ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΥΥ301 / ΠΛΥ 308 ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ **ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ**
ΟΚΤΩΒΡΙΟΣ 2021

Ημερομηνία Παράδοσης: 20-12-2021 **Π. Βασιλειάδης**

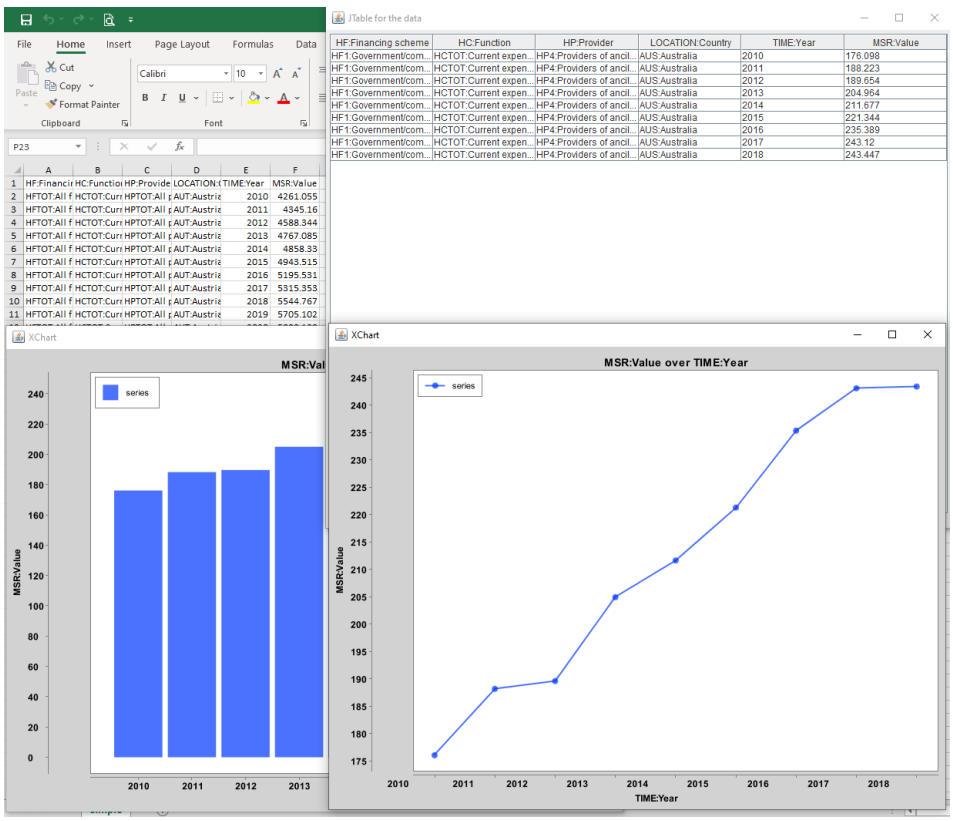
Η προγραμματιστική άσκηση για το μάθημα είναι **υποχρεωτική** και αφορά τη σχεδίαση, υλοποίηση και ρύθμιση ενός συστήματος λογισμικού. Η εργαστηριακή άσκηση προσφέρει **3 μονάδες** στον τελικό βαθμό του μαθήματος και εκπονείται σε ομάδες των **1 - 3 προσώπων**.
ΕΙΔΙΚΑ ΓΙΑ ΤΟ 2021-2022 ΕΠΙΤΡΕΠΕΤΑΙ ΤΟ PROJECT ΝΑ ΕΚΠΟΝΗΘΕΙ ΚΑΙ ΑΤΟΜΙΚΑ!

Φυσικά, πρέπει να πιάσετε τουλάχιστον τη βάση στην εργασία, όπως και στο διαγώνισμα. Σε περιπτώσεις εξαιρετικών εργασιών, η επίδοση επιβραβεύεται με bonus στον τελικό βαθμό.
Το σύστημα πρέπει να υλοποιηθεί σε όλα τα επί μέρους στάδια.

Το φετινό project απαιτεί την κατασκευή ενός προγράμματος το οποίο επιτρέπει σε κάποιον αναλυτή να θέτει φίλτρα σε δομημένα αρχεία δεδομένων (δλδ., αρχεία που οι εγγραφές τους έχουν συγκεκριμένα πεδία), και να οπτικοποιεί τις εγγραφές των αρχείων που πληρούν τα κριτήρια των φίλτρων που βάζει ο αναλυτής.

Κεντρική ιδέα του συστήματος

Ένα δομημένο αρχείο δεδομένων είναι ένα αρχείο κειμένου που περιλαμβάνει εγγραφές, η κάθε μία εκ των οποίων πιάνει μια γραμμή στο αρχείο. Κάθε εγγραφή έχει ένα συγκεκριμένο σύνολο από πεδία, τα οποία χωρίζονται μεταξύ τους με κάποιο διαχωριστικό σύμβολο. Συνήθως αυτό το σύνολο είναι το κόμμα (οπότε έχουμε αρχεία CSV – Comma Separated Values), ή το tab (tsv – Tab Separated Values), ή πιο σπάνια το '|'. Η πρώτη γραμμή του αρχείου περιέχει τα ονόματα των πεδίων.



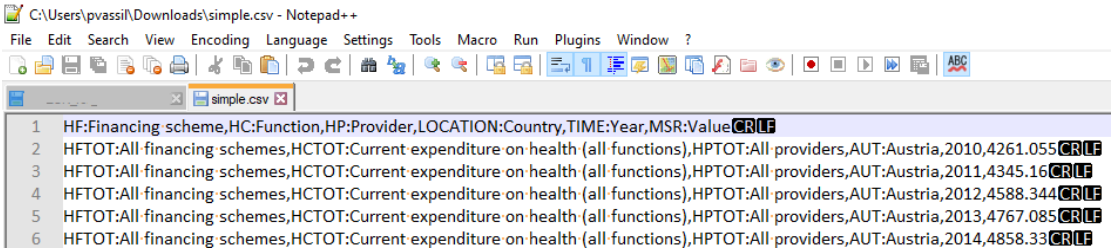
Σχήμα 1. Αρχείο (άνω αριστερά), αποτέλεσμα φίλτρου (άνω δεξιά), γραφικές παραστάσεις (κάτω).

Το πρώτο πράμα που κάνει ο αναλυτής είναι να καταγράψει (register) ένα αρχείο στο σύστημα, ώστε το σύστημα να αναπαραστήσει εσωτερικά τα χαρακτηριστικά του αρχείου: το μονοπάτι του στο λειτουργικό σύστημα, το σύνολο των πεδίων του, καθώς και ένα βοηθητικό όνομα, με το οποίο μπορούμε να αναφερόμαστε σε αυτό, χωρίς να χρειάζεται να χρησιμοποιούμε όλο το path. Για να μπορέσει ο αναλυτής να εργαστεί με μεγάλα αρχεία, πρέπει να μειώσει τον όγκο των εγγραφών που του παρουσιάζονται. Ο αναλυτής, λοιπόν, υποβάλει ένα φίλτρο το οποίο είναι η σύζευξη ενός ή περισσότερων ατομικών φίλτρων. Το σύστημα του επιστρέφει το σύνολο των εγγραφών που πληρούν τα κριτήρια όλων των ατομικών φίλτρων και του το παρουσιάζει. Ο αναλυτής μπορεί να ζητήσει από το σύστημα να καταγράψει την απάντηση σε ένα αρχείο, για να το επεξεργαστεί αργότερα. Επίσης μπορεί να ζητήσει να εμφανιστεί ένα bar chart ή ένα line chart με τα αποτελέσματα που επεστράφησαν.

Λειτουργικότητα του συστήματος

Καταγραφή δομημένου αρχείου δεδομένων. Ο αναλυτής πρέπει να μπορεί να καταγράψει στο σύστημα ένα υπάρχον δομημένο αρχείο δεδομένων, με (α) ένα χαρακτηριστικό όνομα, με το οποίο θα αναφέρεται στο αρχείο στο μέλλον (ώστε να μη δίνει όλο το path), (β) το μονοπάτι για το αρχείο, και (γ) τη συμβολοσειρά με την οποία διαχωρίζονται τα πεδία εντός του αρχείου.
Υποθέσεις για το αρχείο:

- Η πρώτη γραμμή του αρχείου περιέχει τα ονόματα των πεδίων του
- Όλες οι γραμμές έχουν τον ίδιο αριθμό διαχωριστικών (δλδ., τον ίδιο αρ. πεδίων, ακόμα και αν κάποιο πεδίο είναι χωρίς τιμή)



Σχήμα 2: οι πρώτες γραμμές από ένα δομημένο αρχείο. Παρατηρήστε πώς το κόμμα χωρίζει τις διάφορες στήλες, και πώς η πρώτη γραμμή μας λέει τα ονόματα των πεδίων.

Ανάκτηση μεταπληροφορίας. Ο αναλυτής μπορεί να ανακτήσει τη λίστα από τα ονόματα των πεδίων ενός καταγεγραμμένου αρχείου δεδομένων.

Επερώτηση αρχείου με φίλτρο. Ο αναλυτής υποβάλει ερωτήσεις φίλτρων στο αρχείο. Για το σκοπό αυτό, μπορεί να κατασκευάσει ένα σύνθετο φίλτρο, μέσω του συνδυασμού απλών φίλτρων. Έτσι, για διάφορα πεδία του αρχείου, μπορεί να φτιάξει από ένα ατομικό φίλτρο για το καθένα εξ' αυτών, και όλα αυτά μαζί να τα συνθέσει σε ένα σύνθετο φίλτρο.

- Ένα ατομικό φίλτρο αφορά ένα μόνο πεδίο (εξ' ου και ατομικό) και είναι είτε της μορφής `field = value`, είτε της μορφής `field in {value₁, value₂, …, valueₙ}` πάνω σε ένα συγκεκριμένο αρχείο.
- Το σύνθετο φίλτρο έρχεται στη μορφή Map<String, List<String>>, με το πρώτο String να είναι το όνομα του πεδίου του ατομικού φίλτρου και το List<String> να είναι μια λίστα από επιτρεπτές τιμές για το πεδίο αυτό. Προφανώς, στο Map μπορούμε να βάλουμε πολλά τέτοια ζεύγη <πεδίο, λίστα επιτρεπτών τιμών>.

Παράδειγμα: αν π.χ., θέλει κανείς το ατομικό φίλτρο *Month in {Jan, Feb, Mar}*, αυτό περιγράφεται ως το ζεύγος (α) το πεδίο *Month*, και, (β) η λίστα *{Jan, Feb, Mar}* ως επιτρεπτές τιμές. Αν τώρα θέλουμε να προσθέσουμε το επιπλέον φίλτρο *Country = Greece*, και με το συνδυασμό των δύο ατομικών φίλτρων να φτιάξουμε ένα σύνθετο, πρέπει σε ένα Map να βάλουμε (α) ένα key "Month" και ένα value με μια λίστα τιμών <"Jan", "Feb", "Mar">, και (β) ένα key "Country" και ένα value με μια λίστα τιμών <"Greece">. Το αποτέλεσμα είναι μια λίστα από ακριβώς εκείνες τις εγγραφές του αρχείου που πληρούν ακριβώς τα κριτήρια που προδιαγράφηκαν στο φίλτρο, η οποία και επιδεικνύεται στον αναλυτή.
Είναι χρήσιμο, το front-end της εφαρμογής να χρησιμοποιεί ένα Map για να δώσει ως κλειδί ένα φιλικό όνομα σε κάθε ερώτηση-φίλτρο και να κρατά το αποτέλεσμα ως τιμή, ώστε μετά, να μπορεί να ανακτήσει την λίστα αυτή που ανακτήθηκε για περεταίρω επεξεργασία (π.χ., οπτικοποίηση).

Εκτύπωση του αποτελέσματος ενός φίλτρου σε αρχείο. Ο αναλυτής μπορεί να δηλώσει το αποτέλεσμα ποιας ερώτησης επιθυμεί να γραφεί σε ένα αρχείο.
/ όπως θα δείτε, ενώ στο front-end λέμε απλά «σώσε σε ένα αρχείο», στο back-end, ζητείται να σωθεί το αρχείο σε ένα προσδιορισθέν PrintStream. Το σχετικό PrintStream πρέπει είτε να προετοιμασθεί κατάλληλα αν πρόκειται για αρχείο εξόδου, ή να είναι το System.out για απλή εκτύπωση στην κονσόλα. */*

Παρουσίαση αποτελέσματος σε chart. Το αποτέλεσμα μιας ερώτησης μπορεί να παρουσιαστεί σε ένα διδάστατο γράφημα (Chart) με 2 άξονες, τον οριζόντιο (x-axis) και τον κάθετο (y-axis). Το γράφημα μπορεί να είναι είτε Ραβδόγραμμα (BarChart), το οποίο είναι η πιο γενική μορφή οπτικής αναπαράστασης, είτε Γραμμική αναπαράσταση (LineChart) το οποίο χρησιμοποιείται όταν ο οριζόντιος άξονας είναι ο χρόνος, ή κάποιο μέγεθος που αυξάνει μονότονα και σε τακτά διαστήματα (με άλλα λόγια είναι ισομορφικό στους ακέραιους). Ο αναλυτής προσδιορίζει τον τύπο του γραφήματος και ποια πεδία παίζουν το ρόλο του κάθε άξονα, καθώς και αν το αποτέλεσμα θέλουμε να σωθεί σαν αρχείο png (οπότε και πρέπει να δώσει ένα όνομα αρχείου για την αποθήκευση).

Προγραμματιστικές συστάσεις

Διαχείριση οπτικοποίησης γραφικών αναπαραστάσεων

Θα χρησιμοποιήσετε τη βιβλιοθήκη ***XYChart*** που διατίθεται ως free open source software (FOSS) που θα το βρείτε στο https://github.com/knowm/XChart και στο https://knowm.org/open-source/xchart/

Θα βρείτε παραδείγματα οπτικοποίησης γραφικών παραστάσεων στον σύνδεσμο https://knowm.org/open-source/xchart/xchart-example-code/ και στον σύνδεσμο https://github.com/knowm/XChart/tree/develop/xchart-demo/src/main/java/org/knowm/xchart/demo/charts
καθώς και στο υλικό που ανεβάσαμε στο βοηθητικό υλικό, όπου και θα βρείτε και το σχετικό download της βιβλιοθήκης (από http://knowm.org/open-source/xchart/xchart-change-log).

Το βασικό jar που θα πρέπει να βρίσκεται στο build path της εφαρμογής σας είναι το `xchart-3.8.1.jar`.

Προφανώς, το να μάθετε να χρησιμοποιείτε μια βιβλιοθήκη γραφικών απεικονίσεων θα σας βοηθήσει στο μέλλον και ευρύτερα, πέραν του μαθήματος.

Software Architecture & Specifications

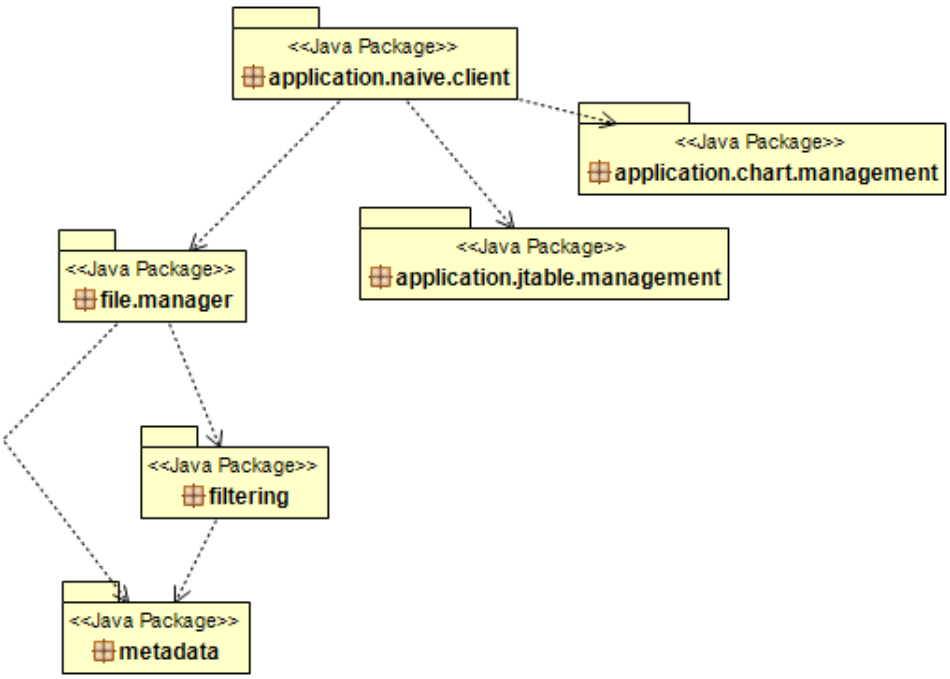
Η αρχιτεκτονική του λογισμικού σε πακέτα, σας δίδεται σχετικά προκαθορισμένα. Το υπό κατασκευή σύστημα είναι ένα **σύστημα 2 επιπέδων**, ενός **front-end client** κομματιού που είναι υπεύθυνο για την διάδραση με τον αναλυτή και ενός **back-end server** κομματιού που είναι υπεύθυνο για την διεκπεραίωση των use cases που προκύπτουν από την αρχική περιγραφή της λειτουργικότητας του συστήματος.

Στην back-end πλευρά του server υπάρχουν διάφορα packages (πακέτα) που το καθένα έχει τη δική του λειτουργικότητα. Τα πακέτα είναι:

`file.manager`: το πακέτο υποστηρίζει τη διαχείριση των αρχείων, και συγκεκριμένα (α) την καταγραφή τους, (β) την επιστροφή κάποιας μεταπληροφορίας (π.χ., τις στήλες) κάθε εγγεγραμμένου αρχείου, (γ) την διεκπεραίωση της επιβολής ενός φίλτρου σε ένα αρχείο και την επιστροφή των αποτελεσμάτων και (δ) την καταγραφή των αποτελεσμάτων σε ένα PrintStream. Η κεντρική μηχανή μεταβιβάζει αρμοδιότητες (delegates) σε δύο υποσυστήματα του back-end, που περιγράφονται ευθύς αμέσως.

`filtering`: το πακέτο διεκπεραιώνει τη δουλειά της αποτίμησης ενός φίλτρου. Για το σκοπό έχει μια filtering engine η οποία, αφού λάβει την πληροφορία για το ποιο φίλτρο θα επιβληθεί σε ποιο αρχείο, εκτελεί το σχετικό φιλτράρισμα και επιστρέφει τις εγγραφές που πληρούν τα κριτήρια του φίλτρου..

`metadata`: το πακέτο περιλαμβάνει ένα διαχειριστή μεταπληροφορίας που ξέρει τα αρχεία που έχουν ήδη καταγραφεί και επιστρέφει την μεταπληροφορία αυτή (alias, στήλες και θέσεις αυτών, μονοπάτι, κλπ.) στα άλλα υποσυστήματα.



Σχήμα 1Αρχιτεκτονική των πακέτων του συστήματος

Στη front-end πλευρά έχει προβλεφθεί ένα πακέτο για να στεγάσει τη διεπαφή με τον αναλυτή.

`naive.client`: το πακέτο έχει τη βασική κλάση που θα περιέχει τη main του προγράμματός σας και η οποία δεν θα κάνει υπολογισμούς, παρά μόνο θα ζητά σε ένα application controller να μιλήσει με το back-end σύστημα (και αναλόγως του τι ζητά ο αναλυτής, να εκτελέσει τη σχετική εργασία και να φέρει τα όποια αποτελέσματα)

`jtable.management`: πακέτο με την υπευθυνότητα να παίρνει ένα αποτέλεσμα φιλτραρίσματος και να το δείχνει σε ένα JTable.

`chart.management`: πακέτο με την υπευθυνότητα να παίρνει ένα αποτέλεσμα φιλτραρίσματος και να το δείχνει σε ένα LineChart ή ένα BarChart.

Έχετε δικαίωμα να υλοποιήσετε τις κλάσεις που λείπουν στο εσωτερικό των πακέτων με όποιο τρόπο θέλετε εσείς. **Είναι υποχρεωτικό και απαράβατο, όμως, ΝΑ ΜΗΝ ΑΛΛΑΞΕΤΕ τα**

interfaces που σας δίνονται, αλλά να τα υλοποιήσετε επακριβώς. Το ποιες κλάσεις θα εντάξετε μέσα στο κάθε πακέτο είναι δικό σας θέμα και αντικείμενο της σχεδίασης, υλοποίησης και ελέγχου που θα κάνετε, καθώς και της αξιολόγησής τους.

Τι σας δίνεται έτοιμο

Σας δίνονται στον φάκελο
http://www.cs.uoi.gr/~pvassil/courses/sw_dev/exercises/supportingMaterial/2021-2022/

Α. ένα αρχείο .zip με το σκελετό ενός Eclipse Java project με
(α) τη δομή του src, σημαντικό κομμάτι του κώδικα στο front-end, και τα σχετικά interfaces of the back-end,
(β) αρχεία με δεδομένα (υπο-φάκελος resources) για να κάνετε ελέγχους και για να τρέξετε το σύστημα που θα φτιάξετε,
(γ) ένας φάκελος libs με το σχετικό jar που σας χρειάζεται για να κάνετε εύκολη τη ζωή σας.
(δ) τεστ που υποδεικνύουν ελέγχους για τις πιο σημαντικές μεθόδους.

Η κεντρική ιδέα. Στον client πρέπει να φτιάξετε το προγραμματάκι που θα χρησιμοποιεί κατά βάση τον Application Controller για να μιλήσει με το back-end. Το back-end στηρίζεται σε κάποια interfaces, για τα οποία θα πρέπει να φτιαχτούν τα σχετικά factories και οι υποστηρικτικές κλάσεις.
Στα τεστ, έχουν δοθεί κάποια αρχικά τεστ, τα οποία περιλαμβάνουν ονόματα κλάσεων από το back-end που φυσικά δεν υπάρχουν. Μπορείτε να διαφοροποιηθείτε στα ονόματα που θα χρησιμοποιήσετε για τις κλάσεις που υλοποιούν τα interfaces, αλλά η λογική των τεστ σας δείχνει τι πρέπει να γίνεται. **Τα test θα πρέπει να τρέχουν με επιτυχία στο τελικό σας παραδοτέο! Φυσικά θα πρέπει να φτιάξετε και επί μέρους tests και εσείς**.
Χρησιμοποιήστε Junit 4 και όχι 5 (έχει πολύ περισσότερο υλικό στο διαδίκτυο).

Important ToDo. Εσείς πρέπει:
- Να μετονομάσετε το δοθέν Eclipse project ώστε στο όνομα που θα έχει, τα "AM1", "AM2", "AM3" να αντικατασταθούν από τους συγκεκριμένους Αρ. Μητρώου των μελών της ομάδας, με αύξουσα σειρά, (ώστε αφού τα κάνετε turnin να ξέρουμε ποιανού είναι κάθε project). Π.χ., αν στην ομάδα μετέχουν οι φοιτητές με ΑΜ 345, 344, 567, το project υποχρεωτικά πρέπει να ονομάζεται ως: `344_345_567_StructuredFileFilter` (Στο Eclipse, σχεδόν τα πάντα μετονομάζονται με δεξί κλικ -> Refactor -> Rename).
- Να συμπληρώσετε και τις υπόλοιπες κλάσεις του συστήματος.
- Να προσθέσετε ένα αρχείο κειμένου Readme.txt που θα λέει (α) ονόματα και ΑΜ, (β) αν τυχόν χρειάζεται, τι απαιτείται για να τρέξει το πρόγραμμά σας χωρίς προβλήματα στον έλεγχο από τους βοηθούς (π.χ., κάποιες ειδικές βιβλιοθήκες που τυχόν χρησιμοποιήσατε)

Β. Ένας φάκελος με παραδείγματα που μπορείτε να δείτε για τις οπτικοποιήσεις.
Γ. Ένας φάκελος με το κατεβασμένο XYChart.jar και κάποια σχετικά URL που μπορεί να σας ενδιαφέρουν

Οδηγίες και υποχρεώσεις

ΕΙΝΑΙ ΑΠΑΡΑΙΤΗΤΟ ΤΟ PROJECT ΣΑΣ ΝΑ ΦΤΑΣΕΙ ΜΕΧΡΙ ΚΑΙ ΤΗΝ ΥΛΟΠΟΗΣΗ ΚΑΙ Ο ΚΩΔΙΚΑΣ ΣΑΣ ΝΑ ΤΡΕΧΕΙ! (Δεν είναι προβιβάσιμα projects που ο κώδικάς τους δεν τρέχει)

ΕΙΝΑΙ ΑΠΑΡΑΙΤΗΤΟ Ο ΚΩΔΙΚΑΣ ΣΑΣ ΝΑ ΣΥΝΟΔΕΥΕΤΑΙ ΑΠΟ ΤΑ ΑΝΤΙΣΤΟΙΧΑ TESTS!

Είναι <u>απαράβατο</u> είναι ΝΑ ΥΛΟΠΟΙΗΣΕΤΕ ΕΠΑΚΡΙΒΩΣ ΚΑΘΕ INTERFACE σας δίδεται. Αυτή είναι και η ιδέα, να δείτε στην πράξη, τι στην ευχή τα θέλουμε αυτά τα interfaces: μόλις σας δόθηκε ένα «συμβόλαιο» λειτουργικότητας μεταξύ υποσυστημάτων (π.χ., του back-end server με τον client), το οποίο πρέπει να υλοποιήσετε επακριβώς.

Επιβάλλεται να ακολουθήσετε τις βασικές αρχές ενθυλάκωσης (υποχρεωτικά), χαμηλής σύζευξης, DIP, OCP, abstract coupling, factories κλπ (όσο αυτό είναι εφικτό και εύλογο).

Υπόδειγμα αναφοράς: για τα επιμέρους στάδια, μπορείτε να συμπληρώνετε / αναθεωρείτε σταδιακά την αναφορά σας. Για διευκόλυνσή σας, υπάρχει ένα υπόδειγμα στο http://www.cs.uoi.gr/~pvassil/courses/sw_dev/exercises/TemplateFinalReport.zip .
ΥΠΟΧΡΕΩΤΙΚΑ ΠΡΕΠΕΙ ΝΑ ΑΚΟΛΟΥΘΗΣΕΤΕ ΤΟ ΥΠΟΔΕΙΓΜΑ ΠΟΥ ΣΑΣ ΔΙΝΕΤΑΙ!

Υλικό. Όλο το υποστηρικτικό υλικό για το project θα βρίσκεται στο URL
http://www.cs.uoi.gr/~pvassil/courses/sw_dev/exercises/supportingMaterial/
στον υποφάκελο της φετινής χρονιάς

Στη διάρκεια του εξαμήνου:

- Σίγουρα θα δοθούν περαιτέρω εξηγήσεις στην εκφώνηση & ενδεχομένως να αλλάξει/εμπλουτισθεί κάποιο μέρος της εκφώνησης! (άρα το δημοσιευθέν αρχείο της εκφώνησης μπορεί να αλλάζει).
- **Αργότερα** στο εξάμηνο, **θα σας ζητηθεί να εγγράψετε την ομάδα σας** σε μία φόρμα εγγραφής. Ομάδες που δεν εγγραφούν κινδυνεύουν να μην βαθμολογηθούν. Θα δοθούν οδηγίες αφού έχουμε μια πιο καθαρή εικόνα του τι γίνεται με το εξάμηνο.

Χρονοδιάγραμμα

Στη συνέχεια παρατίθενται στάδια της ανάπτυξης, ενδιάμεσες προθεσμίες (milestones) και καταληκτικές ημερομηνίες ολοκλήρωσης (deadlines).

[**04/10**]	Εκφώνηση
3 weeks	*Setup of Infrastructure*
[**24/10:: 23.59**]	Εγκατάσταση Java (version 1.8 is safe), Eclipse (last v.) στους Η/Υ σας Για όσους έχουν δυσκολία με την Java: ασκήσεις επανάληψης Εξοικείωση και πειραματισμός με το υλικό που σας δίδεται Εκκίνηση εργασιών στα Use Cases της εκφώνησης Δεν υπάρχει κάτι να παραδώσετε
4 weeks	*Design of classes && first implementations*
[21/11:: 23.59]	Turnin: DLV1.1: First version of the report (pdf) with (a) use cases + (b) the OREOS part for the test cases + (c) a first version of the class diagram(s) with the design of the system (προαιρετικά: any other diagrams)
4 weeks	*Complete implementation*
[**19/12:: 23.59**] **ΑΝΕΛΑΣΤΙΚΑ**	Turnin : DLV 2.1: a single, all-encompassing zip file with the code for all classes DLV 2.2: FINAL version of the report (pdf) with all the design and the documentation of the project
@ ALL turnin's	***From the folder resources, you can exclude all files that you do not use as input (which cannot be larger than 10MB); still: update the README.txt accordingly!***

Δεν θα ξεπεράσουμε το όριο των Χριστουγέννων. Η πράξη έχει αποδείξει ότι στις γιορτές οι ομάδες αποσυντονίζονται σε πολύ μεγάλο βαθμό. Έτσι, **Η ΠΡΟΘΕΣΜΙΑ ΤΗΣ ΠΑΡΑΔΟΣΗΣ ΕΙΝΑΙ ΑΝΕΛΑΣΤΙΚΗ!** Θα πιεστείτε περισσότερο πριν τις γιορτές, αλλά θα φύγετε για τις γιορτές χωρίς το φορτίο του project.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ!