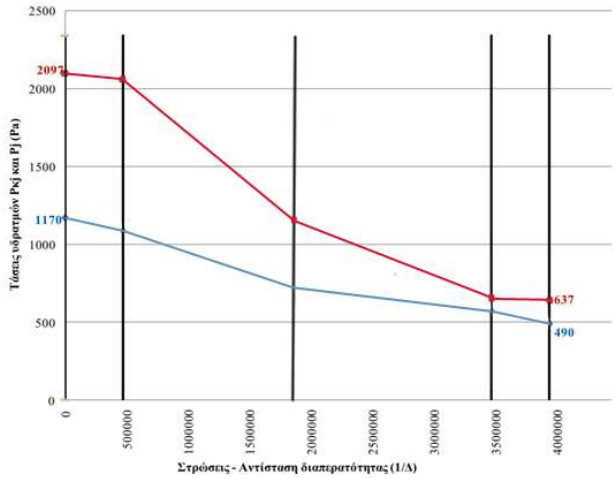


Σχήμα 1: Διάγραμμα Glaser με προβληματικό σημείο που δημιουργείται συμπύκνωση υδρατμών

Αν η καμπύλη[και η [P] δεν έχουν κοινό σημείο, τότε δεν υπάρχει κίνδυνος συμπύκνωσης μέσα στο δομικό στοιχείο. Πρέπει, δηλαδή, σ' όλο το πάχος του τοιχώματος να είναι P < , διαφορετικά θα υπάρχει σχηματισμός συμπυκνωμάτων σε κάποια θέση ή περιοχή του δομικού στοιχείου.

Οι τιμές της $[P_{kj}]$, εξαρτώνται, όπως έχει προαναφερθεί, από τη θερμοκρασία και επομένως είναι απαραίτητο να εκτελεσθούν πρώτα θερμικοί υπολογισμοί για να προκύψει η θερμοκρασιακή κατανομή μέσα στο τοίχωμα. Οι τιμές της επικρατούσας σε κάθε θέση – στρώση του τοιχώματος μερικής πίεσης των υδρατμών $[P_{\ j}]$, βρίσκονται υπολογίζοντας την πτώση τάσης υδρατμών $[\Delta P_{\ j}]$, που δημιουργεί κάθε στρώση (j), αφού για την διαχεόμενη παροχή υδρατμών θα ισχύει:

$$g = K_D * (P_i - P_e) = \frac{\delta_j}{d_j} * (P_j - P_{j+1}) \quad \Longrightarrow \quad \Delta P_j = \frac{d_j}{\delta_j} * K_D * \Delta P$$



Σχήμα 2:Διάγραμμα Glaser σε τοίχωμα που δεν δημιουργείται συμπύκνωση υδρατμών

1.6 Μέτρα αποφυγής υγροποιήσεων στο εσωτερικό δομικού στοιχείου

Στην περίπτωση που είναι πιθανή η υγροποίηση των υδρατμών στο εσωτερικό κατασκευής, για να αποφευχθεί αυτό το φαινόμενο, πρέπει να τοποθετηθεί ένα φράγμα υδρατμών. Τα φράγματα υδρατμών έχουν πολύ υψηλές τιμές της αντίστασης διαπίδυσης των υδρατμών, μ, που είναι δυνατόν να φτάσουν μέχρι και 10^5.

Το φράγμα των υδρατμών προκαλεί απότομη και σημαντική πτώση της μερικής τάσης των υδρατμών και συμβάλλει ώστε η μερική τάση των υδρατμών να διατηρείται μικρότερη από την αντίστοιχη τάση των κορεσμένων υδρατμών και συνεπώς να μην γίνεται υγροποίησή τους στο εσωτερικό της κατασκευής. Το φράγμα υδρατμών τοποθετείται πριν από τη στρώση στην οποία (μπορεί να) γίνεται η υγροποίηση των υδρατμών, κατά την κατεύθυνση της διαπίδυσης των υδρατμών. Δηλαδή, θα πρέπει να τοποθετείται προς τη θερμή πλευρά του τοιχώματος.

Ως φράγματα υδρατμών χρησιμοποιούνται πλαστικοποιημένες μεμβράνες από άσφαλτο ή πίσσα, ασφαλτόχαρτα, πισσόχαρτα, ασφαλτόπανα, φύλλα από αλουμίνιο, φύλλα από πολυαιθυλένιο, πλαστικά υλικά με τη μορφή λωρίδων και πλαστικά χρώματα, μη υδατοπερατά.

Σε περιπτώσεις όπου ο κίνδυνος δημιουργίας υγροποιήσεων είναι μικρός, όταν δηλαδή σε κάποιο σημείο του τοιχώματος είναι $P_u > P_{uk}$, αλλά η διαφορά είναι μικρή, τότε είναι πιθανόν να εξαλειφθεί αυτή η πιθανότητα και να μην υπάρξει κανένα πρόβλημα, με αλλαγή της διάταξης / σειράς τοποθέτησης των στρώσεων (εφόσον αυτό, βέβαια, είναι εφικτό).

Γενικό κατασκευαστικό αξίωμα είναι: «Η αντίσταση διάχυσης των υδρατμών πρέπει να μειώνεται από μέσα προς τα έξω».

2 Προγραμματισμός

2.1 Τι είναι ο προγραμματισμός

Προγραμματισμός είναι η διαδικασία του σχεδιασμού, συγγραφής, δοκιμής, και αποσφαλμάτωσης του κώδικα που απαρτίζει ένα πρόγραμμα. Ο κώδικας αυτός γράφεται σε μια γλωσσά προγραμματισμού. Ο σκοπός του προγραμματισμού είναι η δημιουργία προγραμμάτων που εκτελούν μια σειρά από λειτουργιές και έχουν μια συγκεκριμένη προβλεπόμενη συμπεριφορά. Η διαδικασία της συγγραφής ενός προγράμματος, συχνά απαιτεί τεχνογνωσία σε αρκετά διαφορετικά θέματα.

2.2 Κατηγορίες προγραμματισμού

Οι δυο πιο διαδεδομένες κατηγορίες προγραμματισμού σήμερα είναι ο δομημένος και ο αντικειμενοστραφής προγραμματισμός (object Oriented Programming).

2.2.1 Δομημένος προγραμματισμός

Στην επιστήμη υπολογιστών δομημένος προγραμματισμός (structured programming) ή διαδικαστικός προγραμματισμός (procedural programming) είναι μία προσέγγιση στον προγραμματισμό, η οποία βασίζεται στην έννοια της κλήσης διαδικασίας. Η διαδικασία, γνωστή επίσης και ως ρουτίνα, υπορουτίνα, μέθοδος ή συνάρτηση (δεν σχετίζεται άμεσα με τη μαθηματική έννοια της συνάρτησης), είναι απλά ένα αυτοτελές σύνολο εντολών προς εκτέλεση.

Ο δομημένος προγραμματισμός βασίζεται στην αρχή του διαίρει και βασίλευε, καθώς διασπά το βασικό πρόβλημα σε μικρότερα υποπροβλήματα (γνωστά επίσης και ως εργασίες). Κάθε εργασία με πολύπλοκη περιγραφή διαιρείται σε μικρότερες, έως ότου οι εργασίες να είναι αρκετά μικρές, περιεκτικές και εύκολες προς κατανόηση.

Ιστορικά ο δομημένος προγραμματισμός αναπτύχθηκε ύστερα από έρευνα κατά τη δεκαετία του 1960, ως βελτίωση του ήδη υπάρχοντος διαδικαστικού προγραμματισμού. Ένα από τα πιο σημαντικά αποτελέσματα αυτής της έρευνας ήταν η ανάπτυξη της γλώσσας Pascal (γλώσσα προγραμματισμού), από τον Νίκλαους Βιρτ (Niklaus Wirth) το 1971, η οποία σύντομα έγινε η προτιμώμενη γλώσσα διδασκαλίας σε πολλά πανεπιστήμια. Η έννοια της διαδικασίας επομένως ήταν προϋπάρχουσα αλλά δεν έπαιζε τόσο σημαντικό ρόλο στη δομή των υπό συγγραφή εφαρμογών, καθώς τα δεδομένα ήταν αρκετά διαχωρισμένα από τις διαδικασίες και έπρεπε ο προγραμματιστής να θυμάται για κάθε διαδικασία ποια άλλη καλούσε, αλλά και ποια δεδομένα διαφοροποιούνταν. Καθώς όμως οι περισσότερες διαδικαστικές γλώσσες γρήγορα υιοθέτησαν στοιχεία ώστε να υποστηρίζουν δομημένο προγραμματισμό, οι δύο όροι σήμερα έχουν πρακτικώς ταυτιστεί. Με τον καιρό οι δομημένες γλώσσες έφτασαν να μην επαρκούν για τη συγγραφή προγραμμάτων, επεκτάθηκαν και ως λύση υιοθετήθηκε ο αντικειμενοστραφής προγραμματισμός.

2.2.2 Αντικειμενοστραφής προγραμματισμό (object-oriented programming)

Στην επιστήμη υπολογιστών αντικειμενοστραφή προγραμματισμό (object-oriented programming), ή ΑΠ, ονομάζουμε ένα προγραμματιστικό υπόδειγμα το οποίο εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990, αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού. Πρόκειται για μία μεθοδολογία ανάπτυξης προγραμμάτων, υποστηριζόμενη από κατάλληλες γλώσσες προγραμματισμού, όπου ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται αντικείμενο και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη, τύπου δεδομένων ονόματι κλάση. Η πρωταρχική καινοτομία του ΑΠ έγκειται στο ότι μια κλάση προδιαγράφει τόσο δεδομένα όσο και τις διαδικασίες οι οποίες επιδρούν επάνω τους.

Έτσι μπορεί να οριστεί μία προδιαγραφή δομής αποθήκευσης (π.χ. μία κλάση "τηλεόραση") η οποία να περιέχει τόσο ιδιότητες (π.χ. μία μεταβλητή "τρέχον κανάλι") όσο και πράξεις ή χειρισμούς επί αυτών των ιδιοτήτων (π.χ. μία διαδικασία "άνοιγμα της τηλεόρασης"). Στο εν λόγω παράδειγμα κάθε υλική τηλεόραση (κάθε αντικείμενο αποθηκευμένο πραγματικά στη μνήμη) αναπαρίσταται ως ξεχωριστό, "φυσικό" στιγμιότυπο αυτής της πρότυπης, ιδεατής κλάσης. Επομένως μόνο τα αντικείμενα καταλαμβάνουν χώρο στη μνήμη του υπολογιστή ενώ οι κλάσεις αποτελούν απλώς "καλούπια". Οι αιτίες που ώθησαν στην ανάπτυξη του ΑΠ ήταν οι ίδιες με αυτές που οδήγησαν στην ανάπτυξη του δομημένου προγραμματισμού (ευκολία συντήρησης, οργάνωσης, χειρισμού και επαναχρησιμοποίησης κώδικα μεγάλων και πολύπλοκων εφαρμογών), όμως τελικώς η αντικειμενοστρέφεια επικράτησε καθώς μπορούσε να αντεπεξέλθει σε προγράμματα πολύ μεγαλύτερου όγκου και πολυπλοκότητας.

Οι περισσότερες αντικειμενοστραφείς έννοιες εμφανίστηκαν αρχικά στη γλώσσα προγραμματισμού Simula 67, η οποία ήταν προσανατολισμένη στην εκτέλεση προσομοιώσεων του πραγματικού κόσμου. Οι ιδέες της Simula 67 επηρέασαν κατά τη δεκαετία του '70 την ανάπτυξη της Smalltalk, της γλώσσας 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. Όταν η τελική έκδοση της Smalltalk έγινε διαθέσιμη το 1980 η έρευνα για την αντικατάσταση του δομημένου προγραμματισμού με ένα πιο σύγχρονο υπόδειγμα ήταν ήδη εν εξελίξη.

Την ίδια περίπου εποχή, και επίσης με επιρροές από τη Simula, ολοκληρωνόταν η ανάπτυξη της C++ ως μίας ισχυρής επέκτασης της δημοφιλούς γλώσσας προγραμματισμού C στην οποία είχαν "μεταμοσχευθεί" αντικειμενοστραφή χαρακτηριστικά. Η επιρροή της C++ καθ' όλη της δεκαετία του '80 ήταν καταλυτική με αποτέλεσμα τη σταδιακή κυκλοφορία αντικειμενοστραφών εκδόσεων πολλών γνωστών διαδικαστικών γλωσσών προγραμματισμού. Κατά το πρώτο ήμισυ της δεκαετίας του '90 η βαθμιαία καθιέρωση

στους μικροϋπολογιστές των γραφικών διασυνδέσεων χρήστη (GUI), για την ανάπτυξη των οποίων ο ΑΠ φαινόταν ιδιαιτέρως κατάλληλος, και η επίδραση της C++ οδήγησαν στην επικράτηση της αντικειμενοστρέφειας ως βασικού προγραμματιστικού υποδείγματος.

Το 1995 η εμφάνιση της Java, μίας ιδιαίτερα επιτυχημένης, πλήρως αντικειμενοστραφούς γλώσσας που έμοιαζε συντακτικώς με τη C/C++ και προσέφερε πρωτοποριακές για την εποχή δυνατότητες, έδωσε νέα ώθηση στον ΑΠ. Παράλληλα εμφανίστηκαν ποικίλες άτυπες βελτιώσεις στο βασικό προγραμματιστικό υπόδειγμα, όπως οι αντικειμενοστραφείς γλώσσες μοντελοποίησης λογισμικού, τα σχεδιαστικά πρότυπα κλπ. 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 αντικείμενοστρέφεια.

2.3 Η γλωσσά προγραμματισμού java

Η γλωσσά προγραμματισμού java, προδιαγράφηκε και υλοποιήθηκε από την εταιρία Sun Microsystems, και σχεδιάστηκε να είναι ανεξάρτητη λειτουργικού συστήματος (δηλαδή να μπορεί να τρέχει χωρίς τροποποιήσεις σε διαφορετικές πλατφόρμες), και να είναι ταυτόχρονα αρκετά ασφαλής για την χρήση της σε κατασκευή δικτυακών προγραμμάτων και αρκετά δυνατή για να αντικαταστήσει την κλασική γλωσσά μηχανής.

Αρχικά, ο περισσότερος ενθουσιασμός γύρω από την java στρέφονταν γύρω από τις δυνατότητες την να κατασκευάζει εφαρμογές για χρήση στο internet, που ονομάζονται applets. Εκείνη την εποχή, η χρήση της σε εφαρμογές εκτός από applet ήταν το λιγότερο περιορισμένη. Σήμερα η java έχει την βιβλιοθήκη swing, ένα από τα πιο πλήρη εργαλεία για την κατασκευή γραφικών περιβαλλόντων που υπάρχει στις γλώσσες προγραμματισμού. Αυτή η προσθήκη την έχει κάνει μια από τις πιο δημοφιλής γλώσσες για την ανάπτυξη εφαρμογών.

Ο κώδικας java μετατρέπεται σε άπλες δυαδικές εντολές, περίπου όπως και η γλωσσά μηχανής ενός κλασικού μικροεπεξεργαστή. Όμως σε αντίθεση με την C όπου ο κώδικας της μετατρέπεται σε δυαδικές εντολές προορισμένες για να τρέξουν μόνο σε ένα συγκεκριμένο μοντέλο επεξεργαστή, ο κώδικας της java μετατρέπεται σε εντολές προορισμένες να τρέξουν σε έναν εικονικό επεξεργαστή. Αυτές οι εντολές μετά εκτελούνται από έναν μεταγλωττιστή java. Αυτός ο μεταγλωττιστής κάνει όλες τις εργασίες ενός κλασικού επεξεργαστή , αλλά τις κάνει σε ένα ασφαλές, εικονικό περιβάλλον.

Ο μεταγλωττιστής της java είναι σχετικά ελαφρύς σε απαιτήσεις, και μικρός σε μέγεθος. Μπορεί να μετατραπεί σε ότι μορφή είναι απαραίτητο για την χρήση του σε κάποια πλατφόρμα (κινητό τηλέφωνο, λειτουργικό σύστημα υπολογιστή και άλλα.) Ο μεταγλωττιστής μπορεί να τρέξει σαν αυτόνομο λογισμικό ή μπορεί να ενωθεί με αλλά λογισμικά, όπως έναν φύλλο-μετρητή. Αυτό σημαίνει γενικά ότι ένα πρόγραμμα που έχει γραφτεί σε java μπορεί να τρέξει σχεδόν παντού, σε κάθε πλατφόρμα που έχει συμβατό μεταγλωττιστή java. Δεν χρειάζεται διαφορετική έκδοσης μιας εφαρμογής για κάθε πλατφόρμα. Αυτοί είναι και οι λόγοι γενικά που το πρόγραμμα μας επιλέξαμε να γραφτεί σε java.

3 Βασικά στοιχεία της γλώσσας JAVA

3.1 Σχόλια

3.1.1 Γενικά

Η java υποστηρίζει σχόλια στον κώδικα της, αυτής της μορφής.

```
/* εδώ είναι σχόλια
 Πολλών γραμμών  */
```

```
// εδώ είναι σχόλιο μιας γραμμής.
```

Τα σχόλια πολλών γραμμών έχουν χαρακτήρες που δηλώνουν την αρχή και το τέλος τους. Δεν μπορούν να ομαδοποιηθούν βέβαια, δηλαδή δεν μπορείς να έχεις ένα σχόλιο πολλαπλής γραμμή μέσα σε ένα άλλο σχόλιο πολλαπλής γραμμής γιατί ο μεταγλωττιστής θα μπερδευτεί. Τα σχόλια μιας γραμμής είναι χρήσιμα για μικρά σχόλια μέσα στης μεθόδους. Δεν υπάρχει πρόβλημα να χρησιμοποιηθούν μαζί με τα σχόλια πολλών γραμμών.

3.1.2 Σχόλια javadoc

Ένα σχόλιο πολλαπλών γραμμών που ξεκινά με το «/**» υποδηλώνει μια ειδική κατηγορία σχόλιων που ονομάζονται javadoc σχόλια. Ένα σχόλιο javadoc είναι σχεδιασμένο για να χρησιμοποιηθεί από το πρόγραμμα javadoc. Αυτό το πρόγραμμα χρησιμοποιείται για την αυτόματη δημιουργία τεκμηρίωσης για την χρήση κάθε μεθόδου. Η βιβλιοθήκη τεκμηρίωσης της java έχει δημιουργηθεί εξολοκλήρου από αυτό το πρόγραμμα.

Μέσα στο σχόλιο, οι γραμμές που ξεκινάνε με «@» αναλύονται σαν ειδικές οδηγίες προς το πρόγραμμα javadoc, δίνοντας του πληροφορίες σχετικά με τον κώδικα της μεθόδου που αντιπροσωπεύει αυτό το σχόλιο, δηλαδή τι δέχεται σαν είσοδο αυτή η μέθοδος, τι επιστρέφει κτλ.

3.2 Τύποι μεταβλητών

3.2.1 Γενικά.

Το σύστημα των τύπων σε μια γλωσσά προγραμματισμού, περιγράφει πως τα στοιχεία δεδομένων του (οι μεταβλητές και οι σταθερές), αποθηκεύονται στην μνήμη, και την σχέση που έχουν μεταξύ τους. Σε στατικές γλώσσες προγραμματισμού, όπως η c/c++, ο τύπος ενός στοιχείου είναι ένα απλό και αμετάβλητο χαρακτηριστικό. Σε πιο δυναμικές γλώσσες, όπως η lisp, οι μεταβλητές μπορούν να αποθηκεύσουν αόριστα στοιχεία και γενικά να αλλάξουν τον τύπο τους κατά βούληση. Το αρνητικό των δυναμικών γλωσσών είναι ότι την ώρα τις εκτέλεσης της εφαρμογής χρησιμοποιούν επεξεργαστική ισχύ για την ανάλυση του τι γίνεται κάθε στιγμή με τον τύπο της κάθε μεταβλητής.

Η java συνδυάζει τα καλύτερα χαρακτηριστικά και από τις στατικές γλώσσες και από τις δυναμικές γλώσσες. Όπως και σε μια στατική γλωσσά, κάθε μεταβλητή και προγραμματιστικό στοιχεία στην java έχει έναν τύπο που είναι γνωστά την ώρα της μεταγλώττισης, ώστε κατά την εκτέλεση μιας εφαρμογής να μην σπαταλούνται κύκλοι του

επεξεργαστή για τους τύπους. Όμως παρόλα αυτά η java κρατάει πληροφορίες κατά την εκτέλεση για τα αντικείμενα της και τις χρησιμοποιεί για να επιτρέπει δυναμική συμπεριφορά. Ο κώδικας της java μπορεί να φορτώσει και να δημιουργήσει νέους τύπους και αντικείμενα κατά την εκτέλεση και να τους χρησιμοποιήσει.

Οι τύποι στην java ανήκουν σε 2 κατηγορίες. Τους πρωτογενείς τύπους που αντιπροσωπεύουν απλές τιμές. Είναι σταθερά στοιχεία , όπως σταθερές, ή νούμερα, και τους τύπους αναφοράς (ή τύπους κλάσης), που περιλαμβάνουν αντικείμενα και πίνακες.

3.2.2 Πρωτογενείς τύποι

Τα νούμερα, οι χαρακτήρες, και οι τιμές Boolean (αληθής, ψευδής), είναι πρωτογενείς στοιχεία στην java. Αντιθέτως με ορισμένες άλλες αντικειμενοστραφείς γλώσσες προγραμματισμού, δεν είναι αντικείμενα. Για αυτές τις περιπτώσεις που είναι επιθυμητό να χρησιμοποιηθεί ένα βασικό στοιχειό σαν αντικείμενο υπάρχουν κλάσεις που προσφέρουν αυτήν την δυνατότητα.

Ένα χαρακτηριστικό της java είναι ότι οι πρωτογενείς τύποι είναι αυστηρά ορισμένοι. Το μέγεθος τους είναι το εξής:

Τύπος	**Τιμή**
Boolean	True / false
Char	16-bit, Unicode character
byte	8-bit, signed, two's complement integer
short	16-bit, signed, two's complement integer
int	32-bit, signed, two's complement integer
long	64-bit, signed, two's complement integer
float	32-bit, IEEE 754, floating-point value

3.2.3 Δήλωση μεταβλητών

Οι μεταβλητές δηλώνονται μέσα σε μεθόδους ή κλάσεις, με την λέξη κλειδί του τύπου, ακολουθουμένη από ένα ή περισσότερα ονόματα χωρισμένα με κόμμα μεταξύ τους.

```
int foo;
double d1, d2;
boolean isFun;
```

Υπάρχει η δυνατότητα μαζί με την δήλωση τους να τους αντιστοιχίσουμε και μια τιμή.

```
int foo = 42;
double d1 = 3.14, d2 = 2 * 3.14;
boolean isFun = true;
```

Οι μεταβλητές που ορίζονται σαν μελή μιας κλάσης θέτονται στις προκαθορισμένες τιμές τους αν δεν πάρουν εναλλακτική τιμή μέσω ενός δομητή (constructor). Οι προκαθορισμένες τιμές τους είναι: 0 για τους αριθμητικούς τύπους, null για τους χαρακτήρες (\0), και οι Boolean μεταβλητές παίρνουν την τιμή false. Οι τοπικές μεταβλητές όμως, που δηλώνονται μέσα σε μια μέθοδο πρέπει οπωσδήποτε να αποκτήσουν τιμή πριν χρησιμοποιηθούν.

Όταν μια μεταβλητή αριθμητικού τύπου χρησιμοποιείται σε μια πράξη με μια άλλη μεταβλητή μεγαλυτέρου αριθμητικού τύπου, το αποτέλεσμα μετατρέπεται αυτόματα στον μεγαλύτερο τύπο.

```
byte b = 42;
int i = 43;
int result = b * i; // b μετατρεπετε σε int πριν τον πολλαπλασιασμο
```

Αυτό δεν μπορεί να γίνει αντιστρόφως αυτόματα. Βέβαια:

```
int i = 13;
byte b = i;     // Compile-time error, Λαθος
byte b = (byte) i;  // Αυτό δουλευει.
```

Οι μετατροπές από αριθμούς κινητής υποδιαστολής σε ακέραιους χρειάζονται μια ειδική εντολή λόγο του με την μετατροπή χάνουμε ακρίβεια.

3.2.4 Reference Types

Σε μια αντικειμενοστραφή γλωσσά προγραμματισμού όπως η java, δημιουργούνται νέοι πολύπλοκοι τύποι δεδομένων κάθε φορά που δημιουργείται μια κλάση. Κάθε κλάση παίρνει τον ρόλο ενός νέου τύπου δεδομένων στην γλωσσά. Για παράδειγμα, αν δημιουργήσουμε μια νέα κλάση που ονομάζεται «Car», ταυτόχρονα δημιουργούμε και έναν νέο τύπο δεδομένων που ονομάζεται «car». Ο τύπος ενός αντικείμενου καθορίζει γενικά πως χρησιμοποιείται και που μπορεί να αντιστοιχιστεί. Όπως και με τους πρωτογενείς τύπους, ένα αντικείμενο τύπου «car» να αντιστοιχιστεί σε μια μεταβλητή τύπου «car», ή να περαστεί σε μια μέθοδο που δέχεται σαν είσοδο τιμές τύπου «car»

```
Private void method (double aDouble,Car aCar)
{
  ……
}
```

Ένας τύπος δεν είναι μια απλή ιδιότητα. Οι κλάσεις πολλές φορές σχετίζονται με άλλες κλάσεις και το ίδιο κάνουν αντίστοιχα και οι τύποι δεδομένων που αντιπροσωπεύουν. Όλες οι κλάσεις υπακούουν μια ιεραρχία "**υπέρ-υπό**" οπού η **υπο**κλάση είναι μια ειδική περίπτωση της **υπερ**κλάσης. Οι αντίστοιχοι τύποι έχουν την ίδια σχέση, οπού ο τύπος της υποκλάσης θεωρείται ειδική περίπτωση του τύπου της υπερκλάσης. Ένα αντικείμενο της υποκλάσης μπορεί λοιπόν να χρησιμοποιηθεί στην θέση ενός αντικείμενου από την υπέρ-κλάση Για παράδειγμα αν φτιάξουμε μια νέα κλάση τύπου «γάτα», που είναι υποκλάση

μιας κλάσης «ζώο», τα αντικείμενα τύπου «γάτα» μπορούν να χρησιμοποιηθούν οπουδήποτε θα μπορούσαν να χρησιμοποιηθούν τα αντικείμενα τύπου «ζώο».

Οι πρωτογενείς τύποι στην java χρησιμοποιούνται και μεταφέρονται "by value". Με άλλα λόγια όταν ένας πρωτογενής τύπος αντιστοιχείται σε μια μεταβλητή, ή περνά σαν είσοδος σε μια μέθοδο, άπλα αντιγράφεται. Αντιθέτως τα αντικείμενα πάντα χρησιμοποιούνται και μεταφέρονται "by reference". Μια αναφορά είναι απλά το όνομα του αντικείμενου. Το τι μια μεταβλητή τύπου reference κρατάει, είναι ένας δείκτης σε ένα αντικείμενο αυτού του τύπου. Όταν το αντικείμενο περνά σε μια μέθοδο, μονό ο δείκτης αντιγράφεται και όχι το αντικείμενο που αντιπροσωπεύει ο δείκτης.

3.2.5 Συμβολοσειρές.

Οι συμβολοσειρές στην Java είναι αντικείμενα. Είναι δηλαδή τύπου αναφοράς. Βέβαια αντικείμενα συμβολοσειρών έχουν «βοήθεια» από τον μεταγλωττιστή, ώστε να φέρονται πιο πολύ σαν πρωτογενείς τύποι. Οι συμβολοσειρές στον κώδικα της JAVA μετατρέπονται αυτόματα από τον μεταγλωττιστή σε αντικείμενα συμβολοσειρών. Μπορούν να χρησιμοποιηθούν κατευθείαν, πχ περνώντας σαν είσοδοι σε μεθόδους.

```
System.out.println( "Hello, World..." );
String s = "I am the walrus...";
String t = "John said: \"I am the walrus...\"";
```

Το σύμβολο «+» στην JAVA είναι υπερφορτωμένο ώστε να μπορεί να χρησιμοποιηθεί για την ένωση συμβολοσειρών όσο και για τις κλασικές αριθμητικές πράξεις. Μαζί με το σύμβολο += είναι οι μόνοι υπερφορτωμένοι τελεστές στην Java

```
String quote = "Four score and " + "seven years ago,";
String more = quote + " our" + " fathers" + " brought...";
```

Η JAVA φτιάχνει ένα αντικείμενο συμβολοσειράς που αποτελείται από όλες τις συμβολοσειρές που προσθέτονται.

3.2.6 Πίνακες

Ένας πίνακας είναι μια ειδική μορφή αντικείμενου που μπορεί να αποθηκεύσει μέσα του μια οργανωμένη συλλογή από στοιχεία δεδομένων διαφορών τύπων. Ο τύπος δεδομένων των στοιχείων του πινάκα είναι και ο τύπος του πινάκα. Ο αριθμός των στοιχείων που κρατάει είναι μια σταθερή ιδιότητα του που ονομάζεται το μήκος του.

Τα αντικείμενα πινάκα διαφέρουν από τους άλλους τύπους αντικειμένων στην JAVA σε 3 χαρακτηριστικά:

- Η JAVA δημιουργεί μια ειδική κλάση τύπου πινάκα για μας κάθε φόρα που δηλώνουμε έναν ένα τύπο πινάκα.
- Η JAVA μας αφήνει να χρησιμοποιήσουμε τα σύμβολα [] για να προσπελάσουμε τα στοιχεία ενός πινάκα. Θα μπορούσαμε να κατασκευάσουμε τις δίκες μας κλάσεις που συμπεριφέρονται σαν πίνακες άλλα θα έπρεπε να ανεχτούμε το γεγονός ότι θα έπρεπε να έχουμε μεθόδους όπως get() & set() για να πάρουμε τα στοιχεία αντί να

χρησιμοποιούμε τα προκαθορισμένα αυτά σύμβολα. Η JAVA προσφέρει μια ειδική μορφή του δομητή ώστε να μας αφήνει να κατασκευάσουμε ένα στιγμιότυπο πινάκα, με το μέγεθος που θέλουμε χρησιμοποιώντας τα σύμβολα [], ή να δεχτεί κατευθείαν δεδομένα από μια λίστα με τιμές

Το μέγεθος ενός πινάκα μπορεί να βρεθεί χρησιμοποιώντας την μέθοδο length

```
char [] alphabet = new char [26];
int alphaLen = alphabet.length;       // alphaLen == 26
```

```
String [] musketeers = { "one", "two", "three" };
int num = musketeers.length;        // num == 3
```

Αυτό είναι ιδιαιτέρως χρήσιμο κυρίως αν χρησιμοποιηθούν πίνακες οπού το μέγεθος τους δεν είναι γνωστό από την αρχή.

Το μέγεθος του πινάκα είναι η μόνη προσπελάσιμη ιδιότητα ενός πινάκα. Είναι μια public τιμή, και όχι μέθοδος (μην ξεχνάμε ότι και ένας πίνακας είναι ένα αντικείμενο μιας κλάσης στην ουσία, ότι ισχύει για τις κλάσεις ισχύει και για τους πίνακες.)

Η πρόσβαση στα δεδομένα ενός πινάκα λοιπόν γίνεται όπως και σε άλλες γλώσσες προγραμματισμού, μπορούμε να έχουμε πρόσβαση σε μια τιμή βάζοντας έναν φυσικό αριθμό ανάμεσα στα σύμβολα [χ]. Στο παρακάτω παράδειγμα φτιάχνουμε έναν πινάκα 10 τιμών, και γεμίζουμε όλες τις τιμές του με 0.

```
double [] aDoubleArray = new double [ 10 ];
for ( int i=0; i <aDoubleArray.length; i++ )
  aDoubleArray [ i ] = 0
```

Αυτός ο τρόπος μπορεί να χρησιμοποιηθεί για να έχουμε πρόσβαση σε όλα τα δεδομένα του πινάκα.

```
for( int i : aDoubleArray ) ή for ( int i=0; i <aDoubleArray.length; i++ )
  System.out.println( i );
```

Η προσπάθεια πρόσβασης σε μια θέση του πινάκα που δεν υπάρχει, προκαλεί το πρόγραμμα να "πετάξει" εξαίρεση.

```
String [] states = new String [50];

try {
  states[0] = "California";
  states[1] = "Oregon";
  ...
  states[50] = "McDonald's Land"; // Error: array out of bounds
} //κώδικας για την αποφυγή σφάλματος
catch ( ArrayIndexOutOfBoundsException err ) {
  System.out.println( "Handled error: " + err.getMessage( ) );
}
```

3.2.7 Αναλύοντας

Στην JAVA τα νούμερα είναι πρωτογενείς τύποι και όχι αντικείμενα. Άλλα για κάθε πρωτογενή τύπο, η JAVA ορίζει και μια ανάλογη κλάση. Αυτές οι κλάσεις έχουν μια μέθοδο ανάλυσης (parse) που διαβάζει μια συμβολοσειρά και επιστρέφει τον ανάλογο πρωτογενή τύπο. Για παράδειγμα:

```
byte b = Byte.parseByte("16");
int n = Integer.parseInt( "42" );
long l = Long.parseLong( "99999999999" );
float f = Float.parseFloat( "4.2" );
double d = Double.parseDouble( "99.99999999" );
boolean b = Boolean.parseBoolean("true");
```

Αυτό γίνεται γιατί ένας χαρακτήρας, παράδειγμα το 5, έχει μια αριθμητική τιμή που τον αντιπροσωπεύει ανάλογα με την κωδικοποίηση του, η όποια δεν είναι απαραίτητο ότι είναι ίδια με την τιμή που πιστεύουμε ότι έχει. Αυτές οι εντολές χρησιμοποιούνται πολύ συχνά. Για παράδειγμά, οι αριθμητικές τιμές που βάζει ο χρήστης στο παράθυρο του προγράμματος, καθώς και τα δεδομένα που αποθηκεύονται σε ένα αρχείο, συνήθως κατά την ανάκτηση τους χρειάζονται να αναλυθούν με μία από τις παραπάνω μεθόδους.

3.3 Βασικά του αντικειμενοστραφή προγραμματισμού

Κεντρική ιδέα στον αντικειμενοστραφή προγραμματισμό είναι η κλάση (**class**). Μια κλάση είναι μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων, είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών, εννοιολογικών αντικειμένων, σε ένα περιβάλλον προγραμματισμού. Πρακτικά είναι ένας τύπος δεδομένων, ή αλλιώς το προσχέδιο μίας δομής δεδομένων με δικά της περιεχόμενα, τόσο μεταβλητές όσο και διαδικασίες.

3.3.1 Εμβέλεια (Scope)

Τα περιεχόμενα της κλάσης δηλώνονται είτε ως δημόσια (**public**) είτε ως ιδιωτικά (**private**), με τα ιδιωτικά να είναι προσπελάσιμα μόνο από κώδικα εντός της ίδιας κλάσης. Οι διαδικασίες των κλάσεων συνήθως καλούνται μέθοδοι (**methods**) και οι μεταβλητές τους γνωρίσματα (**attributes**) ή πεδία (**fields**). Μία κλάση πρέπει ιδανικά να είναι εννοιολογικά αυτοτελής, να περιέχει δηλαδή μόνο πεδία τα οποία περιγράφουν μία κατηγορία αντικειμένων και δημόσιες μεθόδους οι οποίες επενεργούν σε αυτά όταν καλούνται από το εξωτερικό πρόγραμμα, χωρίς να εξαρτώνται από άλλα δεδομένα ή κώδικα εκτός της κλάσης, και επαναχρησιμοποιήσιμη, να αποτελεί δηλαδή μαύρο κουτί δυνάμενο να λειτουργήσει χωρίς τροποποιήσεις ως τμήμα διαφορετικών προγραμμάτων.

3.3.2 Αντικείμενο (object)

Αντικείμενο (object) είναι το στιγμιότυπο μίας κλάσης, δηλαδή αυτή καθαυτή η δομή δεδομένων (με αποκλειστικά δεσμευμένο χώρο στη μνήμη) βασισμένη στο καλούπι που προσφέρει η κλάση. Παραδείγματος χάρη, σε μία αντικειμενοστραφή γλώσσα προγραμματισμού θα μπορούσαμε να ορίσουμε κάποια κλάση ονόματι **BankAccount**, η οποία αναπαριστά έναν τραπεζικό λογαριασμό, και να δηλώσουμε ένα αντικείμενο της με όνομα **MyAccoun**t. Το αντικείμενο αυτό θα έχει δεσμεύσει χώρο στη μνήμη με βάση τις μεταβλητές και τις μεθόδους που περιγράψαμε όταν δηλώσαμε την κλάση. Έτσι, στο αντικείμενο θα μπορούσε να περιέχεται ένα γνώρισμα **Balance** (=υπόλοιπο) και μία μέθοδος **GetBalance** (=επέστρεψε το υπόλοιπο). Ακολούθως θα μπορούσαμε να δημιουργήσουμε ακόμα ένα ή περισσότερα αντικείμενα της ίδιας κλάσης (π.χ. στο παράδειγμα οι τραπεζικοί λογαριασμοί των διαφορετικών πελατών) τα οποία θα είναι ίδιες δομές που περιέχουν διαφορετικά στοιχεία.

3.3.3 Ενθυλάκωση δεδομένων (data encapsulation)

Ενθυλάκωση δεδομένων (data encapsulation) καλείται η ιδιότητα που προσφέρουν οι κλάσεις να «κρύβουν» τα ιδιωτικά δεδομένα τους από το υπόλοιπο πρόγραμμα και να εξασφαλίζουν πως μόνο μέσω των δημόσιων μεθόδων τους θα μπορούν αυτά να προσπελαστούν. Αυτή η τακτική παρουσιάζει μόνο οφέλη καθώς εξαναγκάζει κάθε εξωτερικό πρόγραμμα να φιλτράρει το χειρισμό που επιθυμεί να κάνει στα πεδία μίας κλάσης μέσω των ελέγχων που μπορούν να περιέχονται στις δημόσιες μεθόδους της κλάσης.

3.3.4 Αφαίρεση δεδομένων

Αφαίρεση δεδομένων καλείται η ιδιότητα των κλάσεων να αναπαριστούν αφαιρετικά πολύπλοκες οντότητες στο προγραμματιστικό περιβάλλον. Μία κλάση αποτελεί ένα αφαιρετικό μοντέλο κάποιας κατηγορίας αντικειμένων. Επίσης οι κλάσεις προσφέρουν και αφαίρεση ως προς τον υπολογιστή, εφόσον η καθεμία μπορεί να θεωρηθεί ένας μικρός και αυτάρκης υπολογιστής (με δική του κατάσταση, μεθόδους και μεταβλητές).

3.3.5 Κληρονομικότητα (inheritance)

Κληρονομικότητα (inheritance) ονομάζεται η ιδιότητα των κλάσεων να επεκτείνονται σε νέες κλάσεις, ρητά δηλωμένες ως κληρονόμους (υποκλάσεις ή 'θυγατρικές κλάσεις'), οι οποίες μπορούν να επαναχρησιμοποιήσουν τις μεταβιβάσιμες μεθόδους και ιδιότητες της γονικής τους κλάσης αλλά και να προσθέσουν δικές τους. Στιγμιότυπα των θυγατρικών κλάσεων μπορούν να χρησιμοποιηθούν όπου απαιτούνται στιγμιότυπα των γονικών (εφόσον η θυγατρική είναι κατά κάποιον τρόπο μία πιο εξειδικευμένη εκδοχή της γονικής), αλλά το αντίστροφο δεν ισχύει. Παράδειγμα κληρονομικότητας είναι μία γονική κλάση **Vehicle** (=Όχημα) και οι δύο πιο εξειδικευμένες υποκλάσεις της **Car** (=Αυτοκίνητο) και

Bicycle (=Ποδήλατο), οι οποίες λέμε ότι "κληρονομούν" από αυτήν. Πολλαπλή κληρονομικότητα είναι η δυνατότητα που προσφέρουν ορισμένες γλώσσες προγραμματισμού μία κλάση να κληρονομεί ταυτόχρονα από περισσότερες από μία γονικές. Από μία υποκλάση μπορούν να προκύψουν νέες υποκλάσεις που κληρονομούν από αυτήν, με αποτέλεσμα μία ιεραρχία κλάσεων που συνδέονται μεταξύ τους "ανά γενιά" με σχέσεις κληρονομικότητας.

3.3.6 Υπερφόρτωση μεθόδου (method overloading)

Υπερφόρτωση μεθόδου (method overloading) είναι η κατάσταση κατά την οποία υπάρχουν, στην ίδια ή σε διαφορετικές κλάσεις, μέθοδοι με το ίδιο όνομα και πιθανώς διαφορετικά ορίσματα. Αν πρόκειται για μεθόδους της ίδιας κλάσης διαφοροποιούνται μόνο από τις διαφορές τους στα ορίσματα και στον τύπο επιστροφής.

3.3.7 Υποσκέλιση μεθόδου (method overriding)

Υποσκέλιση μεθόδου (method overriding) είναι η κατάσταση κατά την οποία μία θυγατρική κλάση και η γονική της έχουν μία μέθοδο ομώνυμη και με τα ίδια ορίσματα. Χάρη στη δυνατότητα του πολυμορφισμού ο μεταγλωττιστής «ξέρει» πότε να καλέσει ποια μέθοδο, βασισμένος στον τύπο του τρέχοντος αντικειμένου. Δηλαδή πολυμορφισμός είναι η δυνατότητα των αντικειμενοστραφών μεταγλωττιστών να αποφασίζουν δυναμικά ποια είναι η κατάλληλη να κληθεί μέθοδος σε συνθήκες υποσκέλισης.

3.3.8 Αφηρημένη κλάση (abstract class)/ διασύνδεση (interface)

Αφηρημένη κλάση (abstract class) είναι μία κλάση που ορίζεται μόνο για να κληρονομηθεί σε θυγατρικές υποκλάσεις και δεν υπάρχουν δικά της στιγμιότυπα (αντικείμενα). Η αφηρημένη κλάση ορίζει απλώς ένα "συμβόλαιο" το οποίο θα πρέπει να ακολουθούν οι υποκλάσεις της όσον αφορά τις υπογραφές των μεθόδων τους (όπου ως υπογραφή ορίζεται το όνομα, τα ορίσματα και η τιμή επιστροφής μίας διαδικασίας). Μία αφηρημένη κλάση μπορεί να έχει και μη αφηρημένες μεθόδους οι οποίες υλοποιούνται στην ίδια την κλάση (αν και φυσικά μπορούν να υποσκελίζονται σε υποκλάσεις). Αντιθέτως οι αφηρημένες μέθοδοί της είναι απλώς ένας ορισμός της υπογραφής τους και εναπόκειται στις υποκλάσεις να τις υλοποιήσουν. Μία αφηρημένη κλάση που δεν έχει γνωρίσματα και όλες οι μέθοδοί της είναι αφηρημένες και δημόσιες καλείται διασύνδεση (interface). Οι κλάσεις που κληρονομούν από μία διασύνδεση λέγεται ότι την "υλοποιούν".

3.4 Ένα απλό παράδειγμα για την δημιουργία αντικειμένων

Εδώ θα μάθουμε πώς να:

- Σχεδιάζουμε ένα αντικείμενο
- Να αποθηκεύουμε δεδομένα σε ένα αντικείμενο
- Να επεξεργαζόμαστε δεδομένα σε ένα αντικείμενο
- Να δημιουργούμε ένα νέο στιγμιότυπο ενός αντικειμένου

3.4.1 Η δήλωση της κλάσης

Τα δεδομένα που ένα αντικείμενο περιλαμβάνει και πως τα διαχειρίζεται, ορίζονται στην διαδικασία τις δημιουργίας τις κλάσης. Για παράδειγμα παρακάτω είναι ο βασικός ορισμός μιας κλάσης με το αντικείμενο **book**.

```
public class Book
```

Αξίζει να πάρουμε μια στιγμή να αναλύσουμε την παραπάνω δήλωση. Η πρώτη γραμμή περιλαμβάνει 2 λέξεις κλειδιά τις **java**, το **public** και το **class**

Το **public** ελέγχει το ποια κομμάτια του προγράμματος μπορούν να έχουν πρόσβαση σε αυτήν την κλάση. Για την ακρίβεια, για κλάσεις πρώτου επιπέδου (δηλαδή κλάσεις που δεν περιλαμβάνονται μέσα σε άλλες κλάσεις), όπως η κλάση μας, πρέπει να είναι οπωσδήποτε **public**. Η λέξη κλειδί **class** χρησιμοποιείται για να δηλώσει ότι περιλαμβάνεται μέσα από τις αγκύλες είναι μέρος τις κλάσης μας. Πάντα μετά από αυτά τοποθετείται το όνομα τις κλάσης .

3.4.2 Πεδία (fields)

Τα πεδία χρησιμοποιούνται για να αποθηκεύσουν τα δεδομένα του αντικειμένου και το σύνολο τους ορίζει το σύνολο του αντικειμένου μας. Καθώς φτιάχνουμε ένα Αντικείμενο «βιβλίο» θα είχε νόημα στην συγκεκριμένη περίπτωση να κρατάνε δεδομένα για τον τίτλο του βιβλίου, τον συγγραφέα και τον έκδοτη.

```
public class Book {

  //fields
  private String title;
  private String author;
  private String publisher;
 }
```

Τα πεδία δεν είναι τίποτα άλλο από κανονικές μεταβλητές, με μια μόνο σημαντική διαφορά. Για να τηρηθεί ο κανόνας της ενθυλάκωσης επιβάλλεται η λέξη κλειδί "**private**". Αυτό σημαίνει ότι αυτές οι μεταβλητές μπορούν να προσπελαστούν μόνο με μεθόδους μέσα από την κλάση που τους περιλαμβάνει. Είναι αξιοσημείωτο ότι ο περιορισμός δεν

είναι υποχρεωτικός στην java, δηλαδή ο compiler δεν θα σταματήσει τον πρόγραμμα αν κάνουμε τις μεταβλητές public.

3.4.3 Ο δομητής (constructor)

Οι περισσότερες κλάσεις έχουν μια μέθοδο δομητή. Είναι η μέθοδος που καλείται όταν το αντικείμενο πρωτοκατασκευάζεται, και μπορεί να χρησιμοποιηθεί για να θέσει την αρχική του κατάσταση

```
public class Book {

  //fields
  private String title;
  private String author;
  private String publisher;

  //constructor method
  public Book(String bookTitle, String authorName, String publisherName);
  {
   //populate the fields
   title = bookTitle;
   author = authorName;
   publisher = publisherName;
 }
```

Ο δομητής χρησιμοποιεί το ίδιο όνομα όπως και η κλάση και πρέπει να είναι public. Παίρνει τα δεδομένα που περνάνε μέσα στην μέθοδο και θέτει τις τιμές στα πεδία τις κλάσης θέτοντας το αντικείμενο στην αρχική κατάσταση που το θέλουμε.

Για παράδειγμα έστω ότι θέλουμε να καλέσουμε αυτήν την κλάση από μια άλλη κλάση, θα την δηλώναμε ως έξης:

```
Private Book ονομαΜεταβλητης=νέο Book("title","author","publisherName");
```

Πολλές φορές για να μην σκεφτόμαστε κάθε φορά και αλλά ονόματα για τις μεταβλητές μας μπορούμε να κάνουμε το εξής:

```
public class Book {

  //fields
  private String title;
  private String author;
  private String publisher;

  //constructor method
  public Book(String title, String author, String publisher)
  {
   //populate the fields
   this.title = title;
   this.author = author;
   this.publisher = name;
  }
 }
```

Στην συγκεκριμένη περίπτωση το όνομα τις μεταβλητής που περνάει είναι το ίδιο με το όνομα του πεδίου τις κλάσης. Για να αντιμετωπίσουμε αυτό το ενδεχόμενο χρησιμοποιούμε

την λέξη κλειδί τις java **«this»** πριν την μεταβλητή μας για να δηλώσουμε ότι θέλουμε να απευθυνθούμε στην μεταβλητή τις κλάσης και όχι σε αυτήν που περνά . Το "**this**" μπορεί να χρησιμοποιηθεί σε όλες τις μεταβλητές τις κλάσης σε κάθε περίπτωση, και όχι μόνο σε αυτή. Απλά για λόγους συντομίας μπορούμε να μην το βάλουμε όταν δεν χρειάζεται.

3.4.4 Προσθέτοντας μεθόδους

Οι μέθοδοι είναι πράξεις που ένα αντικείμενο μπορεί να κάνει και είναι γραμμένες σαν μέθοδοι. Αυτή την στιγμή στο παράδειγμα μας έχουμε μια κλάση η οποία μπορεί να απόκτηση υπόσταση αλλά δεν κάνει τίποτα άλλο. Ας προσθέσουμε μια μέθοδο που ονομάζεται **display book data** που θα προβάλει τα δεδομένα που βρίσκονται τώρα στο αντικείμενο

```
public class Book {

  //fields
  private String title;
  private String author;
  private String publisher;

  //constructor method
  public Book(String bookTitle, String authorName, String publisherName )
  {
   //populate the fields
   title = bookTitle;
   author = authorName ;
   publisher = publisherName ;
  }

  public void displayBookData()
  {
 System.out.println("Title:" + title);
System.out.println("Author: " + author);
 System.out.println("Publisher: " + publisher);
    }
 }
```

Το μόνο που κάνει αυτή η μέθοδος είναι να προβάλει τα περιεχόμενα κάθε μεταβλητής της κλάσης στην οθόνη.

Μπορούμε να προσθέσουμε όσες μεθόδους και μεταβλητής χρειαζόμαστε άλλα για τώρα ας θεωρήσουμε ότι η κλάση book είναι έτοιμη. Έχει 3 πεδία για να κρατάει δεδομένα για ένα βιβλίο, μπορεί να αποκτήσει υπόσταση και μπορεί να προβάλει τα δεδομένα που περιλαμβάνει.

3.4.5 Δημιουργώντας ένα στιγμιότυπο αντικειμένου

Για να δημιουργήσουμε ένα στιγμιότυπο από την κλάση **Book** συνήθως χρειαζόμαστε μια άλλη κλάση μέσα στην οποίο το στιγμιότυπο θα δηλωθεί ως μεταβλητή. Θα δημιουργήσουμε μια νέα κλάση στον ίδιο φάκελο με το όνομα **BookTracker.java**

```
public class BookTracker {

  public static void main(String[] args) {

   Book firstBook = νέο Book("Hyperion","Dan Simmons","Anubis");
  }}
```

Στην αριστερή πλευρά από το ίσον έχουμε την δήλωση της μεταβλητής που θα κρατήσει το αντικειμένο. Ουσιαστικά δεσμεύει χώρο στην μνήμη του υπολογιστή που μπορεί να αποθηκεύσει ένα στιγμιότυπο αντικειμένου book και σε αυτόν τον χώρο μνήμης αντιστοιχείται το όνομα **firstbook**. Στην δεξιά πλευρά από το ίσον έχουμε την δημιουργία του νέου στιγμιότυπου. Αυτό που κάνει είναι ότι πάει στον δομητή του book class και τρέχει τον κώδικα μέσα του. Τώρα ας δούμε τα δεδομένα για να βεβαιωθούμε ότι όντως δημιουργήσαμε ένα νέο στιγμιότυπο. το μόνο που έχουμε να κάνουμε είναι να καλέσουμε την μέθοδο του αντικειμένου **displayBookData**

```
public class BookTracker {

  public static void main(String[] args) {

   Book firstBook = νέο Book("Hyperion","Dan Simmons","Anubis");
   firstBook.displayBookData();
  }
 }
```

Το αποτέλεσμα είναι:

```
Title: Hyperion
Author: Dan Simmons
Publisher: Anubis
```

3.4.6 Πολλαπλά αντικείμενα

Τώρα μπορούμε να δούμε την δύναμη του αντικειμενοστραφή προγραμματισμού

Μπορούμε να τροποποιήσουμε το πρόγραμμα:

```
public class BookTracker {

  public static void main(String[] args) {

   Book firstBook = new Book("Horton Hears A Who!","Dr. Seuss","Random
House");
   Book secondBook = new Book("The Cat In The Hat","Dr. Seuss","Random
House");
   Book anotherBook = new Book("The Maltese Falcon","Dashiell
Hammett","Orion");
   firstBook.displayBookData();
   anotherBook.displayBookData();
   secondBook.displayBookData();
  }
 }
```

Γράφοντας μια κλάση πλέον έχουμε την ικανότητα να δημιουργήσουμε όσα αντικείμενα θέλουμε. Επιπλέον μελλοντικές διορθώσεις που ο προγραμματιστής μπορεί να θέλει να υλοποιήσει στην κλάση και που αφορούν την δομή δεδομένων ή την συμπεριφορά τους, εφαρμόζονται αυτόματα σε όλα τα αντικείμενα.

4 Object oriented design

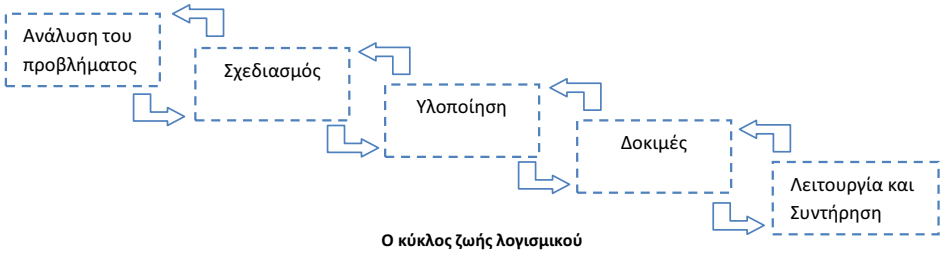
4.1 ο κύκλος ζωής λογισμικού

Σε αυτό το κεφάλαιο θα εξηγήσουμε τον κύκλο ζωής λογισμικού: τις ενέργειες που πραγματοποιούνται από την στιγμή που μια εφαρμογή ξεκίνα σαν ιδέα μέχρι την στιγμή που σταματήσει η χρήση της.

Μια εφαρμογή ξεκίνα συνήθως επειδή κάποιος πελάτης έχει ένα πρόβλημα και είναι πρόθυμος να πληρώσει για την λύση του. Έχει δημιουργηθεί μια επίσημη διαδικασία για την μετατροπή τέτοιων προβλημάτων σε κώδικα. Μια τέτοια διαδικασία αναγνωρίζει και περιγράφει διαφορετικές φάσεις της ανάπτυξης λογισμικού, και δίνει οδηγίες για την αντιμετώπιση τους.

Χρησιμοποιώντας αυτές τις οδηγίες πολλοί προγραμματιστές σπάνε την διαδικασία την ανάπτυξης μιας εφαρμογής στις ακόλουθες 5 φάσεις:

- Ανάλυση του προβλήματος (Problem specifications, analysis)
- Σχεδιασμός (Design)
- Υλοποίηση (implementation)
- Δοκιμές / Ελεγχος (Testing)
- Λειτουργία και Συντήρηση (Deployment)



Ο κύκλος ζωής λογισμικού

Στην φάση της ανάλυσης, αποφασίζουμε το τι πρέπει να κάνει το πρόγραμμα. Αυτό δεν περιλαμβάνει το πώς το πρόγραμμα θα κάνει αυτά που υποτίθεται ότι πρέπει να κάνει. Το αποτέλεσμα της ανάλυσης είναι ένα κείμενο, το οποίο περιγράφει με κάθε λεπτομέρεια τι θα κάνει το πρόγραμμα μόλις ολοκληρωθεί. Κομμάτι αυτού του κείμενου μπορεί να είναι ένα εγχειρίδιο οδηγιών, που αναφέρει στον χρηστή πως να χρησιμοποιήσει το πρόγραμμα για να πάρει τα αποτελέσματα που θέλει. Άλλο κομμάτι τις ανάλυσης είναι οι επιδόσεις που πρέπει να έχει το πρόγραμμα: πόσες εισόδους πρέπει το πρόγραμμα να μπορεί να δεχτεί, σε τι χρόνο, ή ποια είναι η μεγίστη κατανάλωση μνήμης και χώρου που έχει.

Στην φάση του σχεδιασμού, κατασκευάζουμε ένα σχέδιο για το πώς θα προγραμματίσουμε το σύστημα. Ανακαλύπτουμε τις δομές και τα κομμάτια που απαρτίζουν το πρόβλημα μας. Σε περίπτωση αντικειμενοστραφή προγραμματισμού αποφασίζουμε τι κλάσεις θα χρειαστούμε και ποιες θα είναι οι πιο σημαντικές μέθοδοι τους. Το αποτέλεσμα αυτής της

φάσης είναι μια περιγραφή των κλάσεων και των μεθόδων με διαγράμματα που δείχνουν την σχέση και τις εξαρτήσεις που υπάρχουν μεταξύ των κλάσεων.

Στην φάση της υλοποίησης, όπως λέει και το όνομα της φάσης, γραφούμε τον κώδικα για την εκτέλεση των κλάσεων και των μεθόδων που ανακαλύψαμε στην φάση του σχεδιασμού. Το αποτέλεσμα αυτής της φάσης είναι το ολοκληρωμένο πρόγραμμα.

Στην φάση των δοκίμων, δοκιμάζουμε το πρόγραμμα για κάθε δυνατή είσοδο που μπορούμε να σκεφτούμε, για να επιβεβαιώσουμε ότι το πρόγραμμα δουλεύει σωστά. Η έξοδος αυτής της φάσης είναι μια αναφορά που περιγράφει τα τεστ που πραγματοποιήσαμε και τα αποτελέσματα τους. (Προφανώς αν δεν γραφούμε εμπορικές εφαρμογές η εφαρμογές που τον κώδικα τους δεν πρόκειται να δει άλλος, δεν είναι αναγκαίο να κρατάμε κάθε λεπτομέρεια των δοκίμων μας).

Στην φάση της ανάπτυξης, οι τυπικοί χρηστές του προγράμματος το εγκαθιστούν και το χρησιμοποιούν για τον σκοπό που πραγματοποιήθηκε.

4.2 Σχεδιάζοντας μεθόδους και κλάσεις

Στην φάση του σχεδιασμού είπαμε, ο σκοπός μας είναι να ανακαλύψουμε τις απαραίτητες δομές που μας δίνουν την δυνατότητα να προγραμματίσουμε ένα σετ εντολών στον υπολογιστή. Όταν εξετάζουμε ένα πρόγραμμα σε αντικειμενοστραφή προγραμματισμό κάνουμε τις ακόλουθες διαδικασίες

- Ανακαλύπτουμε τις κλάσεις
- Προσδιορίζουμε τις ευθύνες της κάθε κλάσης
- Περιγράφουμε τις σχέσεις μεταξύ των κλάσεων

Μια κλάση αντιπροσωπεύει μια ιδέα. Μπορούμε να δούμε κλάσεις για συμπαγείς οντότητες όπως λογαριασμούς τραπέζης, τετράγωνα, προϊόντα. Άλλες κλάσεις αντιπροσωπεύουν πιο αφηρημένες άνοιες, όπως ροές.

Ένας απλός κανόνας για να ανακαλύψουμε κλάσεις είναι να αναζητήσουμε τα ουσιαστικά στην περιγραφή του προβλήματος. Για παράδειγμα ας πούμε ότι το πρέπει να φτιάξουμε ένα πρόγραμμα που τυπώνει ένα τιμολόγιο, για έναν πελάτη που έχει αγοράσει κάποια προϊόντα. Οι προφανείς κλάσεις που έρχονται στο μυαλό είναι «Τιμολόγιο», «Προϊόν», «Πελάτης» .

Είναι καλή ιδέα γενικά να κρατάμε λίστα με υποψήφιες κλάσεις σε μια κολλά χαρτί. Καθώς σκεφτόμαστε το πρόγραμμα απλά πρέπει να βάζουμε όλες τις ιδέες που μας έρχονται για κλάσεις σε αυτήν την λίστα. Μπορούμε κάλλιστα μετά να μην τις χρησιμοποιήσουμε όλες τις ιδέες μας, ή και να χρειαστεί να προσθέσουμε άλλες, αλλά είναι πάντα χρήσιμο να έχουμε κάποιες ιδέες μπροστά μας.

Καθώς βρίσκουμε κλάσεις πρέπει να θυμόμαστε το εξής: Οι κλάσεις αντιπροσωπεύουν μια σειρά αντικειμένων με κοινά χαρακτηριστικά. Οντότητες με πολλαπλές εμφανίσεις όπως

«Πελάτης» και «Προϊόντα» είναι καλοί υποψήφιοι για αντικείμενα. Βρίσκουμε τι έχουν κοινό και σχεδιάζουμε κλάσεις για να εκμεταλλευτούμε αυτά τα κοινά στοιχεία τους.

Κάποιες οντότητες πρέπει να αντιπροσωπευτούν σαν αντικείμενα, άλλες σαν βασικές μεταβλητές. Για παράδειγμα πρέπει μια διεύθυνση ενός πελάτη να είναι αντικείμενο μιας κλάσης «Διεύθυνση», ή απλά μια συμβολοσειρά; Δεν υπάρχει τέλεια απάντηση. Εξαρτάται από το πρόβλημα που θέλουμε να λύσουμε. Αν το πρόγραμμα μας πρέπει να αναλύσει διευθύνσεις (Για παράδειγμα για να υπολογίσει το κόστος αποστολής), τότε ναι, μια κλάση «Διεύθυνση» είναι ο σωστός σχεδιασμός. Εναλλακτικά αν το πρόγραμμα μας ποτέ δεν θα χρειαστεί μια τέτοια λειτουργιά, δεν χρειάζεται να σπαταλήσουμε χρόνο σχεδιάζοντας κάτι τόσο πολύπλοκο. Είναι η δουλειά μας να προσδιορίσουμε τον πιο αποδοτικό σχεδιασμό, έναν ο όποιος δεν μας περιορίζει πολύ, ούτε είναι υπερβολικά γενικός.

Δεν μπορούν όλες οι κλάσεις που θα χρειαστούμε φυσικά να ανακαλύπτουν στην φάση της ανάλυσης. Τα περισσότερα πολύπλοκα προγράμματα χρειάζονται κλάσεις για πρακτικές εφαρμογές, όπως για πρόσβαση σε αρχεία ή βάσεις δεδομένων, γραφικά περιβάλλοντα, μηχανισμούς ελέγχου, κτλ.

Κάποιες από τις κλάσεις που χρειαζόμαστε μπορεί να υπάρχουν είδη, είτε στην βιβλιοθήκη της γλώσσας προγραμματισμού που χρησιμοποιούμε, είτε σε ένα πρόγραμμα που έχουμε κατασκευάσει παλιότερα. Μπορούμε ακόμα να χρησιμοποιήσουμε τον μηχανισμό της κληρονομικότητας για να επεκτείνουμε κλάσεις που είδη έχουν κατασκευαστεί σε κλάσεις που ανταποκρίνονται στις ανάγκες μας.

Μόλις ένα σετ από κλάσεις αναγνωριστεί, πρέπει να προσδιορίσουμε την συμπεριφορά για κάθε κλάση. Πρέπει να βρούμε ποιες μεθόδους κάθε αντικείμενο να έχει για να λύσουμε ένα πρόβλημα. Ένας εύκολος τρόπος για να ανακαλύψουμε αυτές τις μεθόδους είναι να αναζητήσουμε για ρήματα στην περιγραφή του προβλήματος, και μετά να ταιριάξουμε τα ρήματα στο σωστό αντικείμενο. Για παράδειγμα στο πρόβλημα του τιμολόγιου που θέσαμε νωρίτερα, μια κλάση πρέπει να υπολογίσει την συνολική αξία των προϊόντων. Τώρα πρέπει να ξεκαθαρίσουμε μια κλάση θα είναι υπεύθυνη για αυτήν την μέθοδο. Χρειάζεται οι πελάτες και άρα η κλάση πελάτης να υπολογίσουν τι πρέπει να πληρώσουν; Όχι. Τα προϊόντα μήπως; Πάλι όχι. Η καλύτερη επιλογή είναι να κάνουμε την μέθοδο «υπολογισμός κόστους», ευθύνη της κλάσης τιμολόγιο.

4.3 CRC cards, uml diagrams

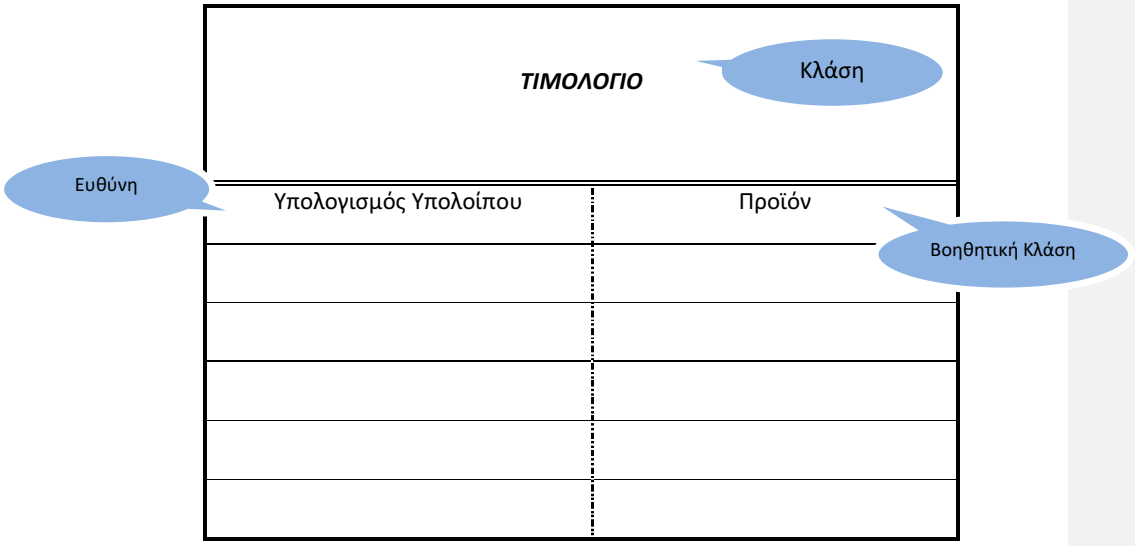
4.3.1 CRC cards

Ένας εξαιρετικός τρόπος για να κάνουμε το παραπάνω είναι η μέθοδος με καρτέλες CRC (Classes-κλάσεις, Responsibilities- ευθύνες, Collaborators-βοηθοί), και στην πιο απλή μορφή της η μέθοδος δουλεύει ως εξής. Χρησιμοποιούμε μια καρτέλα (ένα φύλλο χαρτί) για κάθε κλάση. Καθώς σκεφτόμαστε και αναζητάμε τα ρήματα στην περιγραφή του κάθε προβλήματος, διαλέγουμε την καρτέλα της κλάσης που πιστεύουμε ότι πρέπει να είναι υπεύθυνη για αυτό το πρόβλημα, και γραφούμε αυτήν την ευθύνη στη καρτέλα. Για κάθε ευθύνη γραφούμε διπλά το ποιες άλλες κλάσεις χρειάζονται για να περατωθεί αυτή η

ευθύνη. Μια καρτέλα CRC λοιπόν περιγράφει μια κλάση, της ικανότητες τις, και τις βοηθητικές κλάσεις που χρειάζονται.

Για παράδειγμα, ας πούμε ότι μελετάμε το πρόβλημα του τιμολόγιου που αναφέραμε παραπάνω. Γραφούμε στην καρτέλα της κλάσης του τιμολόγιου, την φράση «υπολογισμός υπόλοιπου». Αν αυτή η κλάση με τις μεθόδους της μπορεί να πραγματοποιήσει την διαδικασία μόνη της δεν κάνουμε τίποτα άλλο. Αν χρειάζεται και την βοήθεια και από άλλες κλάσεις για να πραγματοποιήσει αυτήν την δουλειά τότε τις γραφούμε στην δεξιά πλευρά, διπλά από τον τίτλο της μεθόδου.

Στην περίπτωση μας για να υπολογίσει η κλάση **«τιμολόγιο»** την τελική τιμή πρέπει να ζητήσει από την κλάση **«προϊόν»** το κόστος κάθε προϊόντος. Αν η κλάση **«προϊόν»** δεν έχει μέθοδο **«getΣυνολικήτιμη»** είναι μια καλή ώρα να την προσθέταμε.



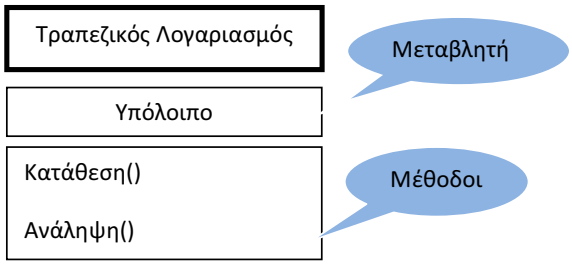
Οι ευθύνες όμως που γραφούμε στις καρτέλες είναι υψηλού επιπέδου και έτσι μερικές φόρες μια ευθύνη μπορεί να χρειαστεί παραπάνω από 2 μεθόδους η κλάσεις για να περατωθεί.

Αυτή η μέθοδος που περιγράψαμε είναι πρόχειρη μέθοδος, την οποία μπορούμε να την χρησιμοποιήσουμε για να είμαστε δημιουργικοί και να ανακαλύψουμε τις κλάσεις που χρειαζόμαστε και τις ιδιότητες τους. Μόλις αποφασίσουμε ότι έχουμε καταλήξει σε ένα καλό σύνολο από κλάσεις, θα θέλουμε να ξέρουμε πως αλληλεπιδρούν μεταξύ τους. Μπορούμε να βρούμε κλάσεις με κοινές ιδιότητες, ώστε μερικές ευθύνες να αντιμετωπιστούν από μια κοινή υποκλάση; Μπορούμε να ομαδοποιήσουμε τις κλάσεις μας σε ομάδες ανεξάρτητες μεταξύ τους; Το πώς ανακαλύπτουμε τις σχέσεις μεταξύ των κλάσεων το βλέπουμε στο άλλο κεφάλαιο.

4.3.2 Ιδιότητες και μέθοδοι σε UML διαγράμματα

Μερικές φόρες είναι χρήσιμο να δείχνουμε τις ιδιότητες (attributes) και τις μεθόδους μιας κλάσης στο διάγραμμα της. Οι ιδιότητες των αντικειμένων μια κλάσης είναι μεταβλητές που τα ονόματα του προκύπτουν από τα διάφορα χαρακτηριστικά του αντικειμένου αυτού. Για παράδειγμα, **«όνομα»** και **«τιμή»** είναι ιδιότητες που μια κλάση **«προϊόν»** να πρέπει να έχει. Συνήθως οι ιδιότητες αντιστοιχούν σε μεταβλητές του στιγμιότυπου (class instance), αλλά αυτό δεν είναι απαραίτητο, μια κλάση μπορεί να έχει διαφορετικό τρόπο οργάνωσης των δεδομένων τους εσωτερικά. Για παράδειγμα η κλάση **«Gregorian Calendar»** έχει ιδιότητες, **«μέρα»**, **«μηνάς»**, **«χρόνος»** και θα ήταν λογικό να σχεδιάσουμε ένα uml διάγραμμα με αυτές τις ιδιότητες. Παρόλα αυτά η κλάση δεν έχει στην πραγματικότητα μεταβλητές στιγμιότυπου που αποθηκεύουν αυτές τις τιμές. Αντιθέτως εσωτερικά μετράει όλες τις ημερομηνίες μετρώντας τα μικροδευτερόλεπτα από την 1/1/1970, ένα προγραμματιστικό τερτίπι που ο χρηστής της κλάσης δεν είναι ανάγκη να ξέρει.

Μπορούμε να δείξουμε τις ιδιότητες και τις μεθόδους σε ένα διάγραμμα κλάσης χωρίζοντας το παραλληλόγραμμο τις κλάσεις (στο uml διάγραμμα), σε 3 κομμάτια, στα οποία το όνομα της κλάσεις μπαίνει στο πρώτο, οι ιδιότητες της στην μέση, και οι μέθοδοι της στην τελευταία. Δεν είναι ανάγκη να παρουσιαστούν τα πάντα στο διάγραμμα, μόνο αυτές που δίνουν την ουσία της κλάσης.



4.4 Σχέσεις μεταξύ κλάσεων

Όταν σχεδιάζουμε ένα πρόγραμμα, είναι χρήσιμο να περιγράφουμε τις σχέσεις που υπάρχουν μεταξύ των κλάσεων μας. Αυτό βοηθάει με διαφόρους τρόπους. Για παράδειγμα αν βρούμε κλάσεις με κοινές ιδιότητες, μπορούμε να γλυτώσουμε λίγο κόπο τοποθετώντας την κοινή ιδιότητα που έχουν σε μια ανώτερη υποκλάση. Αν ξέρουμε ότι μερικές από τις κλάσεις δεν έχουν καμιά σχέση μεταξύ τους, και το πρόγραμμα φτιάχνεται από μια ομάδα, μπορούμε να δώσουμε την κάθε ομάδα κλάσεων σε διαφορετικό προγραμματιστή και να μην ανησυχούμε ότι ο κάθε ένας πρέπει να περιμένει τον άλλο να τελειώσει.

Ας πούμε για την σχέση της κληρονομικότητας. Η κληρονομικότητα είναι πολύ σημαντική σχέση μεταξύ κλάσεων, αλλά δεν είναι η μόνη χρήσιμη σχέση. Η κληρονομικότητα είναι η σχέση μεταξύ μιας πιο γενικής κλάσης (την υπερκλάση) και μια πιο εξειδικευμένης κλάσης (την υποκλάση). Αυτή η σχέση, πολλές φόρες περιγράφεται σαν την **«είναι»** σχέση. Κάθε

αυτοκίνητο **«είναι»** ένα όχημα. Κάθε καταθετικός λογαριασμός **«είναι»** ένας λογαριασμός τραπέζης, κάθε κύκλος **«είναι»** μια έλλειψη (με ίδιο ύψος πλάτος).

Η κληρονομικότητα (η σχέση **«είναι»**) μερικές φόρες χρησιμοποιείται λανθασμένα ενώ η σχέση **«έχει»** θα ήταν πιο σωστή. Για παράδειγμα, ας πούμε ότι έχουμε την κλάση **«λάστιχο»** που αντιπροσωπεύει ένα λάστιχο αυτοκίνητου. Μήπως πρέπει η κλάση **«λάστιχο»** να είναι υποκλάση της κλάσης **«κύκλος»**; Δεν ακούγεται και τόσο σωστό, όμως στην κλάση **«κύκλος»** υπάρχουν διαφορές χρήσιμες μεθόδους, η κλάση **«λάστιχο»** μπορεί να κληρονομήσει από αυτήν μεθόδους που υπολογίζουν την ακτίνα, περίμετρο και το κέντρο, πράγματα που μπορεί να είναι χρήσιμα. Ενώ αυτό μπορεί να είναι βολικό για τον προγραμματιστή, η κληρονομικότητα στην συγκεκριμένη περίπτωση δεν έχει λογική. Δεν είναι αλήθεια ότι κάθε λάστιχο είναι και κύκλος. Το λάστιχο είναι κομμάτι ενός αυτοκίνητου, ενώ ο κύκλος είναι γεωμετρικό σχήμα. Όντως υπάρχει μια σχέση μεταξύ λάστιχων και κύκλων όμως. Ένα λάστιχο έχει έναν κύκλο και το όριο του.

4.5 Ανακεφαλαίωση

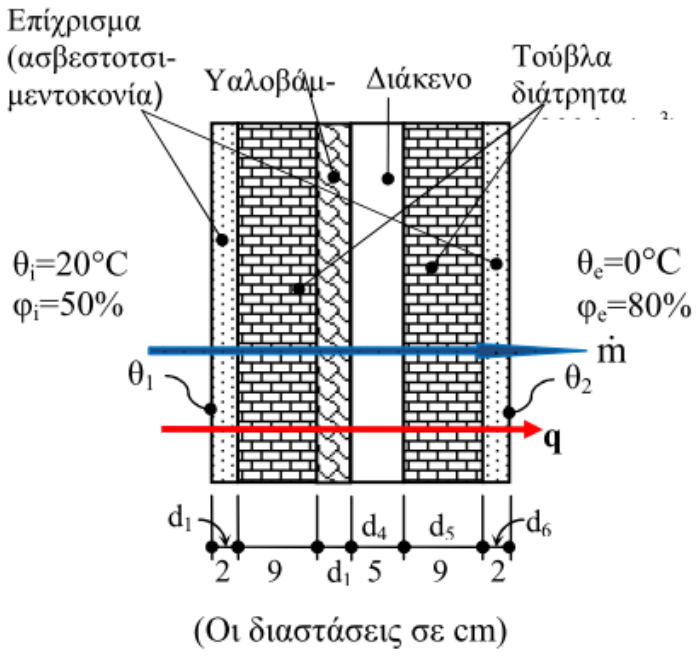
Πριν αρχίσουμε την συγγραφή του κώδικα για ένα πολύπλοκο πρόβλημα πρέπει να βρούμε πρώτα την λύση του. Η μεθοδολογία που περιγράψαμε σε αυτό το κεφάλαιο προτείνει να ακολουθήσουμε μια διαδικασία σχεδιασμού που αποτελείται από τα παρακάτω βήματα:

- Ανακαλύπτουμε τις κλάσεις
- Φτιάχνουμε μια λίστα με τα ουσιαστικά, στην περιγραφή του προβλήματος.
- Αφαιρούμε αυτά τα οποία δεν είναι λογικοί υποψήφιοι για κλάσεις.
- Ξεκαθαρίζουμε τις αρμοδιότητες της κάθε κλάσης
- Φτιάχνουμε μια λίστα με τα καθήκοντα που το πρόγραμμα μας πρέπει να πραγματοποιεί. Διαλέγουμε ένα.
- Σκεφτόμαστε ποια κλάση θα είναι υπεύθυνη για να πραγματοποιήσει αυτό το καθήκον.
- Φτιάχνουμε μια καρτέλα για αυτήν την κλάση και γραφούμε το καθήκον πάνω της.
- Σκεφτόμαστε αν ένα αντικείμενο αυτής της κλάσης μπορεί να ολοκληρώσει το καθήκον μόνο του η αν χρειάζεται την βοήθεια άλλων αντικειμένων από άλλες κλάσεις.
- Αν ναι το σημειώνουμε και τα βοηθητικά αντικείμενα πάνω στην καρτέλα. Δεν πρέπει να φοβόμαστε να μετακινήσουμε, να χωρίσουμε η να αναθεωρήσουμε τα καθήκοντα μεταξύ των κλάσεων.
- Έχουμε τελειώσει όταν έχουμε ολοκληρώσει αυτήν την διαδικασία για όλα τα καθήκοντα και έχουμε πειστεί ότι μπορούν να λάθουνε όλα με τις κλάσεις και τις μεθόδους που έχουμε καταλήξει.
- Περιγράφουμε της σχέσεις μεταξύ των κλάσεων.
- Κατασκευάζουμε ένα διάγραμμα κλάσεων που περιγράφει τις σχέσεις μεταξύ όλων τον κλάσεων που ανακαλύψαμε.

5 Μετατρέποντας το πρόβλημα της υγρομόνωσης σε μορφή προγραμματισμού

5.1 Μια τυπική άσκηση υγρομόνωσης

Πριν ξεκινήσει ο προγραμματισμός του προγράμματος θα πρέπει να γνωρίζουμε το πρόβλημα που πρέπει να επιλύει. Παρακάτω λοιπόν είναι ένα τυπικό πρόβλημα υγρομόνωσης από τις σημειώσεις του κ. Μ. Κτενιαδάκη, Σεμινάριο ΕΝΕΡ4-ΙΚΕΜΤΕΕ 2009 κεφ 1 υγρομόνωση, σελ 1. Για τον εξωτερικό τοίχο κατοικίας που δείχνεται σε τομή στο παρακάτω σχήμα: Να γίνει το διάγραμμα Glaser για να διαπιστωθεί ο κίνδυνος δημιουργίας συμπυκνωμάτων μέσα στο τοίχωμα και να υπολογισθεί η ποσότητα των υδρατμών που συμπυκνώνονται κατά την περίοδο δρόσου (2160h)



Τα δεδομένα είναι:

-
-
-
-
-](Για επίχρισμα ασβεστοτσιμεντοκονιάματος)
- (Για τούβλα διάτρητα 1200)
- (Για υαλοβάμβακα)
- (Για επίχρισμα ασβεστοτσιμεντοκονιάματος)

- $h_i = 7{,}7\ [\frac{W}{m^2K}]$ (Συν. μεταβίβασης θερμότητας,για εσωτερική πλευρά τοιχων)
- $h_e = 25[\frac{W}{m^2K}]$ (Συν. μεταβίβασης θερμότητας, για εξωτερική πλευρά τοίχων)
- $\theta_i = 20C\ με\ \varphi_i = 50\%$
- $\theta_e = 20C\ με\ \varphi_e = 80\%$

Ακόμα, βρίσκουμε τους συντελεστές αντίστασης διαπίδυσης υδρατμών των στρώσεων:

$$\mu_1 = \mu_6 = 15\ (Για\ επίχρισμα\ ασβεστοτσιμεντοκονιάματος)$$

$$\mu_2 = \mu_5 = 10\ (Για\ τούβλα\ διάτρητα)$$

$\mu_3 = 2\ (Για\ πλάκες\ υαλοβάμβακα)$

$$\mu_4 = 1\ (Για\ αερα)$$

Επίσης, χρειάζονται οι πιέσεις κορεσμού (τάση κορεσμένων υδρατμών):

$$P_{k,i} = 2340\ [Pa\ ](Για\ 20C)$$

$$P_{k,e} = 611\ [Pa](Για\ 0C)$$

Εκτελούμε τους σχετικούς υπολογισμούς, συμπληρώνοντας τον παρακάτω Πίνακα:

Βήμα 1ο : Υπολογισμός συντελεστή θερμοπερατότητας του τοιχώματος, U

$\frac{1}{U} = \frac{1}{h_i} + 2 * \frac{d_1}{\lambda_1} + 2 * \frac{d_2}{\lambda_2} + \frac{d_3}{\lambda_3} + \frac{d_4}{\lambda_4} + \frac{1}{h_\varepsilon}$

Και με αντικατάσταση των δεδομένων:

$$U_a = \frac{1}{\frac{1}{7{,}7} + 2 * \frac{0{,}02}{0{,}87} + 2 * \frac{0{,}09}{0{,}52} + \frac{0{,}03}{0{,}41} + 0{,}18 + \frac{1}{25}} = 0.678\ [\frac{W}{m^2K}]$$

Βήμα 2ο : Υπολογισμός των θερμοκρασιών στις επιφάνειες επαφής

Η συνολική θερμοκρασιακή διαφορά (πτώση) είναι: $\Delta\theta = \theta_i - \theta_1 = 20°C$. Οπότε, για τις επιμέρους θερμοκρασιακές πτώσεις, έχουμε διαδοχικά:

θ_1, πάνω στην εσωτερικη επιφάνεια: $\frac{q}{A} = U * (\theta_i - \theta_\varepsilon) = h_i * (\theta_i - \theta_1)$ $\Delta\theta_i = \frac{U}{h_1} * \Delta_\theta \Rightarrow \Delta\theta_i = \frac{0.678}{7.7} * 20 = 1.76\ και\ \theta_1 = 20 - 1{,}76 = 18{,}24°C$

$\theta_{1,2}$ μεταξύ επιχρίσματος – τούβλου: $\Delta\theta_1 = \frac{d_1}{\lambda_1} * U * \Delta\theta = \frac{0.02}{0.87} * 0.678 * 20 = 0.31\ και\ \theta_{1,2} = 18.24 - 031 = 17.93°C$

$\theta_{2,3}$ μεταξύ τούβλου – υαλοβάμβακα: $\Delta\theta_2 = \frac{d_2}{\lambda_2} * U * \Delta\theta = \frac{0.09}{0.52} * 0.678 * 20 = 2.35\ και\ \theta_{2,3} = 17.93 - 2.35 = 15.58°C$

$\theta_{3,4}$ μεταξύ υαλοβάμβακα –διακένου: $\Delta\theta_3 = \frac{d_3}{\lambda_3} * U * \Delta\theta = \frac{0.03}{0.041} * 0.678 * 20 = 9.92$ $\kappa\alpha\iota$ $\theta_{3,4} = 15.85 - 9.93 = 5.65°C$

$\theta_{4,5}$ μεταξύ διακένου – τούβλου: $\Delta\theta_4 = \frac{d_4}{\lambda_4} * U * \Delta\theta = 0.18 * 0.678 * 20 = 2.44$ $\kappa\alpha\iota$ $\theta_{1,2} = 5.65 - 2.44 = 3.21°C$

$\theta_{5,6}$ μεταξύ επιχρίσματος – τούβλου: $\Delta\theta_5 = \frac{d_5}{\lambda_5} * U * \Delta\theta = \frac{0.09}{0.52} * 0.678 * 20 = 0.31$ $\kappa\alpha\iota$ $\theta_{1,2} = 3.21 - 2.35 = 0.86°C$

θ_6 στην στρώση του επιχρίσματος : $\Delta\theta_6 = \frac{d_6}{\lambda_6} * U * \Delta\theta = \frac{0.02}{0.87} * 0.678 * 20 = 0.31$ $\kappa\alpha\iota$ $\theta_{1,2} = 0.86 - 0.31 = 0.55°C$

Βήμα 3ο : Υπολογισμός των ισοδυνάμων παχών (για το διάγραμμα Glaser)

Από τα δεδομένα έχουμε διαδοχικά:

- $\mu_1 * d_1 = \mu_6 * d_6 = 15 * 0,02 = 0,3$ [m]
- $\mu_2 * d_2 = \mu_5 * d_5 = 10 * 0,09 = 0,9$ [m]
- $\mu_3 * d_3 = 2 * 0,03 = 0,06$ [m]
- $\mu_4 * d_4 = 10,05 = 0,05$ [m]

Και κατασκευάζουμε το ισοδύναμο τοίχωμα, με κατάλληλη κλίμακα.

Βήμα 4ο : Εύρεση των πιέσεων κορεσμού υδρατμών στις επιφάνειες επαφής.

Βρίσκουμε τις πιέσεις κορεσμού (τάσεις κορεσμένων υδρατμών), στις θερμοκρασίες που υπολογίστηκαν στο βήμα 2. Έχουμε:

- Για $\theta_1 = 18,24°C$ βρίσκουμε $P_{κ1} = 2097$ [Pa]
- Για $\theta_{12} = 17,93°C$ βρίσκουμε $P_{κ12} = 2056$ [Pa]
- Για $\theta_{23} = 15,58°C$ βρίσκουμε $P_{κ23} = 1770$ [Pa]
- Για $\theta_{34} = 5,65°C$ βρίσκουμε $P_{κ34} = 910$ [Pa]
- Για $\theta_{45} = 3,21°C$ βρίσκουμε $P_{κ45} = 771$ [Pa]
- Για $\theta_{56} = 0,86°C$ βρίσκουμε $P_{κ56} = 651$ [Pa]
- Για $\theta_2 = 0,55°C$ βρίσκουμε $P_{κ2} = 637$ [Pa]

Βήμα 5ο: Υπολογισμός των δύο ακραίων μερικών πιέσεων (τάσεων) των υδρατμών

Οι μερικές πιέσεις υδρατμών στις δύο πλευρές του τοιχώματος είναι:

- Στην εσωτερική πλευρά: $P_i = \phi_i . P_{κ,i} = 50\% . 2340 = 1170$ [Pa].
- Στην εξωτερική πλευρά: $P_e = \phi_e . P_{κ,e} = 80\% . 611 = 489$ [Pa].

Επειδή στις δύο πλευρές του τοιχώματος οι συντελεστές μεταβίβασης των υδρατμών , βi και β_e, είναι πολύ μεγάλοι, δεχόμαστε πρακτικά τις ίδιες μερικές πιέσεις υδρατμών και πάνω στις αντίστοιχες επιφάνειες, δηλαδή:

- $P_1 = 1170$ [Pa] και

- P_2 = 489 [Pa]

Έτσι, σε όλο το πάχος του τοιχώματος, η συνολική διαφορά (πτώση) τάσεων των υδρατμών είναι: ΔP= P_1 — P_2 = 1170 — 489 = 681[Pa].

Βήμα 6ο : Υπολογισμός του συντελεστή διαπερατότητας υδρατμών του τοιχώματος

$$\frac{1}{K_D} = \sum \frac{d_j}{\delta_j} \ (\alpha\varphi o\upsilon\ \beta_i\ \kappa\alpha\iota\ \beta_e\ \epsilon\acute{\iota}\nu\alpha\iota\ \pi o\lambda\acute{\upsilon}\ \mu\epsilon\gamma\acute{\alpha}\lambda o\iota, o\pi\acute{o}\tau\epsilon\ \frac{1}{\beta_i} = \frac{1}{\beta_e} \approx 0)$$

Όπου, για κάθε στρώση, η αντίσταση διάχυσης των υδρατμών $\frac{d_j}{\delta_j}$ $\pi\rho o\kappa\acute{\upsilon}\pi\tau\epsilon\iota$:

$$\delta_j \approx \frac{636 * 10^{-9}}{\mu_j} \Rightarrow \frac{d_j}{\delta_j} = \frac{d_j * \mu_j}{636 * 10^{-9}} \Rightarrow \frac{d_j}{\delta_j} = 1.572 * 10^6 * d_j * \mu_j$$

Άρα, έχουμε κατά σειρά:

$$\frac{d_1}{\delta_1} = \frac{d_6}{\delta_6} = 1.572 * 10^6 * 0{,}3 = 471600$$

$$\frac{d_5}{\delta_5} = \frac{d_2}{\delta_2} = 1.572 * 10^6 * 0{,}9 = 1415094$$

$$\frac{d_3}{\delta_3} = 1.572 * 10^6 * 0{,}06 = 94340$$

$$\frac{d_4}{\delta_4} = 1.572 * 10^6 * 0{,}05 = 78600$$

Οπότε:

$$K_D = \frac{1}{2 * \frac{d_1}{\delta_1} + 2 * \frac{d_2}{\delta_2} + \frac{d_3}{\delta_3} + \frac{d_4}{\delta_4}} =$$

$$K_D \approx \mathbf{2.534 * 10^{-7}} \left[\frac{\boldsymbol{kg}}{\boldsymbol{m^2 * h * Pa}}\right]$$

Βήμα 7^{o} : Υπολογισμός των μερικών πιέσεων (τάσεων των υδρατμών στις επιφάνειες επαφής P_j

Η συνολική διαφορά (πτώση) τάσεων των υδρατμών έχει βρεθεί: ΔP=681 Pa

Οπότε, για τις επιμέρους θερμοκρασιακές πτώσεις, έχουμε διαδοχικά:

- $P_{1,2}$, μεταξύ επιχρίσματος - τούβλου: $K_D * \Delta P = \frac{\delta_1}{d_1} * \Delta P_1 \Rightarrow \Delta P_1 = \frac{d_1}{\delta_1} * K_D * \Delta P$ $\Rightarrow \Delta P_1 = 471600 * 2.534 * 10^{-7} * 681 \approx 81$ Pa και $P_{1,2} = 1170 - 81 = 1089$ Pa

- $P_{2,3}$, μεταξύ τούβλου - υαλοβάμβακα: $K_D * \Delta P = \frac{\delta_2}{d_2} * \Delta P_2 \Rightarrow \Delta P_2 = \frac{d_2}{\delta_2} * K_D * \Delta P$ $\Rightarrow \Delta P_2 = 1414800 * 2.534 * 10^{-7} * 681 \approx 244$ Pa και $P_{2,3} = 1089 - 244 = 845$ Pa

- $P_{3,4}$, μεταξύ υαλοβάμβακα - διακένου: $K_D * \Delta P = \frac{\delta_3}{d_3} * \Delta P_3 \Rightarrow \Delta P_3 = \frac{d_3}{\delta_3} * K_D * \Delta P$ $\Rightarrow \Delta P_3 = 94320 * 2.534 * 10^{-7} * 681 \approx 16$ Pa και $P_{3,4} = 845 - 16 = 829$ Pa

-$P_{4,5}$, μεταξύ διακένου - τούβλου: $K_D * \Delta P = \frac{\delta_4}{d_4} * \Delta P_4 \Rightarrow \Delta P_4 = \frac{d_4}{\delta_4} * K_D * \Delta P \Rightarrow$

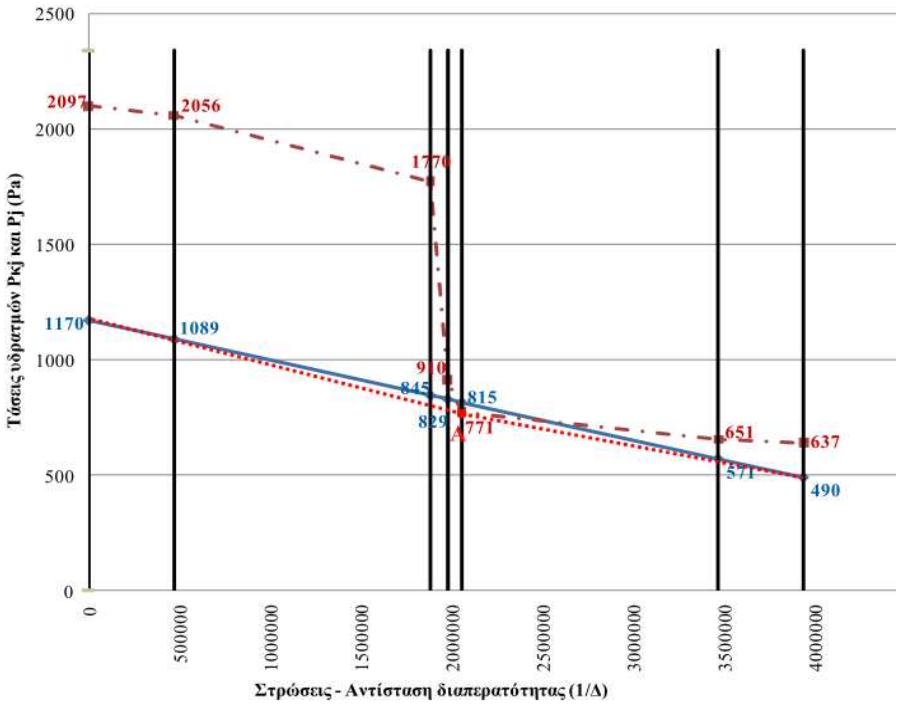
$\Delta P_4 = 78600 * 2.534 * 10^{-7} * 681 \approx 14$Pa και $P_{4,5} = 829 - 14 = 815$ Pa

- $P_{5,6}$, μεταξύ επιχρίσματος - τούβλου: $K_D * \Delta P = \frac{\delta_5}{d_5} * \Delta P_5 \Rightarrow \Delta P_5 = \frac{d_5}{\delta_5} * K_D * \Delta P$ $\Rightarrow \Delta P_5 = 1414800 * 2.534 * 10^{-7} * 244 \approx 81$ Pa και $P_{5,6} = 815 - 255 = 571$ Pa

- P_6, στην στρώση του επιχρίσματος: $K_D * \Delta P = \frac{\delta_6}{d_6} * \Delta P_1 \Rightarrow \Delta P_6 = \frac{d_6}{\delta_6} * K_D * \Delta P$ $\Rightarrow \Delta P_5 = 471600 * 2.534 * 10^{-7} * 681 \approx 81$ Pa και $P_2 = 571 - 81 = 489$ Pa

Βήμα 8°: Κατασκευή των διαγραμμάτων των πιέσεων $\boldsymbol{P_{kj}}$και $\boldsymbol{P_j}$(Περίοδος δρόσου)

Με τις τιμές των πιέσεων κορεσμού και των μερικών πιέσεων, που βρέθηκαν για κάθε διαχωριστική επιφάνεια, χαράσσονται τα αντίστοιχα διαγράμματα, υπό κλίμακα, στο διάγραμμα ισοδυνάμων παχών.



Από το διάγραμμα φαίνεται ότι οι δύο καμπύλες τέμνονται σε ένα σημείο και επομένως αναμένεται εμφάνιση νερού συμπύκνωσης, σε ένα επίπεδο, μέσα στο τοίχωμα. Σε μια τέτοια περίπτωση, το σημείο όπου γίνεται η υγροποίηση είναι το σημείο όπου συναντώνται οι ευθείες που άγονται από τις δύο ακραίες πιέσεις P1 και P2 κατώτερο άκρο της καμπύλης των πιέσεων κορεσμού. Στο παράδειγμα είναι το σημείο Α, στην αριστερή πλευρά του τούβλου, μετά το διάκενο αέρα.

Βήμα 9°: Υπολογισμός ποσότητας υγροποιούμενων υδρατμών (περίοδος δρόσου) W_{dr}

Αρχικά θα υπολογισθεί η ανά ώρα ποσότητα:

$$g_{\delta p} = g_w - g_k$$

Όπου: g_w η ποσότητα των υδρατμών, που εισέρχονται από την θερμή πλευρά μέχρι το επίπεδο συμπύκνωσης, σε kg/m^2h,

Και g_k η ποσότητα των υδρατμών, που εξέρχονται μετά το επίπεδο συμπύκνωσης από την ψυχρή πλευρά, σε kg/m_2h

Η ποσότητα των υδρατμών g_w υπολογίζεται από την σχέση: $g_w = \frac{P_1 - P_A}{\frac{1}{\Delta_1}}$, οπού με $\frac{1}{\Delta_1}$ συμβολίζεται το άθροισμά των αντιστάσεων διάχυσης των υδρατμών $\frac{d_j}{\delta_j}$ μεταξύ της θερμής πλευράς και της θέσης Α. Στο παράδειγμά είναι: $\frac{1}{\Delta_1} = \frac{d_1}{\delta_1} + \frac{d_2}{\delta_2} + \frac{d_3}{\delta_3} + \frac{d_4}{\delta_4} = 2059340 \frac{m^{2h}.Pa}{kg}$

Άρα: $g_w = \frac{1170-771}{2059340} = 0{,}000194 \frac{kg}{h*m^2}$

Η ποσότητα των υδρατμών g_k υπολογίζεται από την σχέση: $g_k = \frac{P_A - P_2}{\frac{1}{\Delta_2}}$, όπου με $1/\Delta_2$ συμβολίζεται το άθροισμα των αντιστάσεων διάχυσης των υδρατμών $\frac{d_j}{\delta_j}$ μεταξύ της θέσης Α και της ψυχρής πλευράς.

Στο παράδειγμα, είναι: $\frac{1}{\Delta_2} = \frac{d_5}{\delta_5} + \frac{d_6}{\delta_6} = 1414800 + 471600 = 1886400 \left[\frac{m^2 * h * Pa}{kg}\right]$.

Άρα $g_k = \frac{771-490}{1886400}$=0.000149[Kg/m^2*h].

Και η ανά ώρα ποσότητα $g_{\delta\rho} = 0.000194 - 0.000149 = 0.000045 \left[\frac{Kg}{h*m^2}\right]$

Για περίοδο δρόσου $t_{\delta\rho} = 2160\ [h]$, βρίσκουμε $W_{\delta\rho} = 0.0972 \left[\frac{Kg}{m^2}\right]$.

Βήμα 10^{ο}Έλεγχος εξάτμισης

Εάν για κάποιο τοίχωμα έχει αποκλειστεί ότι η συμπύκνωση γίνεται σε ευπαθείς στρώσεις και ότι η συγκρατούμενη υγρασία, σε κάποια στρώση, στο τέλος της χειμερινής περιόδου, είναι μικρότερη από την μέγιστη επιτρεπόμενη για αυτή τη στρώση, τότε πρέπει να ελεγχθεί αν επιτυγχάνεται πλήρης εξάτμιση της συγκρατούμενης στην περίοδο δρόσου υγρασίας κατά την διάρκεια του καλοκαιριού (περίοδος ξήρανσης)

Κατά την περίοδο ξήρανσης η διάχυση των υδρατμών γίνεται από το τοίχωμα προς το περιβάλλον. Συγκεκριμένα, οι υδρατμοί θεωρείται ότι διαχέονται από τη θέση που σχηματίστηκαν κατά την περίοδο δρόσου τόσο προς το εξωτερικό περιβάλλον, όσο και προς τον χώρο εσωτερικά.

Για την κατασκευή του αντίστοιχου διαγράμματος Glasser, ακολουθείται παρόμοια διαδικασία, όπως και για την περίοδο δρόσου, αλλά με καθορισμένη την τάση κορεσμένων υδρατμών (περιβάλλον) και την μερική πίεση υδρατμών στο τοίχωμα. Έτσι, βρίσκουμε την πίεση κορεσμού (τάση κορεσμένων υδρατμών), ίση και στις δύο πλευρές του τοιχώματος

Θεωρώντας συνθήκες χώρου και περιβάλλοντος: 12C και ϕ=70% έχουμε:

$$P_{k,i} = P_{k,e} = 1403\ Pa\ (Από\ πίνακα)$$

Επίσης, οι μερικές πιέσεις υδρατμών στις δύο πλευρές του τοιχώματος είναι ίσες και υπολογίζονται για την επικρατούσα σχετική υγρασία (σε χώρο και περιβάλλον)

$$P_i = \varphi_i *> P_{\kappa,i} = 70\% * 1403 = 982Pa = P_e$$

και (όπως πριν $\beta_ι$ και β_e) δεχόμαστε πρακτικά τις ίδιες μερικές πιέσεις υδρατμών και πάνω στις αντίστοιχες επιφάνειες, δηλαδή:

$$P_1 = 982 \text{ [Pa]} = P_2$$

Για τον υπολογισμό της ποσότητας εξατμιζόμενου νερού $W_{\varepsilon\xi}$υπολογίζουμε την ανά ώρα ποσότητα $g_{\varepsilon\xi} = g_w + g_k$ οπού: g_w η ποσότητα των υδρατμών, που εξέρχονται από το επίπεδο συμπύκνωσης, προς την εσωτερική πλευρά, και g_k την ποσότητα των υδρατμών, που εξέρχονται από το επίπεδο συμπύκνωσης προς την εξωτερική.

Η ποσότητα των υδρατμών g_w υπολογίζεται από την σχέση:

$g_w = \frac{P_A - P_1}{\frac{1}{\Delta_1}}$,

οπού 1/Δ1 είναι το ίδιο άθροισμα των αντιστάσεων διάχυσης των υδρατμών που έχει υπολογισθεί προηγουμένως.

Άρα g_w = 0.000204 [Kg/m^{2}h]

Η ποσότητα των υδρατμών g_k υπολογίζεται από την σχέση $g_k = \frac{P_A - P_2}{\frac{1}{\Delta_2}}$ οπού $1/\Delta_2$ είναι το ίδιο άθροισμα των αντιστάσεων διάχυσης των υδρατμών που έχει υπολογιστεί προηγουμένως. *Άρα g_k = 0.000223* [kg/m^{2}h].

Και η ανά ώρα ποσότητα $g_{\varepsilon\xi}$ = 0.000427 [kg/m^{2}h].

Για περίοδο εξάτμισης Τεξ = 3600[h] έχουμε:

$W_{\varepsilon\xi}$=0.000427*3600=1.537 [kg/m^2].

Αφού $W_{\varepsilon\xi} > W_{\delta\rho}$ έπεται ότι κατά τους καλοκαιρινούς μήνες πραγματοποιείται ολοκληρωτική εξάτμιση του συμπυκνώματος που δημιουργήθηκε κατά τους χειμερινούς μήνες.

$\theta(^oC)$	0	2	4	6	8	10	12	14	16	18	20
Pa	611	705	813	934	1073	1228	1402	1598	1817	2063	2338

Πίνακας μερικών πιέσεων υδρατμών του αέρα για διαφορές θερμοκρασίες περιβάλλοντος

5.2 Δυνατότητες που πρέπει να έχει το πρόγραμμα

Εξετάζοντας την παραπάνω άσκηση οι δυνατότητες που θέλουμε να έχει το πρόγραμμα είναι οι εξής:

Να εξετάζει αν:

- υπάρχει κίνδυνος δημιουργίας συμπυκνωμάτων πάνω στην εσωτερική επιφάνεια του τοίχου.
- υπάρχει κίνδυνος δημιουργίας συμπυκνωμάτων μέσα στο τοίχωμα.
- πραγματοποιείται ολοκληρωτική εξάτμιση του συμπυκνώματος που δημιουργήθηκε κατά τους χειμερινούς μήνες.

Να υπολογίζει:

- Ποια είναι η μεγίστη επιτρεπόμενη σχετική υγρασία μέσα στον χώρο για την οποία δεν υπάρχει κίνδυνος δημιουργίας συμπυκνωμάτων στην εσωτερική επιφάνεια του τοίχου.
- Ποιος πρέπει να είναι ο συντελεστής θερμοπερατότητας του τοίχου ώστε να μη υπάρχει κίνδυνος δημιουργίας συμπυκνωμάτων ακόμα και για υγρασία 100%
- Ποιος συντελεστή θερμοπερατότητας του τοιχώματος U.
- Ποιες είναι οι θερμοκρασίες στις επιφάνειες επαφής όλων των στρωμάτων.
- Την πίεση κορεσμού υδρατμών στις επιφάνειες επαφής όλων των στρωμάτων.
- Τις δυο ακραίες μερικές πιέσεων τάσεων των υδρατμών .
- Τον συντελεστή διαπερατότητας υδρατμών του ισοδυνάμου τοιχώματος.
- Της μερικών πιέσεων τάσεων των υδρατμών στις επιφάνειες επαφής των τοιχωμάτων.
- Την ποσότητα υγροποιούμενων υδρατμών την περίοδο δρόσου.
- Την ποσότητα εξατμιζόμενου νερού την περίοδο ξήρανσης.

5.3 Περιγραφή των αντικειμένων που θα χρησιμοποιήσουμε

Ας δούμε τώρα τα κυρία αντικείμενα που θα χρησιμοποιήσουμε.

Η άσκηση περιλαμβάνει τις εξής πληροφορίες:

Τα αρχικά δεδομένα που δίνονται, δηλαδή την θερμοκρασία του αέρα στα 2 περιβάλλοντα που είναι το σύνθετο τοίχωμα ανάμεσα, την σχετική υγρασία του περιβάλλοντος και από τις 2 πλευρές, και τα στρώματα των υλικών. Επίσης δίνεται και ο συντελεστής μεταβίβασης θερμότητας για εσωτερικούς και εξωτερικούς χώρους.

Στα στρώματα των υλικών ο δίνουμε τον τύπο του υλικού, το πάχος του υλικού, τον συντελεστή θερμικής αγωγιμότητας, και τον συντελεστή διαπίδυσης υδρατμών (οι 2 τελευταίοι βρίσκονται από πινάκες).

Κατά την επίλυση της άσκησης υπολογίζουμε, τον συντελεστή θερμικής αγωγιμότητας για το σύνθετο τοίχωμα, την πυκνότητα θερμορροής, την πίεση των υδρατμών στις 2 πλευρές και την διαφορά τους, καθώς και το συντελεστή διαπερατότητας υδρατμών (κ_d) για το σύνθετο τοίχωμα.

Επίσης βρίσκουμε για κάθε στρώμα υλικού του τοιχώματος, την θερμοκρασία των επιφανίων του (και από τις δύο πλευρές του), καθώς και την πτώση θερμοκρασίας που προκαλεί, την αντίσταση διαπερατότητας των υδρατμών του, την πίεση που δέχονται οι επιφάνειες του από την τάση κορεσμένων υδρατμών του, και την μερική τάση των υδρατμών του στις επιφάνειες του.

Συμφώνα με τους τρόπους που αναλύσαμε στο προηγούμενο κεφάλαιο, έχει νόημα να έχουμε τα εξής αντικείμενα:

Ένα αντικείμενο για κάθε στρώμα το οποίο έχει μεταβλητές για κάθε ένα από τα αποτελέσματα που υπολογίζουμε, και μεθόδους για την καταχώρηση και για την προσπέλαση αυτών των μεταβλητών. Ας ονομάσουμε αυτό το αντικείμενο **«Layer»(στρώμα)**

Ένα υπεραντικείμενο που περιλαμβάνει όλα τα Layer και τους μεταβλητές και συντελεστές για το σύνθετο τοίχωμα. Ας ονομάσουμε αυτό το αντικείμενο **«wall» (τοίχωμα)**

Προφανώς μέχρι το τέλος θα έχουμε προσθέσει και αλλά αντικείμενα αλλά αυτά είναι τα προφανή και τα πιο σημαντικά, τα υπόλοιπα βλέποντας και κάνοντας αναλόγως τα προβλήματα που θα παρουσιαστούν γράφοντας το πρόγραμμα.

Έχουμε επίσης και αντίστοιχες μεθόδους για κάθε υπολογισμό, που χρειάζεσαι να γίνει.

Σκεφτόμαστε στο μυαλό μας πως θα λειτούργει το πρόγραμμα χωρίς πολλές λεπτομέρειες, και μετά προβαίνουμε στον σχεδιασμό των καρτών CRC.

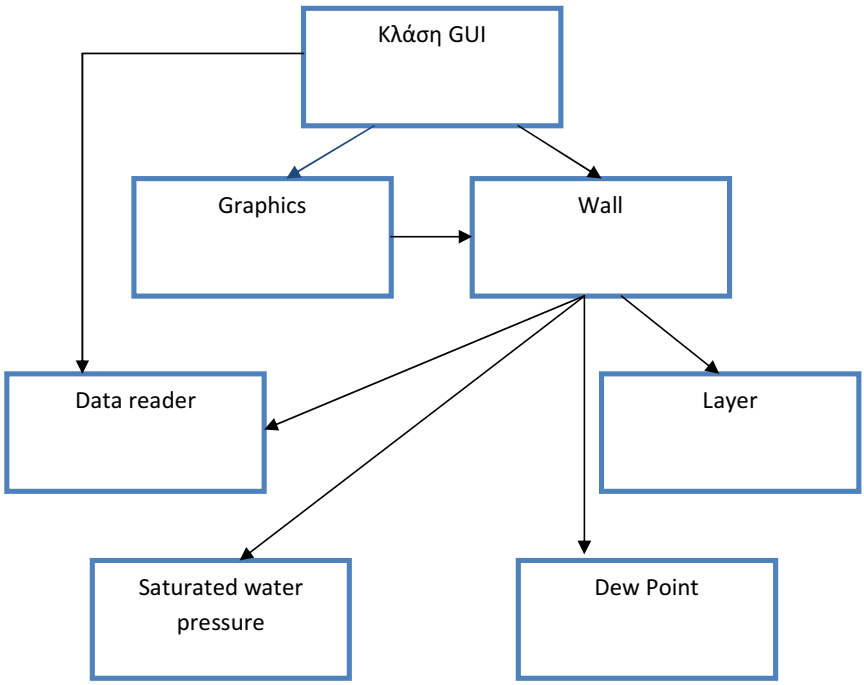
Το πρόγραμμα θα ζητεί από τον χρηστή να δώσει την εσωτερική(την αριστερή), και την εξωτερική(την δεξιά) θερμοκρασία, καθώς και τις αντίστοιχες υγρασίες. Μετά θα πρέπει να δώσει τα επιμέρους στρώματα που απαρτίζουν το τοίχωμα. Το όνομα του υλικού, και το πάχος του. Επειδή ο χρήστης δεν πρέπει να είναι αναγκασμένος να ψάχνει για τιμές από πινάκες τις τιμές των συντελεστών θερμικής αγωγιμότητας και των συντελεστών αντίστασης διαπίδυσης υδρατμών, κάθε φορά που πρέπει να προσθέσει ένα υλικό πιθανός θα πρέπει να έχουμε και ένα αντικείμενο που να περιλαμβάνει αυτά τα δεδομένα από τους πινάκες για τους διαφόρους τύπους υλικών. Ακόμα το πρόγραμμα δεν πρέπει να είναι περιορισμένο σε ένα μόνο νούμερο στρωμάτων υλικού. Ένα τοίχωμα μπορεί να απαρτίζετε από ένα, δυο, ακόμα και δέκα υλικά. Αυτό πρέπει να ληφθεί υπ όψιν. Άρα κάθε φορά που ο χρηστής προσθέτει ένα στρώμα το πρόγραμμα θα πρέπει ανάλογα με το όνομα του υλικού να αναζητά τους συντελεστές του, και να δημιουργεί ένα αντικείμενο **«Layer»** με αρχικές

τιμές το όνομα του υλικού, του πάχος του, και τους 2 συντελεστές του. Πρέπει να υπάρχει κάποιος μετρητής επίσης που κρατεί τον αριθμό των στρωμάτων.

Αφού τελειώσουν οι καταχωρήσεις των αρχικών δεδομένων και αφού πατηθεί ένα κουμπί θα ξεκινάμε οι υπολογισμοί.

Θα υπολογίζει τον συντελεστή θερμοπερατότητας και θα τον αποθηκεύει σε μια μεταβλητή στο αντικείμενο **wall.**
θα υπολογίζει την πυκνότητα θερμορροής.
Θα υπολογίζει στης θερμοκρασίες για κάθε επιφάνεια κάθε στρώματος.
Θα υπολογίζει τις πιέσεις υδρατμών για κάθε επιφάνεια κάθε στρώματος.
Θα γραφεί μια αναφορά με τα αποτελέσματα για την εύκολη παρουσίαση στον χρηστή.

Κάνοντας ένα βασικό σχεδιασμό, προκύπτει το παρακάτω σχεδιάγραμμα.



Σχόλιο [n1]: Ισως σε αυτό το σημείο να ήταν καλό να βάλεις δύο png με τις κλάσεις για
A) το waterinsulation
b) το waterinsulationgui.
Για να υπάρχει μια εποπτεία του προβλήματος.

Σχόλιο [n2]: Ισως σε αυτό το σημείο να ήταν καλό να βάλεις δύο png με τις κλάσεις για
A) το waterinsulation
b) το waterinsulationgui.
Για να υπάρχει μια εποπτεία του προβλήματος.

Σχόλιο [h3]: Βασικα δεν καταλαβά τι εννοεις εδώ. Crc h uml?

NP: UML ennoousa pou ta exeis kanei sto umlet.. Alla kai ta CRC pou eftia3es einai ok.

5.4 Η κλάση layer

5.4.1 Κάρτα CRC

layer	
Συγκράτηση των δεδομένων για κάθε στρώμα	

5.4.2 Ιδιότητες layer

Είπαμε ότι πρέπει να έχει μεταβλητές για κάθε ένα από τα αποτελέσματα που υπολογίζουμε, άρα πρέπει να έχει μια μεταβλητή για:

- Το όνομα του υλικού
- Το πλάτος του (d)
- Τον συντελεστή θερμικής αγωγιμότητας του
- Την θερμική αντίσταση του (d/λ)
- Την θερμοκρασία του στην αριστερή πλευρά του
- Την θερμοκρασία του στην δεξιά πλευρά του
- Την πτώση θερμοκρασίας του
- Τον συντελεστή αντίστασης διάχυσης υδρατμών του
- Την αντίσταση διαπερατότητας υδρατμών του υλικού
- Την τάση κορεσμένων υδρατμών του στην αριστερή του πλευρά
- Την τάση κορεσμένων υδρατμών του στην δεξιά του πλευρά
- Την μερική τάση υδρατμών του στην αριστερή του πλευρά
- Την μερική τάση υδρατμών του στην δεξιά του πλευρά
- Την πτώση της μερικής τάσης υδρατμών του

Άρα έχουμε:

```
Private String name;
//συντελεστής θερμικής αγωγιμότητας
private double CTC;
//συντελεστής αντίστασης διαπίδυσης υδρατμών
private double WVR;
//πλάτος
private double width;
```

```
  //θερμοκρασία στην αριστερή πλευρά
  private double temperature;
  //πτώση θερμοκρασίας μέσα από το υλικό
  private double temperatureFall;
  //θερμοκρασία στην δεξιά πλευρά
  private double temperaturePlus1;
  //αντίσταση διαπερατότητας υδρατμών
  private double antistasiDiaperatotitasYdratmwn;
  //τάση κορεσμένων υδρατμών στην αριστερή πλευρά του υλικού
  private double tasiKoresmenwnIdratmwn;
  //τάση κορεσμένων υδρατμών στην δεξιά πλευρά του υλικού
  private double tasiKoresmenwnIdratmwnPlus1;
  private double merikiTasiYdratmwn;
  private double merikiTasiYdratmwnPlus1;
```

Η λέξη κλειδί **private** χρησιμοποιείται για να περιορίσει την πρόσβαση των μεταβλητών μιας κλάσης μόνο σε μεθόδους που βρίσκονται και εκείνοι μέσα στην κλάση αυτή.

Χρησιμοποιούμε μεταβλητές τύπου double γιατί είναι ο τύπος μεταβλητών που περιγράφει πραγματικούς αριθμούς όπως τα αποτελέσματα που θέλουμε να βγάλουμε[1].

5.4.3 Μέθοδοι πρόσβασης

Θέλουμε επίσης μεθόδους για να θέτουμε κάθε μια από αυτές τις μεταβλητές.

Παραδείγματος χάρη για το πλάτος:

```
public void setWidth(double width)
  {
    this.width=width;
  }
```

Η μέθοδος αυτή δέχεται μια τιμή τύπου **double** και την καταχωρεί στο πεδίο «**width**» του αντικειμένου

Παρομοίως και για τις υπόλοιπες μεταβλητές.

Θέλουμε ακόμα μεταβλητές που να μας επιτρέπουν την πρόσβαση στα δεδομένα που περιλαμβάνει το αντικείμενο.

Παραδείγματος χάρη για το πλάτος πάλι:

```
 public double getWidth()
   {
    return this.width;
   }
```

[1] Για χρήσεις που τα αποτελέσματα του προγράμματος είναι σημαντικό να έχουν μεγαλύτερη ακρίβεια είναι καλύτερο να χρησιμοποιούνται αντί για **double** ο τύπος αριθμών **big decima**l που βρίσκετε στην **java.util.math**

Η μέθοδος αυτή επιτρέπει να κληθεί εκτός της κλάσης και να επιστρέψει την μεταβλητή «width» που βρίσκεται μέσα στο αντικείμενο. Με την ίδια λογική κατασκευάζουμε και τις άλλες μεθόδους για κάθε μεταβλητή της κλάσης που θέλουμε να έχουμε πρόσβαση εξωτερικά.

Τέλος για να γλυτώσουμε την χρήση κάποιων μεταβλητών μπορούμε να κάνουμε ορισμένες πράξεις μέσα στις μεθόδους επιστροφής

Για παράδειγμα δεν δηλώσαμε μεταβλητή για την θερμική αντίσταση του υλικού **(d/λ)** γιατί μπορούμε κάλλιστα εφόσον και το **d** και το **λ** περιέχονται ήδη μέσα στο αντικείμενο να κάνουμε την πράξη στην μέθοδο επιστροφής:

```
public double getWidthdivCTCRatio()
  {
    return (width/CTC);
  }
```

5.4.4 Δομητής layer

Τέλος ας φτιάξουμε τον δομητή. Σε μια κλάση ο δομητής είναι μια ειδική μέθοδος η όποια καλείται την στιγμή της δημιουργίας ενός αντικειμένου. Προετοιμάζει το νέο αντικείμενο για χρήση, και συχνά δέχεται παραμέτρους τις οποίες τις χρησιμοποιεί για να θέσει ότι μεταβλητές χρειάζονται στην δημιουργία του αντικειμένου.

```
public Layer(String layerName, double layerWidth, double layerCTC, double
layerWVR)
  {
    //populate the fields
    name=layerName;
    width=layerWidth;
    CTC=layerCTC;
    WVR=layerWVR;
  }
```

Κάθε φορά που κατασκευάζουμε ένα αντικείμενο στρώματος, τα αρχικά δεδομένα που χρειαζόμαστε για να υπολογίσουμε τα υπόλοιπα είναι μόνο το πάχος του και οι 2 αυτοί συντελεστές που το χαρακτηρίζουν, έχει νόημα αυτές οι μεταβλητές να δηλώνονται στον δομητή κατευθείαν.

5.5 Η κλάση layerData

Ας φτιάξουμε την κλάση **LayerData**. Η κλάση αυτή θα περιέχει τις μεθόδους που θα αναζητούν τους συντελεστές θερμικής αγωγιμότητας, και της αντίστασης διαπίδυσης υδρατμών.

Το μόνο που θέλουμε να δέχεται η μέθοδος που αναζητά είναι το όνομα του υλικού, και να επιστρέφει τους 2 συντελεστές. Επειδή μια μέθοδος δεν μπορεί να επιστρέφει 2 τιμές ταυτόχρονα απλά θα βάλουμε δυο πεδία μέσα στην κλάση για να αποθηκεύουν αυτές τις τιμές και θα τις πάρουμε με **get** μεθόδους.

Άρα:

```
public class LayerData {
  private double CTC;
  private double WVR;
  public void FindElements(double name)
  {
   …
  }
}
```

Τα δεδομένα θα τα βάλουμε μέσα σε ένα έγγραφο κείμενου με αυτήν την μορφή

```
Ονομα#συντελεστης#αλλος συντελεστής
Άλλο ονομα#συντελεστης#αλλος συντελεστής
```

Άρα πρέπει να έχουμε και μία μέθοδο που να διαβάζει αυτό το αρχείο και να επιστρέφει τα δεδομένα του.

```
   /**
   * diavazei ena arxio kai pernaei ta periexomena tou se ena arraylist
   * @param filename to onoma tou arxeiou
   */
public class LayerData {
    private double CTC;
    private double WVR;
    private ArrayList data = new ArrayList();

  private void FileReader(String filename)
  {
     boolean textExist=true;
       try
      {
        BufferedReader inputStream =
        new BufferedReader (new FileReader (filename));

        while (textExist)
        {
         String line = null;
         line = inputStream.readLine(); //reads line;

           if (line != null) //αν η γραμμή δεν είναι κενη
             {
              // System.out.println("Line "+
              (numberOfLines) + " in data.txt is: " + (line));
               textExist=true; // Synthikh ths while, true
               StringTokenizer numberFinder =
              new StringTokenizer(line,"#");//costructon



               while (numberFinder.hasMoreTokens())
               {
                 data.add(numberFinder.nextToken());
                //gemisma metablitou pinaka
               }


             }
           else //when the line is empty
             {
               textExist=false; //txt importing stops
```

```
              // System.out.println(
             "LayerData input from \"data.txt\" is over");
             }

        }
        inputStream.close();
      }
      catch(FileNotFoundException e)
        {
          System.out.println("File data.txt was not found");
          System.out.println("or could not be opened.");
        }
       catch(IOException e)
        {
          System.out.println("Error reading from file data.txt.");
        }

  }
```

Η μέθοδος αυτή διαβάζει το αρχείο κείμενου, χωρίζει κάθε γραμμή στα επιμέρους στοιχεία της (που χωρίζονται μεταξύ τους με τον χαρακτήρα '#') και τα καταχωρεί σε ένα arraylist. Το arraylist είναι ένας μεταβλητός πινάκας, δηλαδή ένας πινάκας μη καθορισμένου μεγέθους, που αυξάνεται ανάλογα με τις ανάγκες μας

Άρα αυτή την στιγμή έχουμε ένα arraylist με το ονόμα "data", μέσα στην κλάση όπου περιλαμβάνει την μέθοδο, σε μια x θέση το όνομα του υλικού και σε μια x+1 και x+2 θέσεις του συντελεστές του.

Άρα

```
public class LayerData {
  private double CTC;
  private double WVR;
  private ArrayList data = new ArrayList();
public void FindElements(String name)
  {
    data.clear();
    FileReader("LayerData.txt");
    //System.out.println("to megethos tou arraylistt einai "+data.size());

    for(int i=0; i<data.size(); i++) //loop
    {
      if(name.equalsIgnoreCase(data.get(i).toString()))
        /*
        * elenxei ola ta stoixeia tou pinaka
        * an to stoixeio simfonei me tin
        * simvoloseira tote kataxorise ta
        * parakatw stoixeia
        */
      {
        CTC=Double.parseDouble(data.get(i+1).toString());
        WVR=Double.parseDouble(data.get(i+2).toString());
      }

    }
  }
```

Υπάρχει όμως ένα πρόβλημα. Το πρόβλημα του αέρα. Στους πινάκες που έχουμε στην διάθεση μας, στην περίπτωση του διακένου γνωρίσουμε μόνο την αντίσταση

θερμοδιαφυγής d/λ και όχι τον συντελεστή θερμικής αγωγιμότητας και αυτόν μόνο για ορισμένα προκαθορισμένα πάχη του διακένου. Στο πρόγραμμα μας δεν πρέπει να περιορίσουμε τον χρηστή σε προκαθορισμένες τιμές πάχους.

Οι τιμές που έχουμε είναι οι εξής

Πάχος στρώματος αέρος [mm]	Αντίσταση θερμοδιαφυγής $\left[\frac{K*m^2}{watt}\right]$
10	0.14
20	0.16
50	0.18
100	0.17
150	0.16

Εφόσον για τα υπόλοιπα στρώματα θα χρησιμοποιήσουμε τον συντελεστή θερμικής αγωγιμότητας για τους υπολογισμούς μας έχει νόημα να κάνουμε και εδώ το ίδιο πράγμα άρα τον υπολογίζουμε

Πάχος στρώματος αέρος (mm)	Θερμική αγωγιμότητα
10	0.071428571
20	0.125
50	0.277777777
100	0.5882352
150	0.9375

Θα βάλουμε αυτά τα δεδομένα σε να αρχείο κείμενου πάλι αλλά αυτή την φορά με την δομή

```
Παχος#συντελεστης
```

Τα δεδομένα που βάζουμε είναι:

```
0.01#0.071428571
0.02#0.125
0.05#0.277777777
0.10#0.5882352
0.15#0.9375
```

Ο συντελεστής αντίστασης διαπίδυσης υδρατμών για τον αέρα δεν χρειάζεται γιατί είναι ίσος με μονάδα.

Θα φτιάξουμε μια μέθοδο λοιπόν η οποία δέχεται το πάχος του διακένου, καλεί πάλι την μέθοδο **FindElements(String name)** που φτιάξαμε πιο πριν για να διαβάζει το αρχείο, και χρησιμοποιεί τα δεδομένα που διαβάζει. Αν ο χρηστής έχει βάλει πάχος στρώματος που υπάρχει στον πινάκα τότε το πρόγραμμα καταχωρεί τον συντελεστή θερμικής αγωγιμότητας που αντιστοιχεί. Αυτό είναι λίγο απίθανο βεβαία μια και ο πινάκας μας έχει μονό πέντε τιμές. Γι αυτό, αν δεν υπάρχει αυτούσιο στον πινάκα, μετατρέπουμε την μέθοδο μας ώστε να κάνει γραμμική παρεμβολή ανάμεσα στις δύο τιμές του πάχους στρώματος

αέρος που είναι διαθέσιμες στον πίνακα. Επειδή για να κάνουμε την σύγκριση για το σε πια νούμερα βρίσκεται ανάμεσα η τιμή που εισάγεται, πρέπει να έχουμε το +1 και το -1 από την τιμή του μετρητή μας, για αυτό και ο μετρητής ξεκινά από το 2.

```
**
   * dexete to paxos tou stromatos aera
   * kai epistrofi ton sintelesti tou thermikis agogimotitas
   * @param width to paxos tou stromatos
   */
   public void FindAirElements(double width)
  {
     data.clear();
    FileReader("AirLayerData.txt");
    //System.out.println("to megethos tou arraylistt einai "+data.size());
/*
        * elenxei ola ta stoixeia tou pinaka
        * an to stoixeio simfonei me tin
        * simvoloseira tote kataxorise ta
        * parakatw stoixeia
        */


    for(int i=2; i<data.size();) //loop
    {
      if(width==(Double.parseDouble(data.get(0).toString())))
       {
        CTC=Double.parseDouble(data.get(1).toString());
       }
      else if(width==(Double.parseDouble(data.get(i).toString())))
      {
        CTC=Double.parseDouble(data.get(i+1).toString());
      }
      /*
       * an den simfonei me tis times tou pinaka
       * brisoume tin sosti timi me gramiki paremboli
       *
       */
      else if((width>Double.parseDouble(data.get(i-2).toString()))&&
          (width<Double.parseDouble(data.get(i).toString())))
      {
        CTC=LinearInterpolation(Double.parseDouble(data.get(i-
2).toString()),
            Double.parseDouble(data.get(i-1).toString()),
            Double.parseDouble(data.get(i).toString()),
            Double.parseDouble(data.get(i+1).toString()), width);
      }
     WVR=1; // ο συντελεστης αντίστασης διαπίδυσης υδρατμών είναι μονάδα.
     i=i+2;
    }
  }
```

Ο τύπος της γραμμικής παρεμβολής είναι:

$$y = y_0 + \frac{(x - x_0) * y_1 - (x - x_0) * y_0}{x_1 - x_0}$$

Η μέθοδος που κάνει γραμμική παρεμβολή ονομάζεται **LinearInterpolation** και ο κώδικας της είναι:

```
   /**
   * kanei gramiki paremboli
   * @param x0 protos sintelestis
```

```
   * @param y0 proto apotelesma
   * @param x1 deuteros sintelestis
   * @param y1 deutero apotelesma
   * @param x endiamesos sintelestis
   * @return i zitoumeni timi
   */
  private double LinearInterpolation(double x0,
  double y0,double x1,double y1,double x)
  {
    double y=y0+((x-x0)*((y1-y0)/(x1-x0)));
    return y;
  }
```

Φυσικά θα χρειαστούμε ακόμα και 2 μεθόδους για να μπορούμε αν διαβάσουμε τις μεταβλητές μας.

```
public double getCTC()
  {
    return CTC;
  }
public double getWVR()
  {
    return WVR;
  }
```

Τέλος λογικά θα χρειαστούμε και μια λίστα με τα υλικά που υπάρχουν στο αρχείο για να το χρησιμοποιήσουμε στο γραφικό περιβάλλον.

Αυτό που χρειαζόμαστε είναι μια μέθοδος που διαβάζει το αρχείο, και σε κάθε σειρά κρατεί μόνο το πρώτο στοιχειό (δηλαδή το όνομα του υλικού), και διαγράφει τα αλλά, στο τέλος επιστρέφοντας ένα πινάκα με την λίστα των υλικών.

Αυτό μπορεί να γίνει ως έχεις: όπως και παραπάνω η μέθοδος αυτή καλεί την μέθοδο **fileReader** που δημιουργεί ένα **arrayList** με τα δεδομένα του αρχείου. Το arraylist τότε θα περιέχει τα στοιχεία με αυτήν την σειρά **0.1.2, 3.4.5, 6.7.8 κτλ**. Εμείς θέλουμε να κρατήσουμε τα στοιχεία 0,3,6 κτλ, που περιέχουν το όνομα. Βλέπουμε ότι τα στοιχεία που θέλουμε να κρατήσουμε είναι πολλαπλάσια του 3, και η λύση δεν είναι και τόσο προφανής:

```
public ArrayList FindElementsForList()
  {
    data.clear();
    FileReader("LayerData.txt");

    /*
     * Για καθε στοιχειό του πινακα διαιρεσε τον δεικτη του
     * αν το υπολοιπο της διαιρεσης με το 3 ειναι διαφορο του 0
     * αφαιρεσε το στοιχειό σε αυτον τον δεικτη
     */
    for(int i=(data.size()-1); i>=0; i--) //loop
    {
      if(i%3!=0)
      {
        data.remove(i);
      }
    }
```

```
    return data;
  }
```

Αυτό γίνεται για όλα τα στοιχεία . Ο λόγος που το κάνουμε ανάποδα είναι γιατί όταν αφαιρείς ένα στοιχειό σε ένα σημείο του arraylist, όλα τα στοιχεία μετά από αυτό πηγαίνουν μια θέση πίσω. Δηλαδή:

```
Arraylist.get(0)=a
Arraylist.get(1)=b
Arraylist.get(2)=c
Arraylist.get(3)=d
```

Αν τώρα αφαιρέσουμε το στοιχειό b το arraylist θα πάρει την μορφή:

```
Arraylist.get(0)=a
Arraylist.get(1)=c
Arraylist.get(2)=d
```

Οπότε αν τρέχαμε τον βρόχο από την αρχή προς το τέλος απλά θα καταστρέφαμε την δομή που είχε η λίστα και δεν θα περνάμε τα αποτελέσματα που θέλουμε[2].

Ο τρόπος ανάκτησης και αποθήκευσης δεδομένων που παρουσιάζεται σε αυτή την κλάση κάθε άλλο παρά αποδοτικός είναι. Είναι πολύ πολύπλοκος και είναι πολύ εύκολο να γίνει λάθος ανάκτηση στον χρόνο του σχεδιασμού μια και χρησιμοποιούμε συγκεκριμένα νούμερα για να πάρουμε της τιμές που μας ενδιαφέρουν. Παρακάτω στο κείμενο θα δούμε την χρήση ενός πιο έξυπνου τρόπου ανάκτησης δεδομένων χρησιμοποιώντας αρχεία XML

[2]Όταν κάνουμε κλήση της μεθόδου **.size()** σε έναν πινάκα η σε ένα **arraylist** η τιμή που επιστρέφεται είναι ένας ακέραιος αριθμός που δηλώνει ποσά στοιχεία υπάρχουν στον πινάκα αυτό. Όταν όμως θέλουμε να προσπελάσουμε τα δεδομένα του πινάκα αυτού ο δείχτης των αντικειμένων του αρχίζει να μετράει από το μηδέν **A[0]** και όχι από το ένα **A[1].** Συνεπώς αν θέλουμε να πάρουμε την τελευταία τιμή ενός μεταβλητού πινάκα που δεν ξέρουμε ποσό θα είναι το μέγεθος του χρησιμοποιούμε την εξής τεχνική:

```
Πινάκας[μεγεθοςΤουΠινακα-1], η αλλιώς πιο προγραμματιστικά: B=A[A.size()-1]
```

5.6 Κλάση saturated water pressure

saturated water pressure	
Υπολογισμός της πίεσης των υδρατμών του αέρα σε mmHG σε συνάρτηση με την θερμοκρασία του	

Η κλάση αυτή στην συγκεκριμένη περίπτωση δεν χρειάζεται να έχει ιδιότητες αντικειμένου μια και το μόνο που κάνει είναι μια απλή αλγεβρική πράξη. Σε τέτοιες περιπτώσεις δηλώνουμε τις μεθόδους τις κλάσης αυτής χρησιμοποιώντας την λέξη κλειδί static. Για να χρησιμοποιήσουμε στατικές μεθόδους δεν χρειάζεται να κατασκευάσουμε αντικείμενο τις κλάσης που τις περιλαμβάνει και η κλήση τους γίνεται με τον εξής τρόπο:

```
ΟνομαΚλασης.ΟνομαΜεθοδου();
```

```
public class SaturatedWaterPressure {
private static final double a1=6.11176750;
private static final double a2=0.443986062;
private static final double a3=0.143053301E-01;
private static final double a4=0.265027242E-03;
private static final double a5=0.302246994E-05;
private static final double a6=0.203886313E-07;
private static final double a7=0.638780966E-10;
private static final double K=273.15;

/**
 * xrisimopoiei tin methodo bolzman gia ton ipologismo
 * pou einai akribis gia times thermocrasias
 * -50-50C
 *
 * @param T Temperature in Kelvin
 * @return the pressure at mmHG(to convert to Pa multiply with 100)
 */
public static double SaturatedWaterPressure(double T){

  double saturatedWaterPressure=0;

   // xrisi metaxi -50 kai 50 C
  // epistrefei tin piesi se mmHG
  //gia metatropi se Pa pollaplasiazoume me 100
  //eisodos se kelvin


  saturatedWaterPressure=a1+(a2*(T-K))+
        (a3*(Math.pow((T-K),2)))+
```

```
        (a4*(Math.pow((T-K),3)))+
        (a5*(Math.pow((T-K),4)))+
        (a6*(Math.pow((T-K),5)))+
        (a7*(Math.pow((T-K),6)));

  return saturatedWaterPressure;
}}
```

Στην συγκεκριμένη περίπτωση η κλήση τις μεθόδου **saturatedWaterPressure** γίνεται:

```
SaturatedWaterPressure.SaturatedWaterPressure(⊖);
```

5.7 Η κλάση wall

5.7.1 CRC card

wall	
Υπολογισμός συντελεστή θερμοπερατότητας του τοιχώματος	
Υπολογισμός των θερμοκρασιών στις επιφάνειες επαφής	
Υπολογισμός των δύο ακραίων μερικών πιέσεων	
Δημιουργία στρώματος υλικού	data, layer
εισαγωγή, αφαίρεση δεδομένων	layer
Υπολογισμός, πυκνότητας θερμορροής	layer
υπολογισμός, τάσης κορεσμένων υδρατμών	saturated water presure,layer
υπολογισμός μερικής τάσης κορεσμένων υδρατμών	layer
υπολογισμός αντίστασης διαπερατότητας υδρατμών	layer
υπολογισμός του συντελεστή διαπερατότητας υδρατμών του τοιχώματος	layer
εγγραφή αναφοράς	layer
Υπολογισμός αν το η θερμοκρασία στην εσωτερική πλευρά του τοίχου είναι μεγαλύτερη από το σημείο δρόσου	dewpoint
Υπολογισμός μέχρι το πια θερμοκρασία δεν θα έχουμε συμπύκνωση στην εσωτερική επιφάνεια του τοιχώματος	dewpoint

5.7.2 Ιδιότητες wall

Ας κατασκευάσουμε τώρα την κλάση **«wall»**

Έγινε η επιλογή αυτή, η κλάση θα περιλαμβάνει και όλες της μεθόδους που κάνουν υπολογισμούς.

Πάμε βήμα-βήμα, βάζοντας αρχικά τις μεταβλητές που ξέρουμε, και προσθέτοντας καινούργιες ανάλογα με τις ανάγκες μας.

Αρχικά λοιπόν είδαμε πριν ότι πρέπει να έχει μεταβλητές για να αποθηκεύσει: τις θερμοκρασίες του περιβάλλοντος στα 2 άκρα του τοιχώματος[3], της συνθήκες υγρασίας στα 2 άκρα του περιβάλλοντος και όλα τα στρώματα των υλικών, δηλαδή έναν πινάκα που περιέχει αντικείμενα layer μέσα του.

```
private double TemperatureInternal=0;
private double TemperatureExternal=0;
private double RelativeHumidityInternal=0;
private double RelativeHumidityExternal=0;
private ArrayList<Layer> layer=new ArrayList<Layer>();
```

Επίσης το αντικείμενο της κλάσης **LayerData** που φτιάξαμε πριν

```
private LayerData Data=new LayerData();
```

5.7.3 Μέθοδοι προσθήκης/αφαίρεσης στοιχείων wall

Πρέπει σκεφτούμε τι θέλουμε να κάνει ο χρηστής με τα δεδομένα των στρωμάτων και να κατασκευάσουμε αντίστοιχες μεθόδους.

Ο χρηστής σίγουρα θα χρειαστεί όταν βάζει τα δεδομένα του να προσθέτει στρώματα υλικού, ίσως να αφαιρέσει ένα στρώμα για να το αντικαταστήσει με ένα αλλά η απλά επειδή έκανε λάθος επιλογή, να αντιστρέψει την θέση δυο στρωμάτων, η ακόμα και να διαγράψει όλα τα στρώματα που έχει προσθέσει με σκοπό να ξαναρχίσει από την αρχή. Ανεξάρτητος το πώς θα είναι τελικά το γραφικό περιβάλλον αυτές οι λειτουργιές πρέπει να υπάρχουν.

Ας κατασκευάσουμε τις αντίστοιχες μεθόδους

Για την προσθήκη αντικειμένου ConstructLayer:

```
public void ConstructLayer(String name,double width)
  {

    // το διάκενο είναι ειδική περίπτωση γιατί τα δεδομένα του
    // τα βρίσκουμε από διαφορετικό αρχείο
```

[3] Υπενθυμίζεται ότι έχει γίνει η σύμβαση ότι η θερμή πλευρά είναι εσωτερικά και ότι είναι στην αριστερή πλευρά του τοιχώματος έτσι όπως το κοιτάμε.

```
    if(name.equalsIgnoreCase("diakeno"))
      Data.FindAirElements(width);
    else
      //βρίσκει τα δεδομένα που μας χρειάζονται για τον τύπο
      //υλικού που βάλαμε
      Data.FindElements(name);

    //τέλος κατασκεύασε το αντικείμενο με τα αρχικά του στοιχεία
    layer.add(new Layer(name,width,Data.getCTC(),Data.getWVR()));
  }
```

Η μέθοδος που φτιάξαμε, είναι όπως την είχαμε σκεφτεί από την αρχή, το μόνο που χρειάζεται για να δουλέψει είναι το όνομα του υλικού, και το πάχος του.

Αν το όνομα που δώσαμε αντιστοιχεί σε **diakeno** τότε το πρόγραμμα καλεί την **FindAirElements** που κατασκευάσαμε στην κλάση **layerData**

Αν όχι τότε καλεί την μέθοδο **findElements** που κατασκευάσαμε στην κλάση layerData

Οποία από τις δυο τελικά και να καλέσει, το αντικείμενο **Data** αποκτάει τις τιμές που χρειαζόμαστε στα πεδία του.

Το τελευταίο βήμα είναι να κατασκευάσουμε ένα νέο στιγμιότυπο αντικειμένου layer. Είχαμε φτιάξει τον δομητή του με τέτοιο τρόπο ώστε να απαιτεί 4 στοιχεία , το όνομα, το πάχος, τον συντελεστή θερμικής αγωγιμότητας, και τον συντελεστή αντίστασης διαπίδυσης υδρατμών για αυτό το υλικό πάντα.

Άρα:

```
στρώμα=new Layer(name,width,Data.getCTC(),Data.getWVR())
```

Και το καταχωρούμε στον μεταβλητό πινάκα:

```
layer.add(στρώμα);
Η χωρίς την παρέμβαση ενδιάμεσης μεταβλητής:
layer.add(new Layer(name,width,Data.getCTC(),Data.getWVR()));
```

Αν όμως θέλουμε να προσθέσουμε αυτό το καινούργιο στρώμα όχι στο τέλος του τοιχώματος αλλά ανάμεσα σε 2 αλλά στρώματα;

Πρέπει να προσθέσουμε μια άλλη μέθοδο ακόμα σχεδόν όμοια με την παραπάνω αλλά με την διαφορά ότι αυτή δέχεται και σαν είσοδο και τον δείκτη του θέσης που θέλουμε να πάει το στρώμα.

```
public void ConstructLayer(String name,double width, int index)
  {

    if(name.equalsIgnoreCase("diakeno"))
      Data.FindAirElements(width);
    else
      Data.FindElements(name);
```

```
    layer.add(index, new  Layer(name,width,Data.getCTC(),Data.getWVR()));
  }
```

Μπορούμε, και χρησιμοποιήσαμε το ίδιο όνομα για αυτήν την μέθοδο με την προηγούμενη, αυτό είναι ένα κλασικό παράδειγμα υπερφόρτωσης.

Η μέθοδος για να την **αφαίρεση ενός στρώματος** (RemoveLayer) κατασκευάζεται πανεύκολα:

```
/**
 * αφαιρει ένα στρώμα υλικού
 * @param index το σημείο που βρίσκεται το υλικό στον πινάκα
 */
  public void RemoveLayer(int index)
  {
    layer.remove(index);
  }
```

Η μέθοδος για να ανταλλάξουμε την θέση μεταξύ δυο στρωμάτων μεταξύ τους θέλει όμως λίγη σκέψη παραπάνω. Στην java γενικά δεν υπάρχει εντολή που να ανταλλάσει της τιμές δυο μεταβλητών μεταξύ τους. Ο τρόπος να τις ανταλλάξουμε είναι να χρησιμοποιήσουμε μια τρίτη προσωρινή μεταβλητή. Δηλαδή έστω ότι έχουμε A=5 και B=8, για να τα ανταλλάξουμε θα κάνουμε:

```
A=5
B=8
Temp=A
A=B
B=Temp
```

Με αυτή την λογική θα ανταλλάξουμε και τις 2 τιμές μέσα στην μέθοδο μας, η οποία σαν είσοδο θα δέχεται 2 ακέραιους που υποδηλώνουν τις θέσεις των προς επεξεργασίας στρωμάτων.

```
 /**
   * Αντιμεταθέτει 2 στρώματα υλικού μεταξύ τους
   * @param a η θέση του πρώτου στρώματος
   * @param b η θέση του δευτέρου στρώματος
   */
  public void swapLayer(int a, int b)
  {
    //πρόσθεσε άλλη μια φορά α στο τέλος του πινάκα
    layer.add(layer.get(a));
    // αντικατέστησε την αρχική θέση του α με το β
    layer.set(a, layer.get(b));
    // αντικατέστησε το β με το α που βάλαμε στο τέλος του πινάκα
    layer.set(b,layer.get(layer.size()-1));
    // αφαίρεσε το α που βάλαμε στο τέλος του πινάκα
    layer.remove(layer.size()-1);
    // τώρα το array list layer ίσως έχει κρατήσει στο
    // index του και την τελευταία θέση με τιμή null
    // αυτό δεν το θέλουμε γιατί καταστρέφει όλο το πρόγραμμα
    //αφαιρούμε όλες της άδειες τιμές
    layer.trimToSize();
```

Στην περίπτωση μας επειδή χρησιμοποιούμε arraylist έχουμε την ευκολία να μην χρησιμοποιήσουμε μια άλλη αυθαίρετη προσωρινή μεταβλητή αλλά τον ίδιο τον πινάκα.

Μένει μόνο η μέθοδος διαγραφής όλων τον υλικών:

```
public void ClearLayer()
  {
    layer.clear();
    layer.trimToSize();
  }
```

5.7.4 Μέθοδοι πρόσβασης δεδομένων

Επόμενο βήμα θα φτιάξουμε τις μεθόδους που θέτουν και που επιστρέφουν μεταβλητές για τα στοιχεία που έχουμε:

```
     /**
   * thetei tin esoteriki thermokrasia
   * @param T thermokrasia
   */

 public void setTemperatureInternal(double T)
  {
    TemperatureInternal=T;
  }
   /**
   * thetei tin exoteriki thermokrasia
   * @param T thermokrasia
   */

 public void setTemperatureExternal(double T)
  {
    TemperatureExternal=T;
  }
   /**
   * thetei tin sxetiki ygrasia tou aera gia
   * ton esoteriko xoro
   * @param RHI i sxetiki igrasia
   */

 public void setRelativeHumidityInternal(double RHI)
  {
    relativeHumidityInternal=RHI;
  }
   /**
   * thetei tin sxetiki ygrasia tou aera gia to
   * exoteriko xoro
   * @param RHE i sxetiki igrasia
   */

 public void setRelativeHumidityExternal(double RHE)
  {
    relativeHumidityExternal=RHE;
  }
```

Προς το παρόν δεν φαίνεται να ότι θα χρειαστούμε πουθενά accessor μεθόδους για αυτές τις μεταβλητές οπότε δεν τις προγραμματίσουμε. Αν φάνει ότι θα την χρειαστούμε θα πράξουμε αναλόγως. Γενικά καλό είναι να μην πνίγουμε το πρόγραμμα μας με άχρηστο κώδικα.

Πάμε να φτιάξουμε τις μεθόδους των υπολογισμών, Πρώτα ας φτιάξουμε την μέθοδο που υπολογίζει τον συντελεστή θερμοπερατότητας του τοιχώματος. Ο συντελεστής θερμοπερατότητας δίνεται από την σχέση:

$$\frac{1}{U}=\frac{1}{h_i}+\frac{d_1}{\lambda_1}+\frac{d_2}{\lambda_2}+\cdots+\frac{d_n}{\lambda_n}+\frac{1}{h_e}\Rightarrow U=\frac{1}{\frac{1}{h_i}+\frac{1}{h_e}+\sum_n^1\frac{d_x}{\lambda_x}}$$

Το h_i και h_e είναι ο συντελεστής μεταβίβαση θερμότητας για εσωτερική και εξωτερική πλευρά κτιρίων και ισούται με 7,7 $\frac{w}{m^2K}$ και 25 $\frac{w}{m^2K}$ αντιστοιχα. Ας τους καταχωρισουμε ως σταθερες στην κλαση μας:

```
Private static final double Hi=7.7;
Private static final double He=25;
```

Εδώ η λέξη κλειδί της java **final** σημαίνει ότι τα πεδία αυτά είναι σταθερά και αμετάβλητα.

Επίσης χρειαζόμαστε και μια μεταβλητή για τον συντελεστή θερμικής αγωγιμότητας που θα υπολογίσουμε

```
Private double thermalTransmittance=0;
```

Φτιάχνοντας την μέθοδο λοιπόν:

```
/**
   * ipologizei ton sintelesti thermoperatotitas
   */
  private void CalculateThermalTransmittance()
  {
    thermalTransmittance=(1/((1/Hi)+(1/He)+CalculateWidthdivCTCSummary()));
  }
```

Όμως δεν έχουμε αυτή την στιγμή το $\sum_n^1\frac{d_x}{\lambda_x}$, οποτε πρέπει να φτιάξουμε μια μέθοδο που το υπολογίζει και αυτό:

```
* ipologizei to arthrisma twn d/λ apo ola ta layers
 */
  private double CalculateWidthdivCTCSummary()
  {
    double summary=0;
    for (int i=0; i<layer.size(); i++)
    {
      Summary+=layer.get(i).getWidthdivCTCRatio();
    }
    widthdivCTCSummary=summary;
    return widthdivCTCSummary;
  }
```

<u>Σημείωση</u> A+=5 σημαίνει A=A+5

Μέθοδος για τον υπολογισμό της πυκνότητας θερμορροής

Η πυκνότητα θερμορροής δίνεται από τον τύπο:

Q=U*(Δθ)

άρα:

```
  /**
   * ipologizei tin piknotita thermorois
   */
  private void CalculateDensityOfThermalTransfer()
  {
    densityOfThermalTransfer=
        thermalTransmittance*(TemperatureInternal-
        TemperatureExternal);
  }
```

Και προσθέτουμε και την μεταβλητή μας για την πυκνότητα θερμορροής στην κλάση.

```
Private double densityOfThermalTransfer=0;
```

Επόμενο βήμα είναι ο υπολογισμός των θερμοκρασιών στις επιφάνειες επαφής. Θα πρέπει να υπολογίσουμε μια μέθοδο που υπολογίζει την θερμοκρασία, σε ένα στρώμα, και στην επιφάνια που δέχεται την θερμότητα (θεωρούμε ότι είναι η αριστερή πλευρά στο πρόγραμμα μας) και στην επιφάνεια στην άλλη άκρη του υλικού (θεωρούμε ότι είναι η δεξιά πλευρά στο πρόγραμμα μας). Επίσης είναι προφανές ότι αν έχουμε 2 στρώματα υλικών η θερμοκρασία στις επιφανές επαφής τους θα είναι ίδια.

Για να αποθηκεύσουμε την θερμοκρασία και στις 2 πλευρές του υλικού θέλουμε 2 μεταβλητές για κάθε στρώμα υλικού που θα έχουμε. Άρα η πιο έξυπνη λύση είναι να χρησιμοποιήσουμε 2 πεδία μέσα στην κλάση layer που φτιάξαμε πριν. Ευτυχώς κατά τον σχεδιασμό της προνοήσαμε είχαμε δει ότι ήταν αναμενόμενο να χρειαστούμε αυτές τις 2 μεταβλητές και τις είχαμε είδη προσθέσει. Αν δεν το είχαμε κάνει, απλά θα προσθέταμε τα πεδία αυτά και τις αντίστοιχες μεθόδους τους τώρα στην κλάση **layer**.

Θέσαμε **temperature** και **temperaturePlus1** την θερμοκρασία στην πλευρά που δέχεται την θερμότητα και την άλλη πλευρά αντίστοιχα

Η θερμοκρασία στο στρώμα στην αρχή του τοιχώματος στην επιφάνια που βρίσκεται σε επαφή με την ατμόσφαιρα δίνεται από τον τύπο:

$$\mathrm{Q} = \mathrm{U} * (\theta_{internal} - \theta_{\alpha}) \;\; \Leftrightarrow \theta_{\alpha} = \theta_{internal} - \frac{Q}{h_i}$$

και η θερμοκρασία στο στρώμα στο τέλος του τοιχώματος στην επιφάνια που βρίσκεται σε επαφή με την ατμόσφαιρα δίνεται από τον τύπο:

$$\theta_{\omega} = \frac{Q}{h_i}$$

Για να υπολογίσουμε τώρα και τις θερμοκρασία στα ενδιάμεσα στρώματα πρέπει να βρούμε την πτώση θερμοκρασίας που προκαλεί, και να την αφαιρέσουμε από την θερμοκρασία που έχει παραμείνει από το προηγούμενο υλικό

Η πτώση θερμοκρασίας που προκαλεί το υλικό δίνεται από τον τύπο:

$$\Delta\theta_{fall} = \frac{d_n}{\lambda_n} * U(\theta_{internal} - \theta_{external})$$

Άρα ας πάμε να κατασκευάσουμε την μέθοδο μας:

```
/**
   * υπολογισμός των επιμέρους θερμοκρασιών
   */
  private void CalculateTemperatures()
  {
    // ipologismos tis thermokrasias stin esoteriki pleura tou tixou
    TemperatureInWall=(TemperatureInternal-(densityOfThermalTransfer/Hi));
    layer.get(0).setTemperature(TemperatureInWall);
    //βάζουμε την θερμοκρασία στην πρώτη επιφάνια του τοίχου
    //σε μια προσωρινή μεταβλητή
    temp=TemperatureInWall;
     /*
     * υπολογισμός των ενδιάμεσων θερμοκρασιών.
     * ο μετρητής ξεκίνα από το 1 για να μπορέσουμε να πάρουμε την
     * προηγούμενη θερμοκρασία
     */
     for (int i=1; i<layer.size(); i++)
     {
       temp=(temp-(layer.get(i-1).
           getWidthdivCTCRatio()*thermalTransmittance*
           (TemperatureInternal-TemperatureExternal)));
     /*
      * εφόσον η θερμοκρασία στις επιφάνειες επαφής 2 στρωμάτων
      * είναι ίδια, μπορούμε με έναν υπολογισμό να θέσουμε και
      * της 2 μεταβλητές ταυτόχρονα
      */
      layer.get(i).setTemperature(temp);
      layer.get(i-1).setTemperaturePlus1(temp);
     }
       //τέλος υπολογισμός στην εξωτερική πλευρά του τοίχου

      layer.get(layer.size()-1).
     setTemperaturePlus1(densityOfThermalTransfer/He);
  }
```

Ας ασχοληθούμε λίγο με τις πιέσεις υδρατμών.

Θέλουμε να υπολογίσουμε τις πιέσεις υδρατμών για κάθε θερμοκρασία που βρήκαμε παραπάνω. Με λίγη ερευνά βρίσκουμε ότι υπάρχει μια εξίσωση η οποία μας δίνει τις πιέσεις κορεσμού υδρατμών για μια γνωστή θερμοκρασία. Ο τύπος είναι[4]:

$$e_s = a_1 + a_2(T - K)^1 + \cdots + a_7(T - K)^6$$

Με:

[4] (*6th order Polynomial (Flatau et. al.,1992) (-50 to 50C)

- a_1=6.11176750,
- a_2=0.443986062,
- a_3=0.143053301E-01,
- a_4=0.265027242E-03,
- a_5=0.302246994E-05,
- a_6=0.203886313E-07,
- a_7=0.638780966E-10,

- και K=273.15 όπου το T πρέπει να δίνεται σε μονάδες Κέλβιν.

Επειδή αυτός ο τύπος θα χρησιμοποιηθεί αρκετές φόρες από το πρόγραμμα μας, και η θέση του θεματικά δεν είναι μέσα στην κλάση **wall**, πρέπει να φτιάξουμε μια κλάση για να τον συμπεριλάβουμε.

5.7.5 Μέθοδοι υπολογισμού

Για να υπολογίσουμε τις πιέσεις τάσης κορεσμένων υδρατμών λοιπόν:

```
for (int i=0; i<layer.size(); i++)
    {

      //τάση κορεσμένων υδρατμών
      layer.get(i).setTasiKoresmenwnIdratmwn
(100*SaturatedWaterPressure.SaturatedWaterPressure
          (TemperatureConvertor.Celsius2Kelvin
          (layer.get(i).getTemperature())));
     }
```

Για να υπολογίσουμε την πίεση θέλουμε την θερμοκρασία του τοιχώματος, αυτό το περνούμε με την κλήση **του getTemperature** του αντικειμένου

<u>Σημείωση:</u> Επειδή η ο τύπος που χρησιμοποιήσαμε πριν που υπολογίζει τις τάσεις υδρατμών θέλει την θερμοκρασία σε μονάδες Κέλβιν, φτιάξαμε και με μέθοδο που κάνει την μετατροπή. Κάλλιστα θα μπορούσαμε να τροποποιήσουμε ελαφρός τον τύπο αλλά είναι καλύτερα τις μαθηματικές εξισώσεις να τις αφήνουμε ανέπαφες. Επίσης ο τύπος μας δίνει την πίεση σε μονάδες στήλης ύδατος. Για να τις μετατρέψουμε σε μονάδες Pascal, πρέπει να πολλαπλασιαστεί με το 100 το αποτέλεσμα.

Για να υπολογίσουμε την πίεση τάσεων στο άλλο άκρο ισχύει ότι και στις θερμοκρασίες, δηλαδή η τιμή της είναι ίδια με την τιμή τις πίεσης τάσεως του επόμενου στρώματος.

Επομένως:

```
for (int i=1; i<layer.size(); i++)
    {
      //tasi koresmenwn idratmwn plus1
      layer.get(i-1).setTasiKoresmenwnIdratmwnPlus1
          (layer.get(i).getTasiKoresmenwnIdratmwn());

    }
```

Φυσικά δεν πρέπει να ξεχάσουμε και την τελευταία τιμή η οποία δεν υπολογίζεται μέσα από τον βρόχο.

```
layer.get(layer.size()-1).setTasiKoresmenwnIdratmwnPlus1
        (100*SaturatedWaterPressure.SaturatedWaterPressure
        (TemperatureConvertor.Celsius2Kelvin
        (layer.get(layer.size()-1).getTemperaturePlus1())));
```

Ας υπολογίσουμε στην ίδια μέθοδο τον συντελεστή διαπερατότητας υδρατμών των στρωμάτων.

Ο συντελεστής διαπερατότητας υδρατμών d/δ δίνεται από τον τύπο

$$\frac{d_j}{\delta_j} = 1.572 * 10^6 * d_j * \mu_j$$

άρα η μέθοδος μας γίνεται:

```
layer.get(i).setAntistasiDiaperatotetitasYdratmwn
(D*layer.get(i).getLayerWidth()*layer.get(i).getWVR());
```

όπου D=1.572*10^6

```
public static final double D=1.572E6;
```

Και ολόκληρη η μέθοδος τελικά είναι:

```
private void CalculateSaturatedWaterPressure()
  {
      for (int i=0; i<layer.size(); i++)
    {
      layer.get(i).setTemperatureFall();
      //τάση κορεσμένων υδρατμών για κάθε στρώμα
      layer.get(i).setTasiKoresmenwnIdratmwn
          (100*SaturatedWaterPressure.SaturatedWaterPressure
          (TemperatureConvertor.Celsius2Kelvin
          (layer.get(i).getTemperature())));

      //αντίσταση διαπερατότητας υδρατμών d/δ
      layer.get(i).setAntistasiDiaperatotetitasYdratmwn
          (D*layer.get(i).getLayerWidth()*layer.get(i).getWVR());
    }
     //τελευταία ακραία τάση κορεσμένων υδρατμών
    layer.get(layer.size()-1).setTasiKoresmenwnIdratmwnPlus1
        (100*SaturatedWaterPressure.SaturatedWaterPressure
        (TemperatureConvertor.Celsius2Kelvin
        (layer.get(layer.size()-1).getTemperaturePlus1())));

    for (int i=1; i<layer.size(); i++)
    {
      //τάση κορεσμένων υδρατμών στο άλλο άκρο των στρωμάτων
      layer.get(i-1).setTasiKoresmenwnIdratmwnPlus1
          (layer.get(i).getTasiKoresmenwnIdratmwn());

    }

  }
```

Το επόμενο βήμα είναι να υπολογίσουμε τις μερικές πιέσεις τάσης. Η πτώση της μερικής πίεσης που προκαλεί κάθε στρώμα δίνεται από τον τύπο:

$$\Delta P_n = \frac{\delta_n}{d_n} * K_D * (Μεγιστη\ Μερικη\ πιεση - Ελαχιστη\ μερικη\ πιεση)$$

Αλλά για να χρησιμοποιήσουμε αυτόν τον τύπο πρέπει πρώτα να υπολογίσουμε τον συντελεστή διαπερατότητας υδρατμών K_D του συνθετου τοιχωματος, ο οποιος όμως δεν είναι τιποτα άλλο από το 1 δια το άθροισμα των συντελεστή διαπερατότητας υδρατμών όλων τον τοιχώματος, πράγμα που έχουμε είδη υπολογίσει:

$$K_D = \frac{1}{\sum \frac{d_j}{\delta_j}}$$

```
Private double KD=0;

private void CalculateKD()
  {
    double temp=0;
    for(int i=0;i<layer.size();i++)
    {
      temp+=layer.get(i).getAntistasiDiaperatotitasYdratmwn();
    }
    KD=1/ temp;
  }
```

Πρέπει ακόμα να φτιάξουμε μια μέθοδο για τον υπολογισμό των μερικών πιέσεων.

Για να υπολογίσουμε τις 2 ακραίες μερικές πίεσης τάσεων των υδρατμών χρησιμοποιούμε τον τύπο

P=$\varphi_{ποσοστο\ υγρασιας}$ *Πιεση υδρατμων για την θερμοκρασια που υπαρχει. Άρα:

```
waterPressureInternal=(100*SaturatedWaterPressure.SaturatedWaterPressure
(TemperatureConvertor.Celsius2Kelvin(TemperatureInternal)))
            *relativeHumidityInternal/100;

waterPressureExternal=(100*SaturatedWaterPressure.SaturatedWaterPressure
(TemperatureConvertor.Celsius2Kelvin(TemperatureExternal)))
            *relativeHumidityExternal/100;
```

```
waterPressureDifference=waterPressureInternal-waterPressureExternal;
```

Ο τύπος υπολογίζει την πτώση τάσης υδρατμών που προκαλεί κάθε στρώμα, δηλαδή όπως κάναμε και στον υπολογισμό της θερμοκρασίας, πρέπει να αφαιρούμε αυτήν την τιμή από την πίεση υδρατμών που έχει το προηγούμενο στρώμα.

```
// υπολογισμός των μερικών πιέσεων τάσεων στης επιφανές επαφής
    //βαλε σε μια μεταβλητή την πίεση στην αριστερή πλευρά
    temp=waterPressureInternal;
    //θέσε την πρώτη μερική τάση υδρατμών ως την πίεση στην αριστερή πλευρά
    layer.get(0).setMerikiTasiYdratmwn(waterPressureInternal);
    //για όλα τα στρώματα
```

```
     for (int i=1; i<layer.size(); i++)
      {
       /*
       *υπολόγισε την μερική πίεση υδρατμών για το τωρινό στρώμα και
       * θέσε το σε μια προσωρινή μεταβλητή
       */
        temp=(temp-(KD*waterPressureDifference*
            layer.get(i-1).getAntistasiDiaperatotitasYdratmwn()));
        //θέσε την μερική πίεση υδρατμών
       στην αριστερή πλευρά του τωρινού στρώματος
        layer.get(i).setMerikiTasiYdratmwn(temp);
        //θέσε την μερική πίεση υδρατμών στην
        δεξιά πλευρά του προηγούμενου στρώματος
        /*
        * εφόσον η δεξιά επιφάνια του προηγούμενου στρώματος, και η αριστερή
        * του τωρινού έχουν επαφή έχουμε την ίδια τιμή
        */
        layer.get(i-1).setMerikiTasiYdratmwnPlus1(temp);

      }
    //υπολόγισε και θέσε Τιεν μερική τάση υδρατμών για το τελευταίο στρώμα
     layer.get(layer.size()-1).setMerikiTasiYdratmwnPlus1(temp-
         (KD*waterPressureDifference*layer.get
         (layer.size()-1).getAntistasiDiaperatotitasYdratmwn()));

  }
```

Χρειαζόμαστε ακόμα μια μέθοδο που υπολογίζει τον πολλαπλασιασμό μεταξύ του συντελεστή διαπίδυσης υδρατμών και το πάχος του κάθε στρώματος και επιστρέφει το άθροισμα του παραπάνω πολλαπλασιασμού για όλα τα στρώματα, για να το χρησιμοποιήσουμε αργότερα για την δημιουργία του διαγράμματος glasser

$$\sum_{1}^{\nu} \mu * D$$

```
     public double calculateMDsummary()
    {
        double temp=0;

        for (int i=0; i<layer.size(); i++)
        {
           temp+=(layer.get(i).getWVR()*layer.get(i).getLayerWidth());
        }
        return temp;
    }
```

Παρομοίως χρειαζόμαστε ακόμα μια μέθοδο που κάνει το ίδιο για τις αντιστάσεις διαπερατότητας υδρατμών για όλα τα τοιχώματα.

```
     public double calculateAntistasiDiaperatotitasIdratmwnSummary()
    {
        double temp=0;

        for (int i=0; i<layer.size(); i++)
        {
           temp+=(layer.get(i).getAntistasiDiaperatotitasYdratmwn());
        }
        return temp;
    }
```

5.7.6 Μέθοδος έναρξης των υπολογισμών

Τέλος σχεδιάζουμε την μέθοδο **calculate** που κάνει όλους τους υπολογισμούς με μια λογική σειρά. Τις μεθόδους που θα χρησιμοποιήσει τις έχουμε είδη γράψει παραπάνω στο ίδιο κεφάλαιο.

Σχόλιο [p4]: Που βρίσκεται αυτή?

Σχόλιο [h5]: kai oi dio einai mesa sto wall.

```
public void Calculate()
    {

        CalculateThermalTransmittance();
        CalculateDensityOfThermalTransfer();
        CalculateTemperatures();
        //CalculateWaterPressures();
        CalculateSaturatedWaterPressure();
        CalculateKD();
        CalculateMerikesPieseisTasewn();
        MyGlasserPanel MyGlasserPanel=new MyGlasserPanel();
    }
```

5.7.7 Κύρια Μέθοδος συγγραφής αναφοράς html.

Σχόλιο [p6]: Που βρίσκεται? Είναι μέσα στο Wall?

Τέλος θα γράψουμε μια μέθοδο που δημιουργεί ένα html αρχείο με μια τελική αναφορά.

```
public void writeReport(String file,boolean enconding)
    {
       // PrintWriter outputStream =null;

        try
            {
            // Create the Buffered Writer object to write to a file called
report.html
              fileOut = new PrintWriter(new FileOutputStream(file));
              //header και μορφοποιηση
              if(enconding)
              fileOut.print(HTMLSTART+NL);
              else
              fileOut.print(HTMLSTART2+NL);
              fileOut.print("<h1>Αρχικά Δεδομένα</h1>");
              fileOut.print(NL);
              fileOut.print(NL);
              fileOut.print("<img src=\"layerImage.jpg\" alt=\"εικονα των
στρωματων\"></img>");
              fileOut.print(NL);
              fileOut.print(NL);
              fileOut.print("Οι θερμόκρασιες που έχεις δώσει ειναι: "+NL
                     +TemperatureInternal+" °C και "+TemperatureExternal+"
°C");
              fileOut.print(NL);
              fileOut.print("Οι σχέτικες υγράσιες που έχεις δώσει ειναι:
"+NL
                     +relativeHumidityInternal+"% και
"+relativeHumidityExternal+"%");
              fileOut.print(NL);
              fileOut.print("Τα στρώματα του τοιχωματός που έχεις δώσει
είναι: ");
              fileOut.print(NL+"<ol>");
              writeLayerData();
              fileOut.print(NL+"</ol>");
              fileOut.print("<h1>Αποτελέσματα</h1>");
               fileOut.print("<h3>Διάγραμμα Glasser</h3>"+NL);
              fileOut.print("<img src=\"glasserImage.jpg\" alt=\"εικονα των
στρωματων\"></img>"+NL);
              fileOut.print("<h3>Συντελεστές</h3>");
```

```
                writeThermalTransmittance();

                writeDensityOfThermalTransfer();
                writeKD();
                fileOut.print("<h3>Θερμοκρασίες</h3>"+NL);
                writeTemperatures();
                fileOut.print("<h3>Πιέσεις</h3>"+NL);
                writeWaterPressures();
                writeMerikesTaseisYdratmwn();
                fileOut.print("<h3>Συγκρίσεις</h3>"+NL+"<ul>");
                writeSigrisi();
                fileOut.print(NL);
                fileOut.print(NL);
                writeElenxosSimpiknosis();
                fileOut.print(NL+"</ul>");
                fileOut.print("<h3>Ολα τα νουμερα των στρωμάτων</h3>");
                fileOut.print(NL+"<ol>");
                writeAllLayerData();
                fileOut.print(NL+"</ol>");
                fileOut.print(NL);
                fileOut.print("<h3>Πινάκας Αποτελεσμάτων</h3>");
                htmlTable();//κατασκευη table
                fileOut.write(HTMLFINISH);
                fileOut.close(); // Close the output stream after all output
is done.
                String path = System.getProperty("user.dir")+"\\";  //βρισκει
το path που τρεχει το προγραμμα
                Runtime.getRuntime().exec("rundll32
SHELL32.DLL,ShellExec_RunDLL " + path+ file);
            }
            catch (FileNotFoundException ioe)
            {
                ioe.printStackTrace();
            }
            catch (IOException ioe)
            {
              ioe.printStackTrace();
            }

    }
```

5.7.8 Βοηθητικές μέθοδοι αναφοράς html

Και τις βοηθητικές μεθόδους που κατασκευάσαμε για να κάνουμε πιο ευανάγνωστη την παραπάνω μέθοδο.

Η **writeThermalTransmittance()** γράφει την αναφορά για τον συντελεστή θερμοπερατότητας στο αρχείο

```
  private void writeThermalTransmittance()
    {
       fileOut.print("Ο συντελεστής θερμοπερατότητας ειναι: "+NL);
       fileOut.print(round(thermalTransmittance)+" W/m²K");
       fileOut.print(NL);
    }
```

Η **writeDensityOfThermalTransfer()** γράφει την αναφορά για την πυκνότητα θερμορροής στο αρχείο.

```
  private void writeDensityOfThermalTransfer()
    {
         fileOut.print("Η πυκνότητα θερμοροιης είναι: "+NL);
```

```
         fileOut.print(round(densityOfThermalTransfer)+" W/m²"+NL);
         fileOut.print(NL);
    }
```

Η **writeTemperatures()** γράφει την αναφορά για τις θερμοκρασίες εσωτερικού και εξωτερικού χώρου στο αρχείο.

```
  private void writeTemperatures()
    {
        fileOut.print("<li>Η θερμοκρασία στην εσωτερική πλευρά του
τοιχώματος"+NL+
                "του εσωτερικού χώρου είναι: ");
        fileOut.print(round(TemperatureInWall)+" °C</li>");
        fileOut.print(NL);
         for (int i=1; i<layer.size(); i++)
        {
           fileOut.println("<li>Η θερμοκρασία μεταξύ "+NL+
                   layer.get(i-1).getLayerName()+" και "+
layer.get(i).getLayerName()+
                   "\n είναι: "+NL
                   +round(layer.get(i).getTemperature())+" °C</li>");
           fileOut.print(NL);
         }
        fileOut.print("<li>Η θερμοκρασία στην εξωτερική πλευρα του τοίχου
είναι: "+NL);
        fileOut.print(round(layer.get(layer.size()-
1).getTemperaturePlus1())+" °C</li>"+NL);
    }
```

Η **writeSigrisi()** γράφει και κάνει την σύγκριση των θερμοκρασιών των επιφανειών του τοιχώματος με τις θερμοκρασίες δρόσου. Για να γίνει αυτό χρειάζεται να σχεδιάσουμε άλλη μια κλάση, την κλάση **DewPoint()** που θα περιλαμβάνει 2 μεθόδους. Περισσότερα στο άλλο κεφάλαιο. Εναλάκτικα

```
private void writeSigrisi()
    {
         fileOut.print("<ul>");
        if(TemperatureInWall>
        DewPoint.CalculateDewPoint(relativeHumidityInternal,
TemperatureInternal))
        {
            fileOut.print("Για οταν η σχετική υγρασια" +
                    " του αέρα ειναι :"+relativeHumidityInternal+ "%"+NL+
                    "η θερμοκρασία στην αριστερή επιφανεια του αριστερού " +
                    "τοίχου είναι" +
                    " μεγαλύτερη από το σημείο δροσου: "+NL+
                    +round(TemperatureInWall)+"
°C"+">"+round(DewPoint.CalculateDewPoint
                    (relativeHumidityInternal, TemperatureInternal))+"
°C"+NL+
                    "και δεν υπάρχει κίνδυνος συμπύκνωσης στην επιφάνεια του
τοίχου"+NL);
        }
        else
            fileOut.print("<li>Για οταν η σχετική υγρασια" +
                    " του αέρα ειναι :"+relativeHumidityInternal+ "%"+NL+
                    "η θερμοκρασία στην αριστερή επιφανεια του αριστερού " +
```

```
                    "τοίχου είναι" +
                    " μικρότερη από το σημείο δροσου: "+NL+
                    round(TemperatureInWall)+"
°C"+"<"+round(DewPoint.CalculateDewPoint
                    (relativeHumidityInternal, TemperatureInternal))+"
°C"+NL+
                    "και <p><b>υπάρχει κίνδυνος συμπύκνωσης</p></b> στην
επιφάνεια του τοίχου</li>"+NL+NL+
                    "<li>Πρεπει η προσθεθει μόνωση στον τοίχο, ετσι ώστε:
"+NL+"</li>"+
                    "<li>Η θερμοκρασια δροσου να ειναι μικροτερη απο την
θερμοκρασια του τοιχοματος"+NL+
                    "Για να γινει αυτο πρεπει η πυκνότητα θερμορροης να
ειναι το πολυ: "+NL+
                    round((Hi*(TemperatureInternal-
DewPoint.CalculateDewPoint
                    (relativeHumidityInternal, TemperatureInternal))))+
                    " W/m²"+NL+
                    "Και ο συντελεστης θερμοπερατότητας να ειναι :"+NL+
                    round(((Hi*(TemperatureInternal-
DewPoint.CalculateDewPoint
                    (relativeHumidityInternal, TemperatureInternal)))
                    /(TemperatureInternal-TemperatureExternal))) +"
W/m²K</li>" +NL);

         fileOut.print("</ul>");

    }
```

Η **writeWaterPressures()** γράφει τις μερικές πιέσεις υδρατμών στο αρχείο.

```
  private void writeWaterPressures()
    {
         fileOut.print("<ul>");
        fileOut.print("<li> Οι μερικες πιεσεις υδρατμων στα 2 ακρα των
τοιχομάτων είναι :");
        fileOut.print(NL);
        fileOut.print(" στην αριστέρη πλευρά:
"+round(waterPressureInternal)+" Pa");
        fileOut.print(NL);
        fileOut.print(" στην δεξία πλευρά: "+round(waterPressureExternal)+"
Pa</li>");
        fileOut.print(NL);
        fileOut.print(" <li>Η συνολικη πτωση τασης μεταξυ των στρωματον
ειναι :"+NL+
                round((waterPressureInternal-
waterPressureExternal))+"</li>");
        fileOut.print("</ul>");
    }
```

Η **writeKD()** γράφει τον συντελεστή διαπερατότητας υδρατμών του τοιχώματος στο αρχείο

```
      private void writeKD()
    {
        fileOut.print(" Ο συντελεστης διαπερατότητας υδρατμών του τοιχόματος
είναι :" +NL
               +round(KD)+" kg/(m²hPa"+NL);
    }
```

Η **writeMerikesTaseisYdratmwn()** γράφει τις μερικές τάσεις υδρατμών στο αρχείο

```
    private void writeMerikesTaseisYdratmwn()
        {
        fileOut.print("<ul>");
             fileOut.print(NL);
             fileOut.print("Οι επιμέρους τασείς είναι :"+NL);
             for (int i=1; i<layer.size(); i++)
            {
               fileOut.println("<li>Η τασής υδρατμών μεταξύ του "+NL+
                             layer.get(i-1).getLayerName()+" και του "+
layer.get(i).getLayerName()+NL+
                             "\n είναι: "

+round((layer.get(i).getMerikiTasiYdratmwn()))+"Pa</li>");
                fileOut.print(NL);
             }
             fileOut.print("</ul>");
        }
```

5.8 Η κλάση GUI (Graphical User Interface)

Τα περισσότερα σύγχρονα προγράμματα προσφέρουν στους χρήστες τους κάποιο GUI (γραφικό περιβάλλον χρήστη) γιατί αυτός ο τρόπος αλληλεπίδρασης με τον υπολογιστή ταιριάζει αρκετά στην ανθρώπινη εμπειρία και φύση. Σωστά σχεδιασμένα γραφικά προσφέρουν ένα όμορφο, εύχρηστο και λειτουργικό περιβάλλον εργασία. Το πρόγραμμα μας θα πρέπει να έχει φυσικά, και αυτή η κλάση θα περιλαμβάνει τον κώδικα του γραφικού περιβάλλοντος, καθώς και τον κώδικα που θα ξεκινάει τους γενικούς υπολογισμούς που θέλουμε. Εμείς σε αυτό το κεφάλαιο θα επικεντρωθούμε σε αυτό το κομμάτι, γιατί ο προγραμματισμός ενός γραφικού περιβάλλοντος, είναι πέρα από το πεδίο αυτής της πτυχιακής.

5.8.1 Κάρτα CRC

GUI	
Προβολή Γραφικής παράστασης	glasser graphics
Εισαγωγή δεδομένων	wall,layer, data
Έναρξη υπολογισμών	wall
Προβολή γραφικής αναπαράστασης τοιχώματος	layer graphics
Προβολή αναφοράς	wall

5.8.2 Ιδιότητες

Το γραφικό περιβάλλον θα περιλαμβάνει 2 λίστες. Η μία λίστα θα περιλαμβάνει όλα τα πιθανά υλικά στρωμάτων τα που μπορεί να διαλέξει ο χρήστης και η άλλη λίστα θα γεμίζει με τα υλικά των στρωμάτων που θα απαρτίζει το τοίχωμα που θέλει να μελετήσει ο χρήστης.

Αρά με την εκκίνηση του προγράμματος θα πρέπει με κάποιον τρόπο να γεμίζει η πρώτη λίστα

```
   public void initComponents2(){
        listModel = new DefaultListModel();
        targetListModel=new DefaultListModel();

        //αναζητηση των ονοματων και αντιγραφη τους σε ενα array
        array=Data.FindElementsForList();
        //προσθηκη του Διακενου μια και δεν υπαρχει στο αρχειο
        listModel.addElement("diakeno");
        //γεμισμα των στοιχειων
        for(int i=0; i<array.size(); i++)
            listModel.addElement(array.get(i).toString());
   }
```

Την **FindElementsForList()**, την είχαμε σχεδιάσει στο κεφαλαίο 5, στην κλάση **LayerData()** για αυτόν ακριβώς τον λόγο.

Εκτός από τις λίστες, το πρόγραμμα θα πρέπει να έχει 4 πεδία που πρέπει να γεμίσει ο χρήστης.

- θ_1 θερμοκρασία στην μία πλευρά
- θ_2 θερμοκρασία στην άλλη πλευρά
- Σχετική υγρασία στην μια πλευρά
- Σχετική υγρασία στην άλλη πλευρά

Με αντίστοιχα ονόματα μεταβλητών:

- **jFieldTemperatureInternal**
- **jFieldTemperatureExternal**
- **jFieldHumidityExternal**
- **jFieldHumidityInternal**

Επειδή ο καλύτερος τρόπος να περιοριστούν τα προβλήματα σε ένα πρόγραμμα είναι να προσέχουμε τι δεδομένα τους δίνουμε εξαρχής, κατασκευάζουμε μια μέθοδο που περιορίζει τις εισόδους που δέχεται το πρόγραμμα στις μορφή που τις χρειαζόμαστε, δηλαδή σε πραγματικούς αριθμούς.

```
private void LimitInputToPositiveRealNumbers(KeyEvent evt,JTextField field)
   {
        String text=field.getText();
```

```
         System.out.println(text);

         String badchars = "`~!@#$%^&*()_+=\\|\"':;?/>-<, ";
            char c = evt.getKeyChar();
            if((Character.isLetter(c) && !evt.isAltDown()) ||
badchars.indexOf(c) > -1)
            {
                evt.consume();
                Toolkit.getDefaultToolkit().beep();
                return;

            }
            if((c == '-') && (text.length() > 0))
            {
                evt.consume();
                System.out.println(2);
                Toolkit.getDefaultToolkit().beep();
                return;
            }

            if((c=='.')&&(text.indexOf('.')!=-1))
            {
                System.out.println("yparxei telia");
                evt.consume();
                Toolkit.getDefaultToolkit().beep();
                return;
            }
            else
                super.processKeyEvent(evt);
   }
```

Και άλλη μια μέθοδο (που αφορά την φόρμα) που δεν αφήνει τους υπολογισμούς να ξεκινήσουν εκτός αν είναι όλα τα δεδομένα στην θέση τους, δηλαδή, τα 4 πεδία που αναφέρθηκαν παραπάνω, καθώς και τουλάχιστον ένα στρώμα υλικού.

```
    public void CalculateButtonChecker()
    {
         if((jFieldTemperatureInternal.getText().isEmpty())||
                   (jFieldTemperatureExternal.getText().isEmpty())||
                   (jFieldHumidityExternal.getText().isEmpty())||
                   (jFieldHumidityInternal.getText().isEmpty())||
                   (targetListModel.isEmpty()))
         {

            calculateButton.setEnabled(false);
            saveButton.setEnabled(false);
         }
        else
        {
            calculateButton.setEnabled(true);
            saveButton.setEnabled(true);
        }

    }
```

5.8.3 Λειτουργικότητα Κουμπιών add, remove, calculate

Θα πρέπει ακόμα να υπάρχουν δύο κουμπιά που θα προσθέτουν ή θα αφαιρούν αντίστοιχα, στρώματα υλικού στην δεύτερη λίστα.

Το κουμπί που θα προσθέτει ονομάζεται **addLayerButton**. Όταν πατηθεί αυτό το κουμπί θα γίνεται ένας έλεγχος για το αν ο χρήστης έχει βάλει τιμή για το πάχος του στρώματος που θέλει να προσθέσει και αν αυτή ξεπερνάει μια μεγίστη τιμή που έχουμε ορίσει.

```
private void addLayerButtonActionPerformed(java.awt.event.ActionEvent evt) {
```

```
   if ((widthField.getText().equals(""))) {
        Toolkit.getDefaultToolkit().beep();
        JOptionPane.showMessageDialog(null,
                "Βάλε το πλάτος του στρώματος που θες να προσθέσεις πρώτα",
                "Error", JOptionPane.ERROR_MESSAGE);
        return;
    } else if((MyPanel.getHorizontalWidthSummary()>=PIXEL_TOTAL)
            ||((MyPanel.getHorizontalWidthSummary())+

(PIXEL_TOTAL*(Double.parseDouble(widthField.getText())*widthMultiplyer)))>PI
XEL_TOTAL)) {
        JOptionPane.showMessageDialog(null, "Το συνολικό πλάτος του
τοιχώματος" +
                "δεν πρεπει να υπερβαίνει το 1 μετρο", "Error",
                JOptionPane.ERROR_MESSAGE);
        return;
    }
```

Αν οι επιλογές είναι έγκυρες και περάσει η επιλογή μας τον παραπάνω έλεγχο, το επιλεγμένο υλικό από την πρώτη λίστα, θα καταχωρείτε στην δεύτερη λίστα και θα δημιουργείτε ένα αντίστοιχο αντικείμενο layer.

```
  // περνει τον δεικτη της επιλεγμενης τιμης του targetList
    // αν δεν ειναι τιποτα επιλεγμενο επιστρεφει -1
    int index = targetList.getSelectedIndex();
    int size = targetListModel.getSize();
    double width=Double.parseDouble(widthField.getText())*widthMultiplyer;

    // καταχωρηση της επιλεγμενης τιμης της κυριας λιστας σε μεταβλητη
    layerName=sourseList.getSelectedValue().toString();

            /*Αν δεν ειναι τιποτα επιλεγμενο η το αντικειμενο στην τελευταια
             * θεση ειναι επιλεγμενο τοτε προσθεσε το καιρνουργιο
αντικειμενο
             * στο τελος της λιστας και επιλεξε το
             */
    if (index == -1 || (index+1 == size)) {

        targetListModel.addElement(layerName+". Width: "
                +widthField.getText()+" "
                +(String) jCompoMeterSelector.getSelectedItem());
        targetList.setSelectedIndex(size);
        process.ConstructLayer(layerName,width);

           /*αλλιως προσθεσε το νεο αντικειμενο μετα την τωρινη επιλογη
             * και επελεξε το
             */
    else {
        targetListModel.insertElementAt(layerName+". Width: "
                +widthField.getText()+" "
                +(String) jCompoMeterSelector.getSelectedItem(), index+1);
        targetList.setSelectedIndex(index+1);}
        process.ConstructLayer(layerName,width,index+1);
```

Αντίστοιχα θα γίνεται για το κουμπί **Remove**. Θα αφαιρεί δηλαδή μια τιμή από την δεύτερη λίστα, και θα καταστρέφει το αντίστοιχο **Layer** αντικείμενο που είχε δημιουργήσει αυτή η τιμή.

Τέλος υπάρχει το κουμπί calculate που ξεκινάει την διαδικασία με τους υπολογισμούς και κάνει τα ακόλουθα:

- Περνάει τα στοιχεία μέσα στις μεταβλητές της κλάσης wall, που έχουμε κατασκευάσει προηγουμένως.
- Ελέγχει αν οι θερμοκρασίες που έχει δώσει ο χρήστης είναι λογικές τιμές
- Κάνει τους υπολογισμούς.
- Φτιάχνει το διάγραμμα glaser
- Γράφει την τελική αναφορά

```
private void calculateButtonActionPerformed(java.awt.event.ActionEvent evt)
{
//παρε τα στοιχεια που εχει θεσει ο χρηστης
        process.setTemperatureInternal
                (Double.parseDouble(jFieldTemperatureInternal.getText()));
        process.setTemperatureExternal
                (Double.parseDouble(jFieldTemperatureExternal.getText()));
        process.setRelativeHumidityExternal
                (Double.parseDouble(jFieldHumidityExternal.getText()));
        process.setRelativeHumidityInternal
                (Double.parseDouble(jFieldHumidityInternal.getText()));
if(Double.parseDouble(jFieldTemperatureInternal.getText())
                <Double.parseDouble(jFieldTemperatureExternal.getText()))
            JOptionPane.showMessageDialog(null,
                    "Η Θ1 πρεπει να ειναι μεγαλύτερη απο την Θ2", "Error",
                    JOptionPane.ERROR_MESSAGE);
        //ελενχος αν η σχετικη υγρασια εχει αποδεκτο νουμερο(οχι μεγαλυτερο
απο 100%)
        else
if(((Double.parseDouble(jFieldHumidityExternal.getText()))>100)||
                ((Double.parseDouble(jFieldHumidityExternal.getText()))<0)||

((Double.parseDouble(jFieldHumidityInternal.getText()))>100)||
                ((Double.parseDouble(jFieldHumidityInternal.getText()))<0))
            JOptionPane.showMessageDialog(null,
                    "Μη έγκυρη τιμή υγρασιάς",
                    "Error", JOptionPane.ERROR_MESSAGE);
process.Calculate();
makeGlasser();  // κατασκευή του διαγράμματος glasser
process.writeReport("report.html",encoding);
```

5.9 Η κλάση dewPoint

DewPoint	
Υπολογισμός σημείου δρόσου	
Υπολογισμός σχετικής υγρασίας	

5.9.1 Μέθοδοι

Η κλάση **DewPoint()** θα περιλαμβάνει 2 μεθόδους.

Η μια μέθοδος θα δέχεται σαν είσοδο την σχετική υγρασία του αέρα, και την θερμοκρασία του και θα επιστρέφει το σημείο δρόσου.

Ο τύπος που μας δίνει το σημείο δρόσου είναι

$$T_d = \frac{b * (\frac{(a * T)}{b + T} + \ln(\frac{Rh}{100}))}{a - (\frac{(a * T)}{b + T} + \ln(\frac{Rh}{100}))}$$

όπου :

- T = θερμοκρασία.
- RH= σχετική υγρασία.
- Td= θερμοκρασία δρόσου.
- α=17.271
- b=237.7 °C

```
private static double a=17.271;
    private static double b=237.7;
    private static double H;
    private static double T;
    private static final double E=2.718281828459045;
    public static double CalculateDewPoint(double humidity,double
airTemperature)
    {
        double dewPoint=0;
        H=humidity;
        T=airTemperature;
        dewPoint=(b*g(H,T))/(a-g(H,T));
        return dewPoint;
    }

    private static double g(double H, double T)
    {
        double g;
        g=((a*T)/(b+T))+Math.log(H/100);
        return g;
    }
```

Η δεύτερη μέθοδος δέχεται είσοδο θερμοκρασία αέρα και θερμοκρασία δρόσου, και βγάζει αποτέλεσμα την σχετική υγρασία.

$$T_d = \frac{b*G}{a-G}\,\mu\varepsilon \quad G=(\frac{(a*T)}{b+T} + \ln(\frac{Rh}{100}))$$

Λύνουμε ως προς Rh

$$T_d = \frac{(b*G)}{a-G} \Rightarrow T_d*(a-G) = b*G \Rightarrow \frac{a-G}{G} = \frac{b}{T_d} \Rightarrow \frac{a}{G} = \frac{b}{T_d} + 1 \Rightarrow G = \frac{a}{\left(\frac{b}{T_d}\right)+1} \overset{G}{\Rightarrow}$$

$$\ln\left(\frac{Rh}{100}\right) = \frac{a}{\left(\frac{b}{T_d}\right)+1} - \frac{a*T}{b+T} \Rightarrow Rh = e^{\left(\frac{a}{\left(\frac{b}{T_d}\right)+1} - \frac{(a*T)}{b+T}\right)} * 100$$

Και η μέθοδος που προκύπτει είναι :

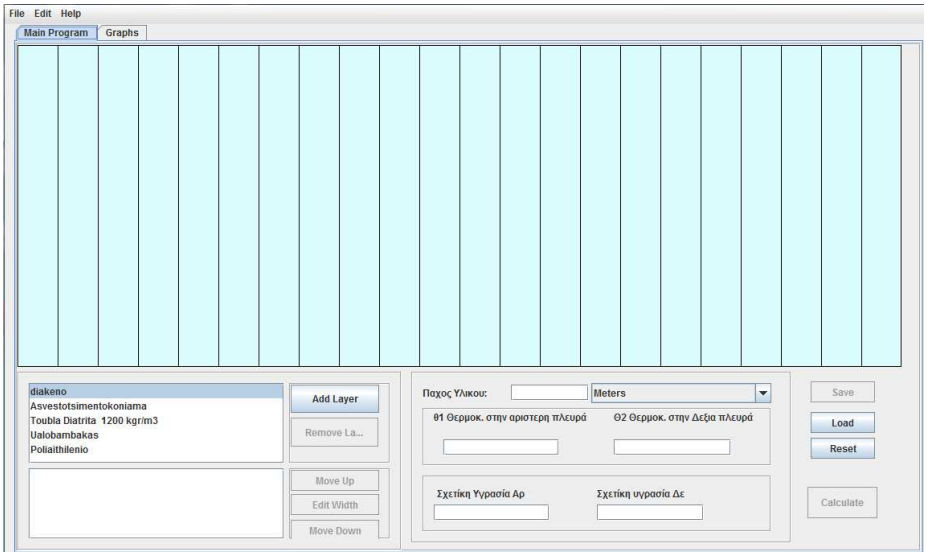
```
public static double CalculateRelativeHumidity(double dewPoint, double
airTemperature)
    {
        H=0;
        T=airTemperature;

        H=Math.pow(E,(((a/((b/dewPoint)+1)))-((a*T)/(b+T))))*100;

        return H;}
```

6 Εγχειρίδιο χρήσης του προγράμματος

Σε αυτό το κεφαλαίο γίνεται επεξήγηση των λειτουργιών του προγράμματος.

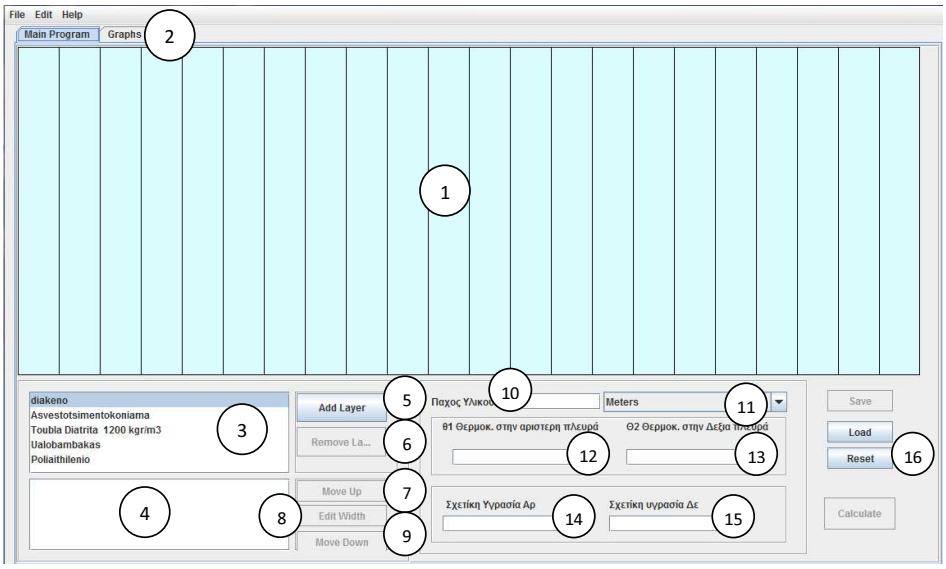


Η βασική αρχή του προγράμματος είναι απλή. Προσθέτουμε ένα-ένα με την σειρά τα στρώματα που απαρτίζουν ένα τοίχωμα, προσθέτουμε την θερμοκρασία του αέρα και την σχετική υγρασία που υπάρχουν από την μια και την άλλη πλευρά του τοιχώματος μας, και το πρόγραμμα κάνει τους υπολογισμούς του και μας προβάλει μια λεπτομερή αναφορά.

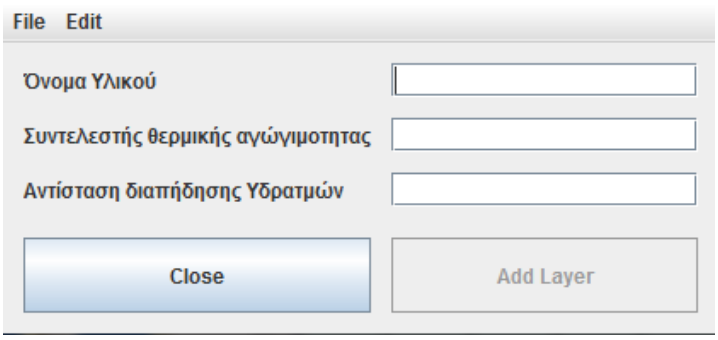
Το πρόγραμμα υπολογίζει:

- Την θερμοκρασία στην επιφάνεια του κάθε στρώματος και από τις 2 πλευρές του.
- Την πτώση θερμοκρασίας που προκαλεί κάθε στρώμα
- Την αντίσταση διαπερατότητας υδρατμών
- Την τάση κορεσμένων υδρατμών στην επιφάνεια του κάθε στρώματος και για τις 2 πλευρές του.
- Την μερική τάση υδρατμών στην επιφάνεια του κάθε στρώματος και για τις 2 πλευρές του
- Τον συντελεστή θερμοπερατότητας του τοιχώματος που έχουμε εισάγει
- Την πυκνότητα θερμορροής του συστήματος
- Σχεδιάζει το διάγραμμα Glaser του συστήματος
- Προειδοποιεί τα σημεία του τοιχώματος στα όποια θα έχουμε συμπύκνωση υδρατμών, και αν το συμπύκνωμα αυτό εξατμίζεται τους θερινούς μήνες

6.1 Layout του προγράμματος

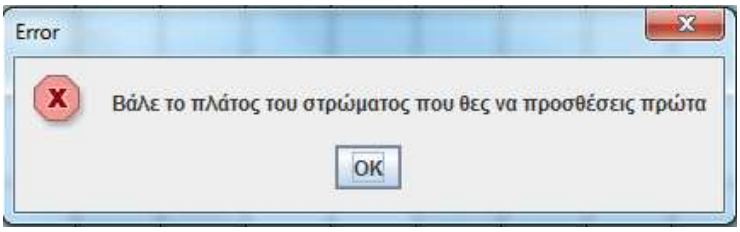


1. Είναι το περιβάλλον που εμφανίζονται τα στρώματα των υλικών που προσθέτουμε Το κάθε στρώμα προβάλλεται σαν ένα παραλληλόγραμμο διαφορετικού χρώματος και μεγέθους (σχετικό με το πάχος του στρώματος). Η αριστερή πλευρά υποδηλώνει την θερμή πλευρά, η δεξιά την ψυχρή, και το grid υποδηλώνει το τοίχωμα.
2. Πατώντας την καρτέλα graph, περνάμε στην καρτέλα που προβάλει το διάγραμμα Glaser
3. Είναι η λίστα με τους τύπους των υλικών που περιλαμβάνει το πρόγραμμα. Επιλέγοντας ένα από αυτά και πατώντας το κουμπί 5, προσθέτετε στο 1 και στο 4.
   Αν χρειαστεί ένα υλικό το οποίο δεν περιλαμβάνεται στον πινάκα, μπορεί να προσδεθεί εύκολα. Πατώντας στο «Edit/Add custom layer» ανοίγει μια νέα φόρμα στην οποίο τοποθετούνται τα χαρακτηρίστηκα του προς πρόσθεση υλικού

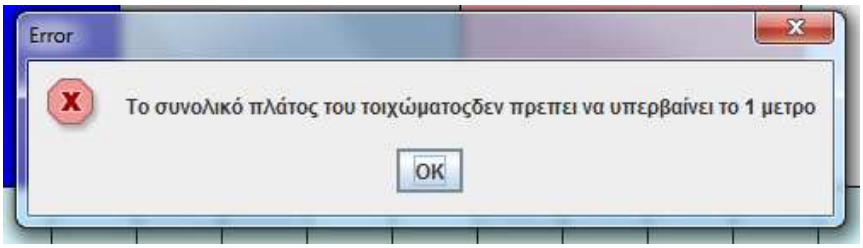


   Μόλις τοποθετηθούν, με το πλήκτρο Add Layer, το υλικό προσθέτετε στην λίστα.
4. Είναι η λίστα με τους τύπους των υλικών που έχει το τοίχωμα μας.

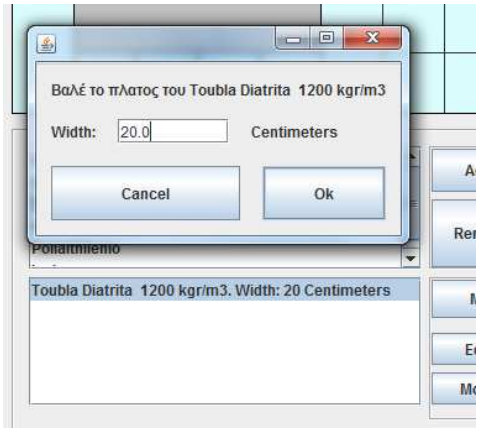
5. Είναι το πλήκτρο που επιτρέπει την προσθήκη ενός νέου στρώματος στην λίστα 4. Για να προσθέσουμε ένα στρώμα πρέπει πρώτα να έχουμε βάλει ένα επιτρεπτό πάχος στρώματος στο πεδίο 10, εναλλακτικά το πρόγραμμα θα προβάλει μήνυμα σφάλματος



Επίσης το συνολικό πλάτος του τοιχώματος δεν μπορεί να υπερβαίνει το 1 μετρό. Αν το πάχος του στρώματος που θέλουμε να προσθέσουμε είναι μεγαλύτερο από ένα μετρό, η το άθροισμα όλων των προηγουμένων στρωμάτων συν αυτό που προσπαθούμε να προσθέσουμε υπερβαίνει το ένα μετρό, το πρόγραμμα προβάλει πάλι μήνυμα λάθους.



6. Επιλέγοντας ένα στρώμα υλικού στην λίστα 4 και πατώντας αυτό το πλήκτρο, αφαιρει το επιλεγμένο στρώμα από την λίστα. Αν δεν έχουμε επιλέξει εμείς κάποιο αφαιρει το τελευταίο που προστέθηκε.
7. Επιλέγοντας ένα στρώμα υλικού στην λίστα 4 και πατώντας αυτό το πλήκτρο το ανεβάζει μια θέση πάνω στην κατάταξη (προς την θερμή πλευρά). Φυσικά η αλλαγή φαίνεται αμέσως στην γραφική εξομοίωση του τοιχώματος.
8. Επιλέγοντας ένα στρώμα υλικού στην λίστα 4 και πατώντας αυτό το πλήκτρο ανοίξει μια καινούργια φόρμα όπου μπορούμε να αλλάξουμε τα στοιχεία του επιλεγμένου στρώματος.



9. Το ίδιο με το πλήκτρο 7, με την διάφορα ότι αυτό κατεβάζει μια θέση στην κατάταξη.
10. Σε αυτό το πεδίο τοποθετείτε το πάχος ενός στρώματος που θα προσδεθεί αν πατηθεί το πλήκτρο 4
11. Επιλεγεί την μονάδα που θα είναι το πάχος του στρώματος. Αν το πεδίο 10 είναι ειδή γεμάτο και αλλάξουμε την μονάδα, η μετατροπή γίνεται αυτόματα.
12. Είναι το πεδίο που δέχεται την θερμοκρασία στην θερμή πλευρά
13. Είναι το πεδίο που δέχεται την θερμοκρασία στην ψυχρή πλευρά Για να βγάλει το πρόγραμμα σωστά αποτελέσματα πρέπει πάντα το πεδίο 12 να έχει μεγαλύτερη θερμοκρασία.
14. Είναι το πεδίο που δέχεται την σχετική υγρασία του αέρα της θερμής πλευράς
15. Είναι το πεδίο που δέχεται την σχετική υγρασία του αέρα της ψυχρής πλευράς
16. Διαγράφει όλα τα δεδομένα, και επιστρέφει το πρόγραμμα στην αρχική του κατάσταση.

6.2 Αποτελέσματα.

Έχοντας βάλει όλα τα δεδομένα και πατώντας το κουμπί calculate, υπολογίζονται τα αποτελέσματα και ανοίγει το φύλλο αποτελεσμάτων στον προεπιλεγμένο φυλλομετρητή του συστήματος.

- Ένα τυπικό html report

Αρχικά δεδομένα

Οι θερμόκρασιες που έχεις δώσει ειναι:
20.0 °C και 0.0 °C
Οι σχέτικες υγράσιες που έχεις δώσει ειναι:
50.0% και 80.0%
Τα στρώματα του τοιχωματός που έχεις δώσει είναι:

1. Όνομα Στρώσης: Asvestotsimentokoniama
   Πάχος Στρώσης: 0.02 m
   Συντελεστης θερμικής Αγωγιμότητας: 0.87 W/mK
   Συντελεστής αντίστασης Διαπύδισης Υδρατμών: 15.0

2. Όνομα Στρώσης: Toubla Diatrita 1200 kgr/m3
   Πάχος Στρώσης: 0.09 m
   Συντελεστης θερμικής Αγωγιμότητας: 0.52 W/mK
   Συντελεστής αντίστασης Διαπύδισης Υδρατμών: 10.0

Αποτελέσματα

Συντελεστές
Ο συντελεστής θερμοπερατότητας ειναι:
2.73 W/m²K
Η πυκνότητα θερμοροιης είναι:
54.65 W/m²

Ο συντελεστης διαπερατότητας υδρατμών του τοιχόματος είναι :
5.30E-7 kg/(m²hPa
Θερμοκρασίες

Η θερμοκρασία στην εσωτερική πλευρά του τοιχώματος
του εσωτερικού χώρου είναι: 12.90 °C

Η θερμοκρασία μεταξύ

Asvestotsimentokoniama και Toubla Diatrita 1200 kgr/m3 είναι:
11.64 °C

Η θερμοκρασία στην εξωτερική πλευρα του τοίχου είναι:
2.18 °C

Πιέσεις

* Οι μερικες πιεσεις υδρατμων στα 2 ακρα των τοιχομάτων είναι :
  στην αριστέρη πλευρά: 1169.33 Pa
  στην δεξία πλευρά: 488.94 Pa

* Η συνολικη πτωση τασης μεταξυ των στρωματον ειναι :
  680.39

  Οι επιμέρους τασείς είναι :
* Η τασής υδρατμών μεταξύ του
  Asvestotsimentokoniama και του Toubla Diatrita 1200 kgr/m3
  είναι: 999.23Pa

Συγκρίσεις

Για οταν η σχετική υγρασια του αέρα ειναι :50.0%
η θερμοκρασία στην αριστερή επιφανεια του αριστερού τοίχου είναι
μεγαλύτερη από το σημείο δροσου:
12.90 °C>9.25 °C
και δεν υπάρχει κίνδυνος συμπύκνωσης στην επιφάνεια του τοίχου

Για να φτασει το σημείο δροσου την θερμοκρασια της
επιφανειας τοιχου θα πρεπει η σχετικη υγρασια του
αέρα να γίνει :63.68%

Δεν δημιουργειται πουθενα συμπυκνωση Υδρατμων

Ολα τα νουμερα των στρωμάτων

1. Όνομα Στρώσης: Asvestotsimentokoniama
   Πάχος Στρώσης: 0.02 m
   Συντελεστης θερμικής Αγωγιμότητας: 0.87 W/mK
   Διαιρεση d/λ: 0.0229 m²K/W
   Συντελεστής αντίστασης Διαπύδισης Υδρατμών: 15.0
   Θερμόκρασια στο αριστερο ακρο: 12.90 °C
   Πτωσή θερμοκρασίας : 1.25 °C
   Θερμόκρασια στο δεξι ακρο: 11.64 °C
   Αντιστασή διαπερατότητας Υδρατμών: 471600.0 d/δ
   Τάση κορεσμένων Υδρατμών στο αριστερο ακρο: 1488.19 Pa
   Τάση κορεσμένων Υδρατμών στο δεξι ακρο: 1370.10 Pa
   Μερική τάση Υδρατμών αριστερο ακρο: 1169.33 Pa
   Πτώση τάσης Υδρατμών: 170.09 Pa
   Μερική τάση Υδρατμών στο δεξι ακρο: 999.23 Pa

2. Όνομα Στρώσης: Toubla Diatrita 1200 kgr/m3
   Πάχος Στρώσης: 0.09 m
   Συντελεστης θερμικής Αγωγιμότητας: 0.52 W/mK
   Διαιρεση d/λ: 0.17307692307692307 m²K/W
   Συντελεστής αντίστασης Διαπύδισης Υδρατμών: 10.0
   Θερμόκρασια στο αριστερο ακρο: 11.64 °C
   Πτωσή θερμοκρασίας : 9.45 °C
   Θερμόκρασια στο δεξι ακρο: 2.18 °C
   Αντιστασή διαπερατότητας Υδρατμών: 1414800.0 d/δ
   Τάση κορεσμένων Υδρατμών στο αριστερο ακρο: 1370.10 Pa
   Τάση κορεσμένων Υδρατμών στο δεξι ακρο: 715.36 Pa
   Μερική τάση Υδρατμών αριστερο ακρο: 999.23 Pa
   Πτώση τάσης Υδρατμών: 510.2976646778401 Pa
   Μερική τάση Υδρατμών στο δεξι ακρο: 488.94 Pa

7 Παράρτημα

7.1 Xml

7.1.1 γενικά

Σε αυτό το κεφάλαιο θα δούμε πως χρησιμοποιείται ο τύπος αρχείων xml (Extensible Markup Language (XML)).

Τα xml είναι ένας μηχανισμός ανεξάρτητος από την γλώσσα προγραμματισμού που χρησιμοποιούμε. Τα xml επιτρέπουν την κωδικοποίηση πολύπλοκων μορφών δεδομένων που το πρόγραμμα μπορεί εύκολα να προσπελάσει. τα XML είναι πολύ δημοφιλής τρόπος για ανταλλαγή δεδομένων. Η δομή του είναι αρκετά εύκολη και υπάρχουν αρκετά προγράμματα που μπορούν να χρησιμοποιηθούν για να δημιουργήσουν τέτοια αρχεία. τα XML χρησιμοποιούν δομή φωλιάς και έτσι μπορούν να χρησιμοποιηθούν για να περιγράψουν ιεραρχικές δομές δεδομένων. Επειδή το φόρμα των xml είναι στάνταρ και διαδεδομένο, βιβλιοθήκες για την προσπέλαση των δεδομένων που περιέχουν είναι διαθέσιμες σε κάθε μοντέρνα γλώσσα προγραμματισμού, και επίσης αρκετά εύκολο να προγραμματιστούν.

Είναι ιδιαίτερα εύκολα να διαβάσεις και να γράψεις xml αρχεία στην JAVA. Για την ακρίβεια είναι πιο εύκολο να χρησιμοποιηθούν από την κλασική μορφή του απλού έγγραφου κειμένου και κάνει το πρόγραμμα πιο λειτουργικό και πιο ανεκτικό στις αλλαγές.

Στο πρόγραμμα τις υγρομόνωσης που εξετάζουμε χρησιμοποιούνται και οι 2 μέθοδοι για κωδικοποίηση δεδομένων, xml στην αποθήκευση και txt για την προσπέλαση των δεδομένων των layer. Αυτό έγινε για να έχουμε μια πιο σφαιρική άποψη των πραγμάτων

Για να δούμε τα πλεονεκτήματα ας δούμε αυτό το παράδειγμα:

Έστω ότι θέλουμε να μεταφέρουμε δεδομένα μεταξύ προγραμμάτων

Η πρώτη απόπειρα είναι αυτή:

```
Toaster
29.95
```

εν αντίθεση να το xml encoding των ίδιων δεδομένων

```
<product>
  <description>Toaster</description>
  <price>29.95</price>
</product>
```

Το πρώτο πλεονέκτημα είναι προφανές: μπορούμε με μια ματιά να καταλάβουμε τι αντιπροσωπεύει το κάθε στοιχειό.

Το άλλο πλεονέκτημα είναι ότι αντιστέκεται στην αλλαγή. Έστω ότι θέλαμε να προσθέσουμε ένα ακόμα στοιχείο

```
Toaster
29.95
General Appliances
```

Το πρόγραμμα που αναλύει την πρώτη δομή κατά μεγάλη πιθανότητα θα «κρασαρε». Στην καλύτερη περίπτωση απλά θα έβγαζε παράξενα αποτελέσματα γιατί αλλά δεδομένα θα μπούνε σε μεταβλητές που δεν πρέπει.

Εναλλακτικά όταν χρησιμοποιούμε xml είναι εύκολο να προσθέσουμε ένα νέο στοιχειό:

```
<product>
  <description>Toaster</description>
  <price>29.95</price>
  <manufacturer>General Appliances</manufacturer>
</product>
```

τώρα το πρόγραμμα που αναλύει το xml μπορεί ακόμα να εξάγει τις παλιές πληροφορίες με τον ίδιο τρόπο που το έκανε και πριν.

Το πρόγραμμα δεν χρειάζεται να ανανεωθεί και είναι ανεκτικό σε διαφορές εκδόσεις του φόρμα των δεδομένων.

Αν τώρα θέλουμε να εξάγουμε τα δεδομένα το μόνο που χρειάζεται (στο σημείο του προγράμματος που διαβάζει τα δεδομένα τουλάχιστον), είναι η προσθήκη μιας ακόμα γραμμής κώδικα για να αναγνωρίζει το νέο στοιχειό.

7.1.2 Η δομή ενός αρχείου xml.

Σε αυτόν τον τομέα θα δούμε τους κανόνες που διέπει η δομή ενός σωστά δομημένου xml.

- Το στάνταρ του xml λέει ότι κάθε xml αρχείο πρέπει να ξεκινάει με την δήλωση:

```
<?xml version=«1.0»?>
```

- Μετά πρέπει να περιλαμβάνεται μια δήλωση από τα στοιχεία που περιέχονται:

```
<!ELEMENT τηλέφωνο (#PCDATA)>
```

- αν το στοιχείο περιλαμβάνει και αλλά στοιχεία μέσα πρέπει να δηλωθεί έτσι:

```
<!ELEMENT άτομο (όνομα, τηλέφωνο, διεύθυνση)>
<!ELEMENT κατάλογος (άτομο)>
```

- η δήλωση των στοιχείων πρέπει να παρεμβάλλονται από αγκύλες

```
[ στοιχεία ]>
```

- μετά τα στοιχεία μπαίνει η δομή των δεδομένων

στο παράδειγμα μας θα ήταν:

```
<Τοιχώμα>
  <Στρώση υλικού>
    <Ονομά> όνομα στρώσης</όνομα>
      <πλάτος> πλάτος υλικού </πλάτος>
      <Συντελεστής Θερμικής Αγωγιμότητας> Νουμερό συντελεστή
       </Συντελεστής θερμικής Αγωγιμότητας>
  </ Στρώση υλικού >
</ Τοιχώμα >
```

7.1.3 Σχεδιάζοντας ένα αρχείο xml

Εδώ είναι περίπου η γενική διαδικασία για το πως να σχεδιάζεις ένα αρχείο xml.

1. Συγκέντρωση όλων των δεδομένων που πρέπει να συμπεριληφθούν στο αρχείο.

Για παράδειγμα έστω ότι θέλουμε να κατασκευάσουμε ένα xml για το πρόγραμμα μας. έναν τηλεφωνικό κατάλογο. Το πρόγραμμα χρειάζεται να μπορεί να ανακτήσει τα ονόματα των στρώσεων υλικού, τον συντελεστή θερμικής αγωγιμότητας του και τον συντελεστή αντίστασης διαπίδυσης υδρατμών.

2. Ανάλυση των δεδομένων προς αποθήκευση.

Η ανάλυση πρέπει να γίνει μέχρι να καταλήξουμε σε δεδομένα που μπορούμε να τα περιγράψουμε σε μια γραμμή. Για παράδειγμα μια διεύθυνση αποστολής δεν μπορούμε να την περιγράψουμε σε μια γραμμή, δηλαδή πρέπει να αναλυθεί θα δεδομένα που την απαρτίζουν. όνομα, δρόμος, πόλη, κτλ.

3. Εύρεση κατάλληλου ονόματος για να περιγράψει το αρχικό στοιχείο του xml.

Μια και σχεδιάζουμε ένα xml που θα αποθηκεύονται τα δεδομένα του τοιχώματος, τυχαίνει το όνομα "τοίχωμα" ναι είναι και το ιδανικό όνομα για την περίπτωσή μας

4. Εύρεση κατάλληλου ονόματος για την περιγραφή κάθε ιχνοστοιχείου του αρχείου.

7.1.4 Προσπέλαση XML αρχείων

Για να διαβάσουμε και ανά αναλύσουμε τα περιεχόμενα ενός xml αρχείου χρειαζόμαστε έναν αναλυτή xml (xml parser).

Ο parser είναι ένα πρόγραμμα που διαβάζει ένα αρχείο, ελέγχει αν είναι συντακτικά σωστό και αναλύει το κείμενο.

Υπάρχουν 2 τύποι xml αναλυτών που είναι διαδεδομένοι. οι Streaming parser διαβάζουν ένα αρχείο και επιστρέφουν αυτά που βρίσκουν ένα την φορά, διαδοχικά. Εν αντίθεση ένας Parser που βγάζει μορφή δέντρου, κτίζει ένα δέντρο δεδομένων που αντιπροσωπεύει το προς ανάλυση κείμενο. Μόλις η διαδικασία τελειώσει μπορούμε να αναλύσουμε το δέντρο των δεδομένων.

Ας δούμε πως να χρησιμοποιούμε έναν parser που βγάζει μορφή δέντρου σύμφωνα με το μοντέλο **DOM (Document Object Model)**.

Για να μετατρέψουμε ένα αρχείο xml σε ένα δέντρο τύπου **Dom** χρειαζόμαστε έναν αντικείμενο **DocumentBuilder** .

Για να πάρουμε ένα τέτοιο αντικείμενο πρώτα καλούμε ένα νέο στιγμιότυπο τις κλάσης **DocumentBuilderFactory** και μετά την μέθοδο **newDocumentBuilder** τις κλάσης αυτής.

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
```

Τώρα που έχουμε ένα DocumentBuilder, μπορούμε να διαβάσουμε ένα αρχείο. Για να διαβάσουμε το αρχείο αυτό από τον σκληρό δίσκο

δημιουργούμε ένα αντικείμενο "αρχείο" και το τοποθετούμε στην μέθοδο

builder.parse(αρχείο)

```
Document doc= builder.parse(νέο file(όνομα αρχείου));
```

Τώρα το **doc** έχει μέσα του μια δομή σαν να ήταν ένας εικονικός δίσκος στον υπολογιστή μας, με τα δεδομένα του μέσα σε διαφορετικούς φακέλους. Για να πάρουμε τα δεδομένα χρησιμοποιούμε την τοποθεσία του εικονικού φακέλου.

Ο πιο εύκολος τρόπος για να διαβάσουμε το **doc** είναι το συντακτικό **xpath** :

```
XPathFactory xpfactory = XPathFactory.newInstance();
Xpath path = xpfactory.newXPath();
```

και μετά για κάθε δεδομένο:

```
String result = path.evaluate(τοποθεσία, doc)
```

Για παράδειγμα έστω ότι θέλαμε να πάρουμε το όνομα στον τηλεφωνικό κατάλογο που έχει μονό μια καταχώριση μέσα:

```
String result = path.evaluate("/τοίχωμα/όνομα στρώματος/συντελεστής θερμικής αγωγιμότητας/", doc)
```

αν έχουμε παραπάνω από ένα στοιχείο το παίρνουμε με την χρήση αγκυλών[5]

```
String result = path.evaluate("/τοίχωμα[η]/όνομα στρώματος/συντελεστής θερμικής αγωγιμότητας/", doc)
```

[5] το μέτρημα δεν ξεκινάει από το 0 αλλά από το 1, σε αντίθεση με άλλες περιπτώσεις αρίθμησης στον προγραμματισμό γενικά . Τηλεφωνικός κατάλογος[0] θα προκαλέσει σφάλμα

Αυτά είναι τα βασικά πράγματα που χρειαζόμαστε για την χρήση xml. τα xml έχουν αλλά πολυπλοκότερα χαρακτηριστικά αλλά ο σκοπός του βιβλίου δεν είναι για την περιγραφή αυτών.

7.2 HTML

Μια ιστοσελίδα είναι γραμμένη σε μια γλωσσά που ονομάζεται html (hypertext markup Language). Όπως και ο κώδικας της java, ο κώδικας html απαρτίζεται από κείμενο που ακόλουθη κάποιους αυστηρούς κανόνες. Όταν ένας φυλλομετρητής διαβάζει μια ιστοσελίδα, αναλύει τον κώδικα και προβάλει την σελίδα, προβάλλοντας χαρακτήρες, παραγράφους, tables, και εικόνες.

Στο πρόγραμμα μας χρησιμοποιήσαμε html για να κατασκευάσουμε την αναφορά με τα αποτελέσματα του προγράμματος. Εφόσον το πρόγραμμα γράφτηκε σε java για να είναι ουδέτερο λειτουργικών συστημάτων, είναι λογικό να χρησιμοποιήσουμε html για να προβάλουμε τα αποτελέσματα, γιατί και αυτή η γλωσσά είναι ανεξάρτητη πλατφόρμας.

7.2.1 HTML tags

Τα αρχεία html απαρτίζονται από κείμενο και tags που καθοδηγούν τον φυλλομετρητή πώς να προβάλει το κείμενο. Τα περισσότερα tags χρησιμοποιούντα σε ζευγάρια, που αποτελούνται από το tag έναρξης και το tag τερματισμού, και κάθε ζευγάρι αναφέρεται στο κείμενο που βρίσκεται μέσα του.

Ας δούμε κάποια άπλα παραδείγματα από τα πιο βασικά tags.

HTML Headings

Τα HTML headings ορίζονται με τα tags <h1> to <h6>.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

HTML Παράγραφοι

Οι HTML παράγραφοι ορίζονται με το tag <p>.

```
<p>Αυτό είναι μια παράγραφος.</p>
<p>Αυτό είναι μια άλλη παράγραφος.</p>
```

HTML Links

Οι δεσμοί HTML ορίζονται με το tag <a>.

```
<a href="http://www.w3schools.com">Αυτος είναι ενας υπερδεσμός</a>
```

HTML εικόνες

Οι εικόνες HTML ορίζονται με το tag <img>.

```
<img src="w3schools.jpg" width="104" height="142" />
```

7.3 The API Documentation

Οι κλάσεις και οι μέθοδοι της βιβλιοθήκης της JAVA έχουν τις οδηγίες τους μέσα στο api documentation. Ένας προγραμματιστής που χρησιμοποιεί τις κλάσεις για τα κατασκευάσει ένα πρόγραμμα, είναι προγραμματιστής προγραμμάτων. Εν αντίθεση οι προγραμματιστές που σχεδίασαν και προγραμμάτισαν τις κλάσεις που υπάρχουν στην βιβλιοθήκη της JAVA είναι προγραμματιστές συστήματος.

Μπορούμε αν βρούμε το documentation στο internet στην σελίδα:

http://java.sun.com/javase/6/docs/api/index.html

Το documentation περιλαμβάνει λίστα με όλες της κλάσεις που υπάρχουν. Στην JAVA υπάρχουν χιλιάδες. Οι περισσότερες από αυτές είναι μάλλον πολύ ειδικευμένες για έναν αρχάριο προγραμματιστή.

8 Βιβλιογραφία

- Grady Booch, James Rumbaugh, and Ivar Jacobson -The Unified Modeling Language User Guide - Addison-Wesley,2005,1999.
- Cay Horstmann - Big Java 5th edition - Wiley publishing
- Dan Pilone, neil Pitman – Uml 2.0 in a Nutshell - O'reilly
- Steven Holzner – 10 Projects you'll never do at work
- Ralph Morelly, Ralph Walde – Java: Object-Oriented Problem Solving, 3th Edition
- Andrew Davison – Killer Game Programming in Java – O'reilly
- Chris Adamson, Joshua Marinacci – Swing Hacks – O'reilly
- Walter Savitch – Absolute Java
- Walter Savitch –Μια Εισαγωγη στην Επίλυση Προβληματων και στον Πρόγραμματισμο – Εκδοσεις Τζιολα
- Often Needed Answers about Temp, Humidity & Dew Point from the sci.geo.meteorology
  http://www.faqs.org/faqs/meteorology/temp-dewpoint/
- Horstmeyer, Steve "Relative Humidity. The Dew Point Temperature...a better approach
- http://en.wikipedia.org/wiki/Dew_point
- Java Documentation http://java.sun.com/j2se/1.4.2/docs/api/
- http://java.sun.com/docs/books/tutorial/
- Βασικές Αρχές Γλωσσών Προγραμματισμού, Ellis Horowitz, Εκδόσεις Κλειδάριθμος
- «Μετάδοση θερμότητας – Θερμομόνωση», Σ.Ι. Καρέκος, ΤΕΕ, 2001.
- «Τεχνικά Υλικά – Τόμος 2», Αιμ. Γ. Κορωναίος – Γ.Ι. Πουλάκος, ΕΜΠ, 2005.
- «Δομική Φυσική», Walter Blasi, ΙΩΝ, 2000.
- «Handbook of Fundamentals», ASHRAE, 2005.
- «Μετάδοση Θερμότητας – Σημειώσεις», Μ. Κτενιαδάκης, ΤΕΙ Κρήτης, 2003.
- «Θέρμανση – Ψύξη – Αερισμός», Σπ. Χαλικιάς, ΟΕΔΒ, 1983.
- «Θερμικές γεφυρώσεις – συμπυκνώσεις υδρατμών στα στοιχεία εξωτερικού
- κελύφους κτιρίου», Γ. Καλύβας, ΤΕΕ, Τεχνικά Χρονικά, 1998.
- Τεχνικά φυλλάδια εταιρειών θερμομονωτικών και δομικών υλικών.
- Saturation Vapor Pressure over Liquid Water
  http://hurri.kean.edu/~yoh/calculations/satvap/satvap.html

9 Ορισμοί

9.1 ελληνικά σύμβολα

Σύμβολο	Μονάδες	Εξήγηση
φ	[]	σχετική υγρασία του αέρα
Θ_a	[K]	Θερμοκρασία
Θ_k	[K]	σημείο δρόσου ή Θερμοκρασία κορεσμού (αναφορικά με μια συγκεκριμένη μάζα υδρατμών) ή
β_i		Εσωτερικός Συντελεστής υδρατμομετάβασης
β_e		Εξωτερικός Συντελεστής υδρατμομετάβασης
δ_j	$\left[\frac{\text{kg}}{\text{m.h.Pa}}\right]$.	Συντελεστής υδρατμοαγωγιμότητας του υλικού (j)
$\left[\frac{d_j}{\delta_j}\right]$	$\left[\frac{\text{m}^2\text{.h.Pa}}{kg}\right]$.	η αντίσταση διάχυσης των υδρατμών του j-στρώματος. Ορίζεται σαν .
μ_j	[]	συντελεστής αντίστασης σε διάχυση των υδρατμών του j-στρώματος $\mu = \frac{639*10^{-9}}{\delta}$

9.2 Λατινικά σύμβολα

Σύμβολο	Μονάδες	Εξήγηση
M_u	[g]	Μάζα υδρατμών
M_{uk}	[g]	ποσότητα υδρατμών σε κορεσμένο αέρα
a	$\left[\frac{g}{m^3}\right]$	Απόλυτη υγρασία του αέρα
w	$\left[\frac{g}{m^3}\right]$	ειδική υγρασία του αέρα
P	[Pa]	Πίεση
P_a	[Pa]	Μερική πίεση αέρα
P_u	[Pa]	Μερική πίεση υδρατμών
P_k,	[Pa]	Πίεση κορεσμού

g	$\left[\frac{kg}{h*m^2}\right]$	ο ανά μονάδα επιφάνειας ρυθμός ροής της μάζας των υδρατμών, δια μέσου ενός επιπέδου τοιχώματος,
m	$\left[\frac{kg}{h}\right]$	Ο ρυθμός ροής μάζας του υδρατμού
K_D	$\left[\frac{\mathrm{kg}}{\mathrm{m^2hPa}}\right]$	Συντελεστής διαπερατότητας (ή διόδου) υδρατμών του τοιχώματος
A	$[m^2]$	Εμβαδό επιφάνειας του δομικού στοιχείου, κάθετη στη ροή των υδρατμών
P_i	[Pa]	Η μερική πίεση των υδρατμών εσωτερικά
P_e	[Pa]	Η μερική πίεση των υδρατμών εξωτερικά
d_j	[m]	Το πάχος του j-στρώματος
P_{kj}		Πίεση της τάσης κορεσμένων υδρατμών j-στρώματος
U		Συντελεστής θερμοπερατότητας του τοιχώματος,
h_i	$\left[\frac{W}{m^2K}\right]$	Συν.μεταβίβασης θερμότητας, για εσωτερική πλευρά τοίχων
h_e	$\left[\frac{W}{m^2K}\right]$	Συν.μεταβίβασης θερμότητας, για εξωτερική πλευρά τοίχων