



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**.

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 ώστε να είναι πλήρως λειτουργικό.

Η προγραμματιστική άσκηση έχει ως κύριο σκοπό να σας εκπαιδεύσει στα εξής:

- στη δόμηση της αρχιτεκτονικής ενός έργου λογισμικού και στην ανάθεση σαφών αρμοδιοτήτων σε κλάσεις και πακέτα,
- στην οργανωμένη επικοινωνία ενός front-end client με ένα back-end σύστημα μέσω ενός σταθερού Application Programming Interface (API),
- στη χρήση ενθυλάκωσης, χαμηλής σύζευξης, αφαιρέσεων (interfaces/abstract classes) και factories όπου αυτά είναι εύλογα,
- στην ενσωμάτωση εξωτερικών βιβλιοθηκών χωρίς η τεχνική τους λεπτομέρεια να διαχέεται σε όλο το σύστημα,
- στη σχεδίαση και υλοποίηση unit/integration tests για επιτυχείς και αποτυχημένες περιπτώσεις χρήσης.

Η βασική ιδέα της φετινής άσκησης είναι η κατασκευή ενός συστήματος διαχείρισης αναφορών (reports). Ένας αναλυτής δημιουργεί reports, προσθέτει σε αυτά διαφορετικά είδη περιεχομένου και μπορεί να τα εξαγάγει σε διαφορετικές μορφές, και συγκεκριμένα, PowerPoint (.pptx) και HTML (.html).

1. Περιγραφή του συστήματος και βασικές έννοιες

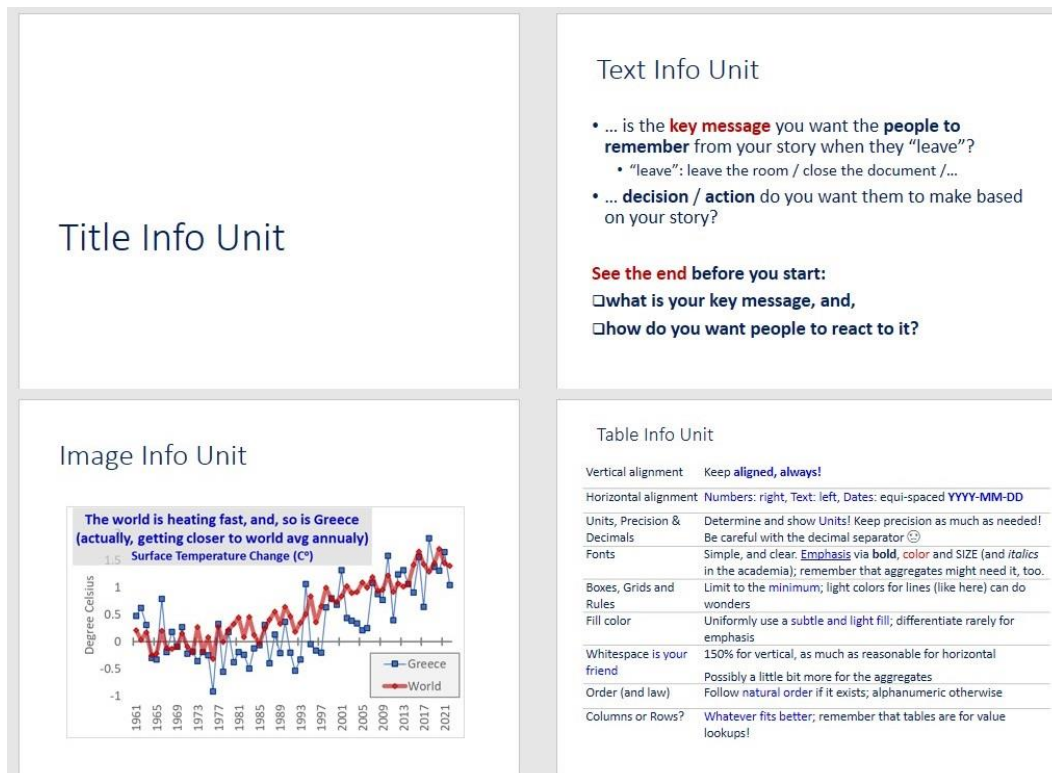
Το σύστημα διαχειρίζεται ένα σύνολο από reports. Κάθε report έχει τουλάχιστον ένα μοναδικό αναγνωριστικό, ένα όνομα και μια διατεταγμένη συλλογή από στοιχεία περιεχομένου. Στην εκφώνηση θα χρησιμοποιούμε τον όρο InfoUnit για ένα αυτοτελές στοιχείο περιεχομένου μιας αναφοράς. Στην περίπτωση που ένα report είναι ένα pptx αρχείο, ένα InfoUnit είναι ένα slide. Στην περίπτωση που ένα report είναι ένα κείμενο, ένα InfoUnit είναι μια ενότητα του κειμένου. Αν τυχόν ζητούνταν (δεν ζητείται), στην περίπτωση που ένα report ήταν ένα αρχείο xlsx, ένα InfoUnit θα ήταν ένα sheet. Εμείς σε κάθε περίπτωση θα αναπαριστούμε στο σύστημά μας reports και InfoUnits με αντικείμενα κλάσεων που θα σχεδιάσετε εσείς, και όταν ζητείται, το σύστημα θα παράγει το σχετικό αρχείο.

REPORT. Ένα report δημιουργείται αρχικά κενό και στη συνέχεια εμπλουτίζεται με InfoUnits. Η σειρά με την οποία προστίθενται τα στοιχεία πρέπει να διατηρείται.

INFOUNIT. Κάθε InfoUnit διαθέτει κοινά χαρακτηριστικά που είναι χρήσιμα ανεξάρτητα από το είδος του, όπως αναγνωριστικό, τίτλο, χρονόσημο δημιουργίας (creationDate) και λέξεις-κλειδιά (tags). Το creationDate αποδίδεται αυτόματα από το σύστημα κατά τη δημιουργία του InfoUnit και δεν παρέχεται από τον χρήστη.

Το σύστημα πρέπει να υποστηρίζει τουλάχιστον τα ακόλουθα είδη InfoUnit:

- **Title InfoUnit:** στοιχείο τίτλου/ενότητας, περιλαμβάνει μόνο ένα τίτλο, χρήσιμο για τη δομή μιας αναφοράς (π.χ., PowerPoint διαφάνεια που σηματοδοτεί την έναρξη νέας ενότητας, ή κεφαλίδα h1 σε ένα html κείμενο).
- **Text InfoUnit:** στοιχείο με τίτλο και κείμενο (π.χ., PowerPoint διαφάνεια με τίτλο και κείμενο, ή κεφαλίδα h2 και κείμενο σε ένα html αρχείο).
- **Image InfoUnit:** στοιχείο (π.χ., PowerPoint διαφάνεια ή τμήμα html αρχείου) που περιλαμβάνει τίτλο και αρχείο εικόνας.
- **Table InfoUnit:** στοιχείο που περιέχει τίτλο και δεδομένα σε μορφή πίνακα (επικεφαλίδες για τις στήλες στην πρώτη γραμμή και γραμμές τιμών).



Σχήμα 1. Μια PowerPoint παρουσίαση με τα 4 είδη InfoUnits (λίγο πιο όμορφα απ’ ότι θα βγάλουμε προγραμματιστικά στο project 😊)

2. Λειτουργικές απαιτήσεις

Στη συνέχεια παρατίθενται οι λειτουργικές απαιτήσεις του συστήματος. Η εξειδίκευση των επί μέρους πληροφοριών που πρέπει να παρασχεθούν για την κατασκευή report και InfoUnits βρίσκεται στον συμπληρωματικό κώδικα που σας δίνεται, και παραλείπεται εδώ χάριν συντομίας.

Διαχείριση reports

ΔΗΜΙΟΥΡΓΙΑ REPORT. Ο χρήστης δίνει τα απαραίτητα στοιχεία και το σύστημα δημιουργεί ένα νέο κενό report. Το νέο report προστίθεται στη λίστα reports που διατηρεί το σύστημα. Το αποτέλεσμα της πράξης πρέπει να δηλώνει αν η δημιουργία ολοκληρώθηκε επιτυχώς ή όχι.

ΛΙΣΤΑ REPORTS. Ο χρήστης μπορεί να ζητήσει τη λίστα των reports που υπάρχουν στο σύστημα. Το back-end επιστρέφει τις αναγκαίες πληροφορίες για την παρουσίασή τους από το front-end.

ΠΡΟΒΟΛΗ ΔΟΜΗΣ REPORT. Δεδομένου ενός report, το back-end επιστρέφει συνοπτική πληροφορία για τα InfoUnits που περιέχει, στη σειρά που έχουν προστεθεί, σαν συμβολοσειρά με id και σύντομη περιγραφή (στη διακριτική σας ευχέρεια ποια θα είναι ανά τύπο InfoUnit).

Διαχείριση στοιχείων περιεχομένου

ΠΡΟΣΘΗΚΗ TITLE INFO UNIT. Ο χρήστης προσδιορίζει το report και τα στοιχεία του νέου Title InfoUnit. Το στοιχείο προστίθεται στο τέλος της τρέχουσας διάταξης.

ΠΡΟΣΘΗΚΗ TEXT INFO UNIT. Αντίστοιχα, προστίθεται ένα Text InfoUnit με το κείμενο και τα υπόλοιπα απαιτούμενα χαρακτηριστικά του.

ΠΡΟΣΘΗΚΗ IMAGE INFO UNIT. Προστίθεται ένα Image InfoUnit που αναφέρεται σε υπαρκτό αρχείο εικόνας. Το σύστημα πρέπει να χειρίζεται με σαφή τρόπο μη έγκυρα ή ανύπαρκτα αρχεία.

ΠΡΟΣΘΗΚΗ TABLE INFO UNIT. Προστίθεται ένα Table InfoUnit με επικεφαλίδες στηλών και δεδομένα συμβατής μορφής.

ΔΙΑΓΡΑΦΗ INFO UNIT. Ο χρήστης μπορεί να αφαιρέσει ένα συγκεκριμένο InfoUnit από ένα report.

Εξαγωγή αναφορών

ΕΞΑΓΩΓΗ ΣΕ POWERPOINT. Το σύστημα δημιουργεί αρχείο .pptx. Κάθε InfoUnit μετατρέπεται σε κατάλληλο οπτικό περιεχόμενο: τίτλος/κείμενο, πίνακας ή εικόνα. Για την τεχνική παραγωγή του αρχείου θα χρησιμοποιηθεί η βιβλιοθήκη Apache POI (βλ. παρακάτω).

ΕΞΑΓΩΓΗ ΣΕ HTML. Το σύστημα δημιουργεί αρχείο .html που παρουσιάζει με αντίστοιχο τρόπο τα στοιχεία του report. Τα απαραίτητα αρχεία εικόνων πρέπει να αντιγραφούν από το σύστημα στο

φάκελο που θα βρίσκεται το html αρχείο (ή σε υποφάκελο αυτού) ώστε να είναι διαθέσιμα με τρόπο που να επιτρέπει τη μεταφορά του φακέλου εξαγωγής ως αυτοτελούς report.

ΕΞΑΓΩΓΗ ΜΕ ΦΙΛΤΡΑΡΙΣΜΑ ΠΕΡΙΕΧΟΜΕΝΟΥ ΜΕ ΒΑΣΗ TAGS. Για όλους τους τύπους αρχείου εξαγωγής, κατά την εξαγωγή με φίλτρο, ο αναλυτής μπορεί να ζητήσει να εξαχθούν μόνο συγκεκριμένα στοιχεία ενός προϋπάρχοντος Report εφαρμόζοντας ένα φίλτρο από μια λίστα από tags. Κάθε InfoUnit που περιέχει έστω και ένα από αυτά τα tags, θα συμπεριλαμβάνεται στο αποτέλεσμα. Το back-end, εσωτερικά καλείται να φτιάξει ένα νέο προσωρινό report και να το γράψει στο σχετικό αρχείο. Το προσωρινό report γράφεται μεν ως αρχείο, αλλά δεν συμπεριλαμβάνεται στη λίστα των reports του συστήματος.

Κατά την εξαγωγή, για κάθε InfoUnit καταγράφουμε τη μετα-πληροφορία, δλδ., την ημερομηνία και τις λέξεις κλειδιά ως εξής:

- PowerPoint: Στα slide notes καταγράφονται οι μεταπληροφορίες ως απλό κείμενο
- HTML: σε ένα μικρό πίνακα με ένα και μόνο κελί, ως οπτικά ξέχωρο κείμενο (π.χ., italics και μικρότερα γράμματα)

3. Προδιαγραφές του Συστήματος και Αρχιτεκτονική (Front-end, Back-end και παρεχόμενο API)

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 αναλυτή που χρησιμοποιεί το σύστημα 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, τη διαχείριση των αντικειμένων που αναπαριστούν τις πληροφορίες κατά τη διάρκεια της εκτέλεσής, τους ελέγχους, και την παραγωγή του τελικού output.

API: Η Γέφυρα Μεταξύ Front-End & Back-End

Η γέφυρα ανάμεσα στις δύο πλευρές είναι ένα πακέτο input (με υποπακέτα) με interfaces & κλάσεις που επιτρέπουν στο front-end να επικοινωνεί με το back-end και **σας δίνεται έτοιμο**. Το πακέτο **input** περιλαμβάνει τα εξής υποπακέτα, έκαστο με τα παρατιθέμενα περιεχόμενα:

- # πακέτο **input.port**: περιέχει το interface **IReportManager**, που λειτουργεί ως συμβόλαιο μεταξύ front-end & back-end (περιορίζει το τι μπορεί να ζητηθεί από το back-end σε αυτές και μόνο τις λειτουργίες που βρίσκονται στο interface)
- # πακέτο **input.init**: περιέχει την κλάση **ReportManagerFactory**, την οποία πρέπει να ολοκληρώσετε ώστε να επιστρέφει τη δική σας concrete υλοποίηση του **IReportManager**, τοποθετημένη εκτός του **input package**.
- # πακέτο **input.request**: περιέχει τις κλάσεις που γεννούν τα αντικείμενα αιτήματος που στέλνει το front-end. Πρακτικά, κάθε αίτημα λειτουργικότητας προς το back-end έχει παραμέτρους, τις οποίες περιλαμβάνει η σχετική κλάση στο πακέτο **input.request**.
- # πακέτο **input.result**: περιέχει τις κλάσεις που γεννούν τα αντικείμενα αποτελέσματος στα αιτήματα που γίνονται, με τις πληροφορίες του αποτελέσματος, καθώς και τα error codes που ενδεχομένως να έχουν προκύψει κατά την εκτέλεση.

Είναι υποχρεωτικό και απαραίτητο ΝΑ ΜΗΝ ΑΛΛΑΞΕΤΕ τα interfaces, requests, results και public enumerations του input.* που σας δίνονται. Η μοναδική αναμενόμενη αλλαγή στο **input.init.ReportManagerFactory** είναι να αντικαταστήσετε την placeholder εντολή **return null**; με τη δημιουργία της concrete υλοποίησής σας.

Front-end

Το front-end είναι υπεύθυνο για τη διάδραση με τον αναλυτή που χρησιμοποιεί το σύστημα. Είναι σημαντικό να επισημάνουμε τα εξής:

1. **Σας δίνεται έτοιμο ένα απλό console-based client πρόγραμμα**, με μια απλή **main()** που επιδεικνύει τον τρόπο χρήσης του API. Αν υλοποιήσετε σωστά το πρόγραμμα, ο client θα τρέχει επίσης σωστά. Αλλά **το πιο σημαντικό είναι να δείτε πώς χρησιμοποιείται ένα API στην πράξη από ένα front-end**.
2. **Ο client δεν πρέπει να γνωρίζει concrete κλάσεις του back-end**. Αν κατασκευάσετε οποιαδήποτε επιπλέον client κλάση (ή ίσως και κάποιο υποσύστημα) αυτό οφείλει επίσης να επικοινωνεί με το back-end αποκλειστικά μέσω του παρεχόμενου **input API**.

Back-end

Το εσωτερικό back-end είναι υπεύθυνο για την εκτέλεση των use cases, τη διαχείριση των domain objects, τους ελέγχους και την εξαγωγή. **Το back-end θα είναι αποτέλεσμα δικής σας σχεδίασης.** Εσείς αποφασίζετε, τεκμηριώνετε και υλοποιείτε:

- τις domain/model κλάσεις και τις σχέσεις μεταξύ τους,
- τις κλάσεις που αναλαμβάνουν τη διεκπεραίωση των use cases,
- αν και πώς θα απομονώσετε την αποθήκευση των reports,
- αν και πώς θα ορίσετε κοινή αφαίρεση για τους διαφορετικούς exporters,
- τα factories και τις εξαρτήσεις που θεωρείτε ότι βελτιώνουν τη σχεδίαση,
- την κατανομή όλων των παραπάνω σε πακέτα.

Στη διάρκεια του μαθήματος θα συζητηθούν ενδεικτικές αρχιτεκτονικές και σχεδιαστικές επιλογές. **Σε επίπεδο αρχιτεκτονικής του κώδικα, δεν απαιτείται να αντιγράψετε κάποια συγκεκριμένη αρχιτεκτονική αναφοράς. Πρέπει όμως να ξέρετε τι σχεδιαστικές επιλογές κάνετε και γιατί. Αυτό είναι και το πιο βασικό εφόδιο που θα αποκτήσετε από το μάθημα!**

Σε επίπεδο επί μέρους σχεδιάσεων, καλό είναι να αξιοποιήσετε τις δυνατότητες του αντικειμενοστρεφούς προγραμματισμού. Για παράδειγμα, η σχεδίασή σας μπορεί να εκμεταλλεύεται κατάλληλα τις δυνατότητες αντικειμενοστρεφούς σχεδίασης (π.χ. interfaces για συμβόλαιο συμπεριφοράς, αφηρημένες κλάσεις για μοντελοποίηση της κοινής κατάστασης διαφορετικών επιμέρους ειδών και εξειδικεύσεις για τα επιμέρους είδη, κοκ), χωρίς όμως να εισάγει περιττή πολυπλοκότητα.

Εξωτερικές βιβλιοθήκες και παραδείγματα POI / HTML

Το σύστημα, όπως προδιαγράφη πρέπει να μπορεί να εξάγει ένα report σε μορφή HTML και σε μορφή Powerpoint. Εσείς θα πρέπει (α) να διαχειριστείτε report και InfoUnits σαν απλά αντικείμενα Java και (β) να κουμπώσετε την υλοποίησή σας αυτή σε ένα σύνολο μετατροπών που μετατρέπουν τα αντικείμενα αυτά της java σε σελίδες html και παρουσιάσεις pptx.

- Τα αρχεία html είναι απλά αρχεία κειμένου. Δεν απαιτείται καμία ειδική βιβλιοθήκη για να τα παράξετε, παρά μόνο αυτές που παρέχει το JDK από μόνο του.
- Για την παραγωγή Powerpoint παρουσιάσεων, θα χρησιμοποιήσουμε τη βιβλιοθήκη Apache POI η οποία χρησιμοποιείται για να μπορεί κανείς να κατασκευάσει προγραμματιστικά αντικείμενα του MS Office.

Επίσης, για τη χρησιμοποίηση της σχετικής βιβλιοθήκης (όπως και όλων των εξωτερικών βιβλιοθηκών σε ένα σύστημα Java) υπάρχουν δύο τρόποι

- Με ενσωμάτωση των βιβλιοθηκών σαν αρχεία .jar σε ένα φάκελο lib του συστήματος που φτιάχνετε, και δήλωσή τους
- Με χρήση Maven, που είναι ο πλέον διαδεδομένος πλέον τρόπος ενσωμάτωσης εξωτερικών βιβλιοθηκών.

Εσείς θα διαλέξετε ένα από τους δύο τρόπους που σας βολεύει καλύτερα. Μην συνδυάζετε τις δύο μεθόδους στο ίδιο project χωρίς λόγο.

Στο υποστηρικτικό υλικό παρέχονται και σύντομες οδηγίες εγκατάστασης για Eclipse.

ΠΑΡΑΔΕΙΓΜΑΤΑ. Για να μπορέσετε να εργαστείτε με τα δύο είδη παραγωγής output, και να μη χρειαστεί να ανακαλύψετε μόνοι σας τις τεχνικές λεπτομέρειες της Apache POI και του Maven, σας παρέχονται δύο μικρά ανεξάρτητα projects-**παραδείγματα**:

- 01_poi_with_local_libraries: απλό Eclipse/Java project που δείχνει πώς προστίθενται εξωτερικά JARs στο Build Path και πώς δημιουργείται PPTX με text, table και image slides.
- 02_poi_with_maven: Maven project με αντίστοιχο παράδειγμα POI και ένα μικρό HTML export example.

Τα παραδείγματα δείχνουν επίσης πώς μπορούν να αποθηκευθούν creationDate και tags στα notes μιας διαφάνειας και σε μικρό metadata block ενός HTML αρχείου.

Τα projects αυτά είναι τεχνικά παραδείγματα και ΟΧΙ μέρος της αρχιτεκτονικής της εφαρμογής σας. Μπορείτε να μελετήσετε και να επαναχρησιμοποιήσετε τον σχετικό τεχνικό κώδικα, αλλά η ενσωμάτωσή του ΣΩΣΤΑ (όχι αυτούσιου) σε μεθόδους κλάσεων στο δικό σας σύστημα είναι ένα βασικό αντικείμενο της παρούσας εργασίας.

Αποτελέσματα πράξεων και σφάλματα

Οι μέθοδοι του δημόσιου API δεν πρέπει να βασίζονται σε μηνύματα που εκτυπώνονται απευθείας από το back-end. Το front-end πρέπει να λαμβάνει αντικείμενα αποτελέσματος, ώστε να μπορεί να αποφασίσει πώς θα παρουσιάσει την επιτυχία ή αποτυχία.

Ενδεικτικές περιπτώσεις αποτυχίας που πρέπει να αντιμετωπίζονται:

- ανύπαρκτο report ή InfoUnit,
- διπλό/μη αποδεκτό όνομα report, όπου αυτό απαγορεύεται από το API,
- ελλιπή ή μη έγκυρα δεδομένα για νέο στοιχείο,
- ασυνεπής πίνακας (π.χ. γραμμές διαφορετικού πλήθους στηλών),
- μη έγκυρη εικόνα ή αποτυχία ανάγνωσης/εγγραφής αρχείου,
- αποτυχία εξαγωγής σε PPTX ή HTML.

Τα error codes που εκτίθενται από το παρεχόμενο API είναι μέρος του συμβολαίου και δεν πρέπει να αλλάξουν. Τεχνικές exceptions που ενδέχεται να προκύψουν από βιβλιοθήκες ή I/O δεν πρέπει να διαρρέουν αυθαίρετα μέχρι το front-end· μεταφράζονται σε κατάλληλο αποτέλεσμα της εφαρμογής, όπου χρειάζεται.

Έλεγχος

Είναι υποχρεωτικό να κατασκευαστούν tests για τα βασικά use cases. Τα tests αποτελούν μέρος του λογισμικού και όχι προαιρετικό συνοδευτικό υλικό.

Συγκεκριμένα αναμένονται JUnit tests:

- Happy Day tests: ελέγχουν ότι κάθε βασικό use case λειτουργεί σωστά με έγκυρα δεδομένα.
- Rainy Day / failure tests: ελέγχουν αντιπροσωπευτικές αποτυχίες, π.χ. ανύπαρκτο report/InfoUnit, μη έγκυρα δεδομένα, λανθασμένος πίνακας, ανύπαρκτη εικόνα, μη αποδεκτό export request.
- Unit tests για σημαντικές domain/service κλάσεις και βασικούς κανόνες.
- Τουλάχιστον ένας συνολικός έλεγχος που δημιουργεί report, προσθέτει διαφορετικά InfoUnits και ελέγχει την εξαγωγή του, και καθολικά και με φίλτρο.

Το παρεχόμενο project έχει σκόπιμα κενό τον φάκελο src/test/java. Το ποια tests χρειάζονται αποτελεί μέρος της άσκησης. Χρησιμοποιήστε JUnit 5.

4. Υλικό και οδηγίες

Όλο το υποστηρικτικό υλικό θα αναρτηθεί στον χώρο του μαθήματος στο URL

http://www.cs.uoi.gr/~pvassil/courses/sw_dev/exercises/supportingMaterial/2026-2027/

Το βασικό υλικό περιλαμβάνει:

- XXXX_YYYY_ZZZZ_reportmanager.zip: ο σκελετός της εργασίας, με το input API, client, resources και build configuration,

Το βοηθητικό υλικό περιλαμβάνει:

- 01_poi_with_local_libraries.zip: παράδειγμα POI με τοπικές βιβλιοθήκες,
- 02_poi_with_maven.zip: παράδειγμα POI/HTML με Maven,
- σύντομο κείμενο οδηγιών εγκατάστασης των δύο παραδειγμάτων στο Eclipse

Για την αναφορά:

- υπάρχει το υποχρεωτικό υπόδειγμα της τελικής αναφοράς στο URL http://www.cs.uoi.gr/~pvassil/courses/sw_dev/exercises/TemplateFinalReport.zip

ΠΡΕΠΕΙ ΝΑ ΑΚΟΛΟΥΘΗΣΕΤΕ ΥΠΟΧΡΕΩΤΙΚΑ ΤΟ ΥΠΟΔΕΙΓΜΑ ΤΕΛΙΚΗΣ ΑΝΑΦΟΡΑΣ ΠΟΥ ΣΑΣ ΔΙΝΕΤΑΙ.

Υποχρεωτικές ενέργειες από σας

- Δουλεύουμε με Eclipse και Java 14, ώστε να είναι δυνατός ο μαζικός έλεγχος των υποβολών από τους διορθωτές.
- Η ομάδα αποτελείται από έναν έως τρεις φοιτητές, καθένας με τετραψήφιο Αριθμό Μητρώου.
- Μετονομάστε το δοθέν project/folder XXXX_YYYY_ZZZZ_reportmanager αντικαθιστώντας τα XXXX, YYYY, ZZZZ με τους AM των μελών της ομάδας, σε αύξουσα σειρά (ώστε, αφού τα κάνετε turnin, να ξέρουμε ποιανού είναι κάθε project). Π.χ., αν στην ομάδα μετέχουν οι φοιτητές με AM 9345, 9344, 9567, το project υποχρεωτικά πρέπει να ονομάζεται ως: 9344_9345_9567_reportmanager (Στο Eclipse, σχεδόν τα πάντα μετονομάζονται με δεξί κλικ -> Refactor -> Rename). Προφανώς, λιγότεροι από 3 φοιτητές βάζουν τα AM τους χωρίς πτπ άλλο π.χ., για τους 2345, 6789 βγαίνει το 2345_6789_reportmanager. Στο αρχιδιακί .project που συνοδεύει το Eclipse project θα πρέπει να μετονομάσετε το project επίσης.

- Αν χρησιμοποιείτε Maven, ενημερώστε επίσης το groupId στο pom.xml ώστε να αντιστοιχεί στην ομάδα σας (π.χ. 1234-5678-9012).
- Προσθέστε Readme.txt με (α) ονόματα και AM των μελών της ομάδας και (β) οτιδήποτε απαιτείται για να τρέξει το πρόγραμμα χωρίς προβλήματα κατά τον έλεγχο.
- Μην προσθέτετε absolute paths από τον προσωπικό σας υπολογιστή. Χρησιμοποιήστε resources και σχετικά paths ώστε το project να μπορεί να μεταφερθεί και να εκτελεστεί αλλού.
- Μην τροποποιείτε το παρεχόμενο input.* API. Οποιαδήποτε concrete implementation του IReportManager πρέπει να βρίσκεται εκτός του input package.

5. Παραδοτέα

Θα κάνετε turnin δύο αρχεία:

- ένα ενιαίο .zip με ολόκληρο το Eclipse/Maven project και όλον τον πηγαίο κώδικα, resources και tests,
- την τελική αναφορά σε μορφή PDF, σύμφωνα με το υποχρεωτικό υπόδειγμα.

Η τελική αναφορά πρέπει να περιλαμβάνει τουλάχιστον:

- περιγραφή των use cases και των βασικών απαιτήσεων,
- σχεδίαση πακέτων και αιτιολόγηση των κύριων dependency decisions,
- class diagram της τελικής υλοποίησης,
- σύντομη περιγραφή των βασικών design choices (interfaces, abstract classes, factories, encapsulation, coupling),
- περιγραφή των tests και σύνδεσή τους με τα use cases (traceability),
- οδηγίες εκτέλεσης, εφόσον απαιτείται κάτι πέρα από τα συνηθισμένα βήματα Eclipse/Maven.

6. Τι αξιολογείται

Η εργασία δεν αξιολογείται μόνο με βάση το αν “τρέχει”. Ιδιαίτερη σημασία έχουν:

- η ορθότητα και πληρότητα των use cases,
- η ποιότητα της αρχιτεκτονικής και η απουσία αδικαιολόγητων κυκλικών εξαρτήσεων μεταξύ πακέτων,
- η ενθυλάκωση και η σωστή ανάθεση ευθυνών,
- η χρήση interfaces/abstract classes/factories εκεί όπου προσφέρουν πραγματικό όφελος, και όχι μηχανικά,
- η απομόνωση των τεχνικών λεπτομερειών POI/HTML από το υπόλοιπο σύστημα,
- η ποιότητα και ουσιαστικότητα των tests, ιδιαίτερα των failure cases,
- η συνέπεια μεταξύ σχεδίασης, υλοποίησης και τελικής αναφοράς.

7. Χρονοδιάγραμμα

Αφού φτιάξετε μια ομάδα συνεργατών, η ομάδα πρέπει να εγγραφεί με τον τρόπο που θα ανακοινωθεί στο site του μαθήματος. Αυτό είναι βασικό για να μη χαθεί κάποια υποβολή από τα scripts που τρέχουμε για να οργανώσουμε τη διόρθωση. Ομάδες που δεν έχουν δηλωθεί έγκαιρα ενδέχεται να αντιμετωπίσουν πρόβλημα στη βαθμολόγηση. Οπότε: **γραφτείτε εγκαίρως!**

Στη διάρκεια του εξαμήνου:

- Σίγουρα **θα δοθούν περαιτέρω εξηγήσεις στην εκφώνηση**. Αν τυχόν χρειαστεί, ενδέχεται να δοθούν τροποποιήσεις ή μικρές διορθώσεις της εκφώνησης. Η τελευταία έκδοση του υλικού στο site του μαθήματος υπερισχύει προηγούμενων αντιγράφων.
- **Πρέπει επίσης να εγγράψετε την ομάδα σας** σε μία φόρμα εγγραφής. Δείτε το web site για τη σχετική ανακοίνωση. Ομάδες που δεν εγγραφούν κινδυνεύουν να μην βαθμολογηθούν.

Η προθεσμία παράδοσης είναι 2026-12-20, ώρα 23:59. Η προθεσμία είναι ανελαστική. Δεν θα ξεπεράσουμε το όριο των Χριστουγέννων, εκτός αν προκύψει εξαιρετική και επίσημα ανακοινωμένη ανάγκη. Η εμπειρία έχει δείξει ότι στις γιορτές οι ομάδες αποσυντονίζονται σημαντικά· στόχος είναι να ολοκληρώσετε το project πριν από αυτές.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ!