ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

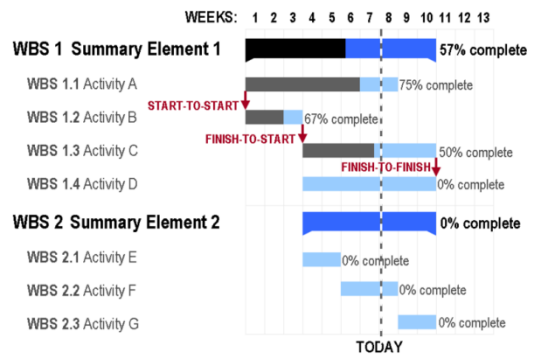
ΜΥΥ301 / ΠΛΥ 308 ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ **ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ**
ΟΚΤΩΒΡΙΟΣ 2022

Ημερομηνία Παράδοσης: 18-12-2022 **Π. Βασιλειάδης**

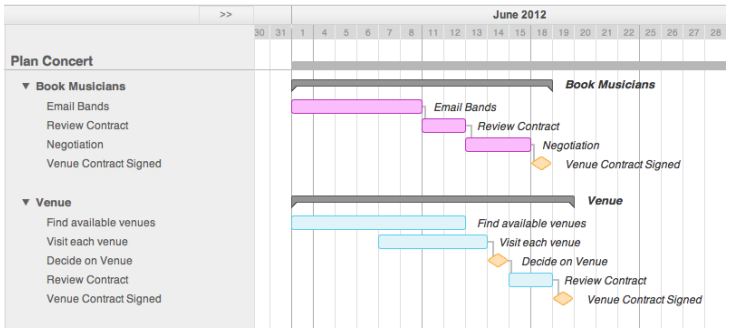
Η προγραμματιστική άσκηση για το μάθημα είναι **υποχρεωτική** και αφορά τη σχεδίαση, υλοποίηση και ρύθμιση ενός συστήματος λογισμικού. Η εργαστηριακή άσκηση προσφέρει **3 μονάδες** στον τελικό βαθμό του μαθήματος και εκπονείται σε ομάδες των **1 - 3 προσώπων**.
ΕΙΔΙΚΑ ΓΙΑ ΤΟ 2022-2023 ΕΠΙΤΡΕΠΕΤΑΙ ΤΟ 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.

Ένα *έργο* (project) είναι μια οργανωμένη προσπάθεια που έχει σχεδιασθεί κατάλληλα, με σκοπό την υλοποίηση ενός καλά ορισμένου στόχου, που μπορεί να είναι ένα νέο προϊόν, υπηρεσία, ή αποτέλεσμα. Λόγω πολυπλοκότητας, βασική αρχή της διαχείρισης έργων είναι η κατάτμησή τους σε επί μέρους εργασίες και ο χρονοπρογραμματισμός τους. Ένα βασικό εργαλείο διαχείρισης έργων είναι τα διαγράμματα Gantt. Δείτε δύο παραδείγματα διαγραμμάτων Gantt.



https://en.wikipedia.org/wiki/Gantt_chart



http://teamgantt.com/blog/2012/05/gantt-chart-example/

Η βασική ιδέα σε ένα διάγραμμα Gantt είναι ότι χρησιμοποιούμε *εργασίες* (tasks) ως το βασικό δομικό υλικό ενός έργου. Μία εργασία είναι μια συμπαγής δράση που έχει ένα συγκεκριμένο στόχο στο πλαίσιο του έργου: είναι ένα «βήμα» προς τον τελικό στόχο. Οι εργασίες έχουν ένα ακέραιο αριθμό που τις χαρακτηρίζει και ένα κείμενο που περιγράφει την ουσία τους. Επίσης έχουν ημερομηνία έναρξης, ημερομηνία λήξης (και κατά συνέπεια μια διάρκεια) και ένα κόστος.

Οι εργασίες είναι είτε απλές είτε σύνθετες. Οι απλές εργασίες έχουν μια ημερομηνία έναρξης, μια ημερομηνία λήξης και ένα κόστος. Οι σύνθετες εργασίες (α) αποτελούνται από άλλες εργασίες και (β) έχουν τα ίδια στοιχεία με τις απλές, αλλά αυτά υπολογίζονται ως εξής: (α) η μικρότερη από τις ημερομηνίες έναρξης των υποεργασιών είναι η έναρξη της σύνθετης, (β) η μεγαλύτερη από τις ημερομηνίες λήξης των υποεργασιών είναι η λήξη της σύνθετης, και (γ) το κόστος της σύνθετης εργασίας είναι το άθροισμα των επί μέρους κοστών των υποεργασιών της.

Αν μια εργασία (απλή ή σύνθετη) δεν περιλαμβάνεται ως μέρος μιας σύνθετης εργασίας, τότε τη λέμε εργασία *κορυφαίου επιπέδου* (top level). Αν μια εργασία είναι μέρος μιας άλλης (αναγκαστικά: σύνθετης) εργασίας, τότε τη λέμε και *υποεργασία* (subtask). *Μπορείτε να κάνετε την απλοποιητική υπόθεση εργασίας ότι οι σύνθετες εργασίες έχουν μόνο απλές εργασίες ως υποεργασίες (δλδ., δε θα τύχει μια σύνθετη εργασία να έχει στις υποεργασίες της μια άλλη σύνθετη).*

Καλείσθε να κατασκευάσετε ένα σύστημα διαχείρισης διαγραμμάτων Gantt. Οι λειτουργίες που πρέπει να υποστηρίζονται είναι:

Φόρτωση από απλό κείμενο. Το σύστημα θα πρέπει να μπορεί να φορτώσει μια αποθηκευμένη αναπαράσταση ενός Gantt diagram από ένα αρχείο κειμένου. Το αρχείο που θα φορτωθεί θα είναι ένα απλό delimited αρχείο κειμένου. Εδώ το δείχνουμε με formatted περιεχόμενο για να εξηγήσουμε τι αναμένεται.

TaskId	TaskText	MamaId	Start	End	Cost
100	***Prepare a shopping list***	***0***			
101	Visit all the closets to see what's missing	100	1	5	50
102	Write down what you need to buy	100	1	6	10
200	**Proceed to the super market**	**0**	**7**	**9**	**10**
300	***Buy and pay***	***0***			
307	Put stuff in the market basket	300	9	15	40
302	Pay at the cashier and leave	300	15	28	442

Κάθε εργασία είναι μία γραμμή του αρχείου. Οι στήλες χωρίζονται με ένα διαχωριστικό string, π.χ., ένα tab. Κάθε εργασία πρέπει να έχει ένα μοναδικό TaskId το οποίο δεν μπορεί να το έχει άλλη. Αν μια εργασία είναι top-level, το πεδίο mamaId είναι 0 (ναι, απαγορεύεται να υπάρχει εργασία με taskId == 0, το 0 στο mamaId είναι μια σύμβαση για να μας πει ότι η εργασία είναι κορυφαίου επιπέδου). Αν μια εργασία είναι σύνθετη, ΔΕΝ περιλαμβάνει στοιχεία για ημερομηνίες και κόστος (θα υπολογισθούν από τα συστατικά της). Στο παραπάνω παράδειγμα:

- Η αρχική μπλε header γραμμή είναι απλά για να δείξει τι πεδία αναμένεται να έχουν οι εργασίες – μπορεί και να λείπει από το αρχείο σας.
- Οι τοπ εργασίες είναι με bold (100, 200, 300).
- Οι σύνθετες είναι με italics (100, 300)

Είναι σημαντικό ότι το αρχείο μπορεί να περιέχει τις εργασίες με οποιαδήποτε σειρά. Εσείς αφού τις φορτώσετε, θα πρέπει να έχετε κρατήσει κάπου την λίστα όλων των εργασιών ταξινομημένη ως εξής:

- Όλες οι top-level εργασίες να είναι ταξινομημένες ως προς το start χρονικό σημείο τους, και αν υπάρχουν ισοπαλίες, να προηγείται αυτή με το μικρότερο taskId.
- Ανάμεσα στη top-level εργασία $task_i$ και την επόμενη top-level εργασία $task_{i+1}$, μπαίνουν οι υποεργασίες της $task_i$ (αν υπάρχουν), που ταξινομούνται ομοίως: αρχικά με βάση το start και αν υπάρχει ισοπαλία με βάση το taskId.

Προφανώς, άμα η συνολική λίστα είναι ταξινομημένη όπως προαναφέρθηκε, αν ζητηθούν μόνο οι top-level εργασίες, είναι κι αυτές ταξινομημένες.

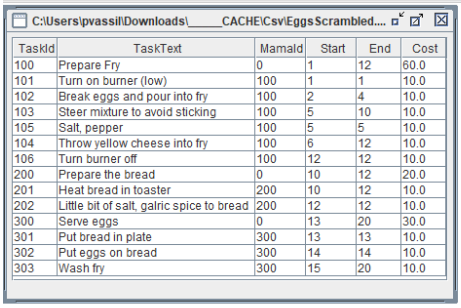
Το παράδειγμα που έχει το βιντεάκι είναι πιο απλό. Οι top-level είναι όλες σύνθετες.

TaskId	TaskText	MamaId	Start	End	Cost
100	***Prepare Fry***	***0***			
101	Turn on burner (low)	100	1	1	10
102	Break eggs and pour into fry	100	2	4	10
103	Steer mixture to avoid sticking	100	5	10	10
104	Throw yellow cheese into fry	100	6	12	10
105	Salt, pepper	100	5	5	10
106	Turn burner off	100	12	12	10
200	***Prepare the bread***	***0***			
201	Heat bread in toaster	200	10	12	10
202	Little bit of salt, galric spice to bread	200	12	12	10
300	***Serve eggs***	***0***			
301	Put bread in plate	300	13	13	10
302	Put eggs on bread	300	14	14	10
303	Wash fry	300	15	20	10

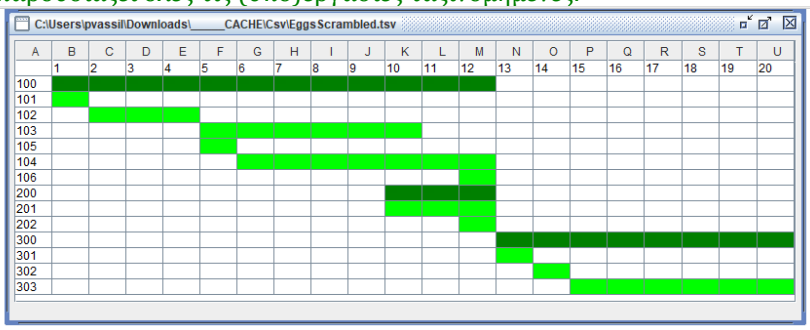
Προσέξτε όμως ότι το αρχειάκι δεν είναι ταξινομημένο σωστά και χρειάζεται ταξινόμηση: η 104 θα πρέπει να πάει μετά την 105, λόγω του ότι ξεκινά πιο μετά!

[ΔΙΟΡΘΩΣΗ. Η ανάκτηση όλων των εργασιών, δεν είναι χωριστό use case: όπως σωστά παρατηρήσατε, αυτό γίνεται αυτόματα μετά τη φόρτωση, και όχι ως χωριστή πράξη!]

Ανάκτηση πλήρως όλων των εργασιών. ~~Αντίστοιχο με το προηγούμενο, αλλά εδώ παρουσιάζονται και όλες οι (υπο)εργασίες αναλυτικά. Πάλι ταξινομημένο~~. Αμέσως μετά τη φόρτωση, το σύστημα αυτόματα παρουσιάζει όλες τις (υπο)εργασίες ταξινομημένες.

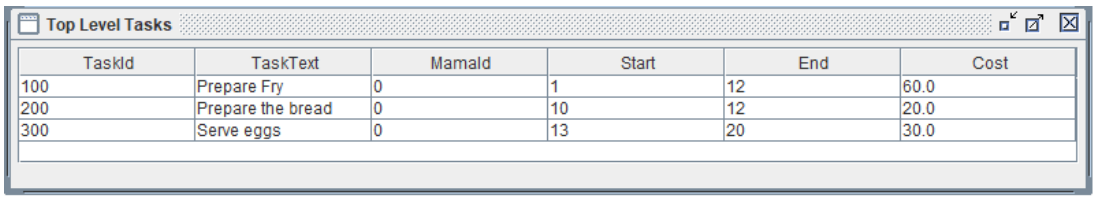

C:\Users\pvassil\Downloads_____CACHE\Csv\EggsScrambled....

TaskId	TaskText	Mamaid	Start	End	Cost
100	Prepare Fry	0	1	12	60.0
101	Turn on burner (low)	100	1	1	10.0
102	Break eggs and pour into fry	100	2	4	10.0
103	Steer mixture to avoid sticking	100	5	10	10.0
105	Salt, pepper	100	5	5	10.0
104	Throw yellow cheese into fry	100	6	12	10.0
106	Turn burner off	100	12	12	10.0
200	Prepare the bread	0	10	12	20.0
201	Heat bread in toaster	200	10	12	10.0
202	Little bit of salt, galric spice to bread	200	12	12	10.0
300	Serve eggs	0	13	20	30.0
301	Put bread in plate	300	13	13	10.0
302	Put eggs on bread	300	14	14	10.0
303	Wash fry	300	15	20	10.0

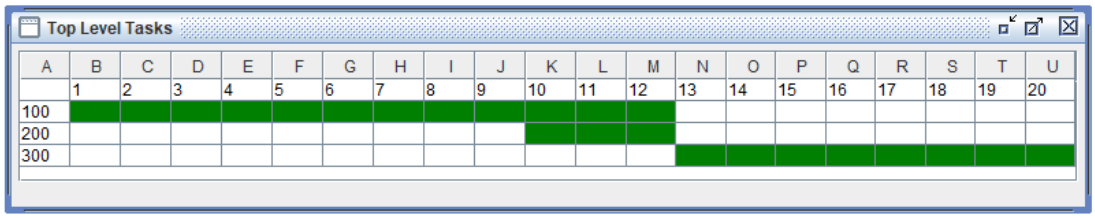

C:\Users\pvassil\Downloads_____CACHE\Csv\EggsScrambled.tsv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
100																					
101																					
102																					
103																					
105																					
104																					
106																					
200																					
201																					
202																					
300																					
301																					
302																					
303																					

Ανάκτηση των εργασιών κορυφαίου επιπέδου. Στο report αυτό μας ενδιαφέρει να παρουσιαστούν μόνο οι εργασίες κορυφαίου επιπέδου, αυτές δλδ., που δεν είναι υποεργασίες κάποιας άλλης εργασίας, ταξινομημένες. Για κάθε τέτοια εργασία επιστρέφονται όλα τα στοιχεία τους: id, περιγραφή, αρχή, τέλος, κόστος. Προφανώς πρέπει, με ενιαίο τρόπο (άρα via a public API της σχετικής κλάσης) να μπορούν να ανακτηθούν όλα τα στοιχεία, είτε δόθηκαν, είτε προκύπτουν από υπολογισμό (π.χ., το κόστος μιας σύνθετης εργασίας).


Top Level Tasks

TaskId	TaskText	Mamaid	Start	End	Cost
100	Prepare Fry	0	1	12	60.0
200	Prepare the bread	0	10	12	20.0
300	Serve eggs	0	13	20	30.0


Top Level Tasks

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
100																					
200																					
300																					

Ανάκτηση όλων των εργασιών που πληρούν το ίδιο κριτήριο περιγραφής. Ανακτώνται ταξινομημένες, μόνο οι εργασίες που οι περιγραφή τους ξεκινά (prefix) με ένα δοθέν string. Στο αρχικό παράδειγμα, αν δοθεί η συμβολοσειρά "Pr" επιστρέφονται οι εργασίες 100 και 200. Αν δοθεί η συμβολοσειρά "P" επιστρέφονται οι εργασίες 100, 200, 307, 302 με αυτή τη σειρά.

Ανάκτηση εργασίας με συγκεκριμένο taskId. Όπως το παραπάνω, αλλά με επακριβή αναζήτηση του taskId (όχι δλδ με prefix).

Αποθήκευση σε απλό κείμενο, html και markdown. Θέλουμε μια αναφορά να μπορεί να αποθηκευθεί. Θα υποστηρίξουμε εναλλακτικούς τρόπους αποθήκευσης: (α) απλό tab-delimited κείμενο, (β) markdown και (γ) html format.

Για ένα text κείμενο, θέλουμε να βγει μια αναπαράσταση, ταξινομημένη, όπως και στο αρχείο εισόδου. Στο παράδειγμά μας, ασχέτως του πόσο ανακατεμένες θα ήταν οι εγγραφές στο input, το αρχείο που θα προέκυπτε θα ήταν σαν αυτό του παραδείγματος (με tabs ανάμεσα στις κολώνες). Εδώ υποχρεωτικά θέλουμε header line στην αρχή.

Για ένα markdown κείμενο, θέλουμε το αντίστοιχο, όπου επιπλέον: (α) η αρχική γραμμή τίτλου είναι σε italics και (β) οι top-level tasks με bold. Δείτε για την πολύ απλή γλώσσα επισημείωσης markdown στο https://en.wikipedia.org/wiki/Markdown

Για ένα html κείμενο, κανένα ιδιαίτερο formatting, απλά ένα πινακάκι, με <tr> και <td> για γραμμές και κολώνες. *Δείτε το παράδειγμα με την εξαγωγή φάσεων χρονοσειράς στο site του μαθήματος (παραδείγματα της ενότητας 6) για το πώς σχεδιάζεται και υλοποιείται μια τέτοια λύση.*

Έξοδος. Έξοδος από το πρόγραμμα. Ο χρήστης ερωτάται αν είναι σίγουρος ότι θέλει να φύγει από το πρόγραμμα και αν ναι, του κάνουμε τη χάρη.

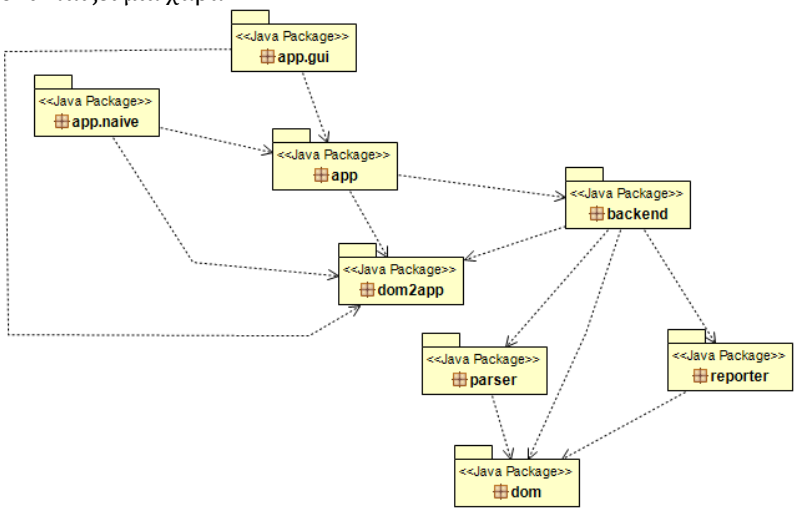
Software Architecture & Specifications

Η αρχιτεκτονική του λογισμικού σε πακέτα, σας δίδεται εν μέρει προκαθορισμένα. Το υπό κατασκευή σύστημα οφείλει να είναι ένα **σύστημα 2 επιπέδων**, ενός **front-end client** κομματιού που είναι υπεύθυνο για την διάδραση με τον διαχειριστή ενός έργου 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.

Α. Στην back-end πλευρά του server οφείλουν να υπάρχουν διάφορα packages (πακέτα), έκαστο με τη δική του λειτουργικότητα. Στην υλοποίηση που έκανα εγώ για να ελέγξω την εκφώνηση, τα πακέτα που έβαλα είναι:

- Ένα πακέτο για την φιλοξενία του κεντρικού controller (που υποστηρίζει τη διεκπεραίωση κάθε use case μέσω της σχετικής μεθόδου).
- Ένα πακέτο για την ανάκτηση από αποθηκευμένο αρχείο.
- Ένα πακέτο για την παραγωγή αναφορών.
- Ένα πακέτο για την φιλοξενία των domain classes.

Β. Στη front-end πλευρά, σας δίνεται ένα πακέτο για να στεγάσει τη αλληλεπίδραση του συστήματος με τον project manager. Εκεί υπάρχει αφενός μια απλή console-based διαπροσωπεία, ή ένα Graphical User Interface, καθώς και οι βοηθητικές κλάσεις. Αν υλοποιηθεί σωστά το back-end, το front-end παίζει μια χαρά.



Δεν είναι υποχρεωτικό να ακολουθήσετε τη δική μου σχεδίαση για το back-end. Είναι ενδεικτική για το πώς δομούνται «ανεξάρτητα» υποσυστήματα, που επικοινωνούν με οργανωμένο τρόπο, αλλά όχι υποχρεωτική (παρακάτω αναφέρονται οι περιορισμοί που σας τίθενται).

Περιορισμοί:
Εκτός από τις domain classes και το front-end, όλα τα άλλα πακέτα του back-end πρέπει να βγάζουν publicly προς το υπόλοιπο σύστημα ένα interface και ένα factory (θα πούμε τι είναι αυτά στο μάθημα).
Είναι υποχρεωτικό και απαράβατο, ΝΑ ΜΗΝ ΑΛΛΑΞΕΤΕ τα interfaces που σας δίνονται ως μέρος της εκφώνησης, 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. Έχετε δικαίωμα να υλοποιήσετε τις κλάσεις που λείπουν στο εσωτερικό των πακέτων με όποιο τρόπο θέλετε εσείς.
Είναι υποχρεωτικό και απαράβατο, ΝΑ ΚΑΤΑΣΚΕΥΑΣΤΟΥΝ UNIT TEST (Happy Day, Rainy Day) για όλα τα use cases που θα εντοπίσετε. Προφανώς tests πρέπει να υπάρχουν και για τις επί μέρους domain/bridge κλάσεις αλλά το βασικό είναι να ελεγχθούν τα use cases!
Χρησιμοποιήστε Junit 4 και όχι 5 (έχει πολύ περισσότερο υλικό στο διαδίκτυο).

Η κεντρική ιδέα. Σας δίνεται ένα πακέτο app (με υποπακέτα) με τους εξής 2 τρόπους λειτουργίας: ένα naïve client που πρακτικά λειτουργεί εν είδη τεστ, και (β) ένα graphical user interface πλήρως υλοποιημένο (ναι, για όσους θέλουν να το ψάξουν περαιτέρω). Τα κομμάτια αυτά όλα επικοινωνούν με το back-end μέσω μιας γέφυρας από 3 μέρη: (α) τον *AppController*, που κάνει τη δουλειά να υποδέχεται τα αιτήματα των χρηστών από τις προαναφερθείσες boundary classes (το GUI και το naïve client) τις οποίες να ζητά να διεκπεραιωθούν (β) από το back-end που εκπροσωπείται από το *IMainController* το οποίο είναι μια βιτρίνα που κρύβει όλο το back-end, (γ) και του οποίου οι μέθοδοι επιστρέφουν αντικείμενα από κλάσεις που περιέχονται στο πακέτο *dom2app*, τις οποίες ξέρουν και το front- και το back-end.
Οτιδήποτε βρίσκεται μετά το back-end, ξεκινώντας με την υλοποίηση του *IMainController* και το σχετικό factory (αυτό που σας εδόθη είναι το μόνο σημείο στον κώδικα που θέλει διόρθωση) είναι προς σχεδίαση, υλοποίηση και έλεγχο από εσάς.

Υλικό και οδηγίες

Υλικό. Όλο το υποστηρικτικό υλικό για το project θα βρίσκεται στο URL
http://www.cs.uoi.gr/~pvassil/courses/sw_dev/exercises/supportingMaterial/2022-2023/
όπου θα βρείτε 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/classes of the back-end/front-to-back-bridges,
(β) αρχεία με δεδομένα (υπο-φάκελος resources) για να κάνετε ελέγχους και για να τρέξετε το σύστημα που θα φτιάξετε, καθώς και το πώς αναμένεται να είναι το περιεχόμενο των reports,
(γ) ένας φάκελος libs με το σχετικό jar που μπορείτε να αξιοποιήσετε για το prefix search (δείτε στο Apache commons για το interface Trie και την κλάση PatriciaTrie).

Επίσης σας δίνεται και ένα **video** με επεξηγήσεις.

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!

Important ToDo. Επιπλέον όλων, 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_GanttManager`** (Στο Eclipse, σχεδόν τα πάντα μετονομάζονται με δεξί κλικ -> Refactor -> Rename). Στο αρχειάκι .project που συνοδεύει το Eclipse project θα πρέπει να μετονομάσετε το project επίσης.
- 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)

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).

[**03/10**]	Εκφώνηση
3 weeks	*Setup of Infrastructure*
[**23/10:: 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*
[15/11:: 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*
[**18/12:: 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

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!