ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2024-2025

ΟΜΑΔΑ AM1-AM2-AM3

ΦΟΙΤΗΤΗΣ 1, AM:AM1

ΦΟΙΤΗΤΗΣ 2, AM:AM2

ΦΟΙΤΗΤΗΣ 3, AM:AM3

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΔΕΚΕΜΒΡΙΟΣ 2020

ΙΣΤΟΡΙΚΟ ΕΚΔΟΣΕΩΝ ΤΗΣ ΠΑΡΟΥΣΑΣ ΑΝΑΦΟΡΑΣ

Ημερομηνία	Έκδοση	Περιγραφή	Συγγραφείς
yyyy/mm/dd	v.01	Οργάνωση απαιτήσεων σε use cases	ΧΧ,ΥΥ,ΖΖ
yyyy/mm/dd	v.02	Αρχική σχεδίαση κλάσεων και ελέγχων	ΧΧ,ΥΥ,ΖΖ
yyyy/mm/dd	…	Διορθώσεις στις uses cases, επεκτάσεις στη σχεδίαση κλάσεων και ελέγχων	ΥΥ,ΖΖ (ο ΧΧ αποχώρησε)
yyyy/mm/dd	…	ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ	ΥΥ

Στην τρέχουσα σελίδα, με **γκρι, αχνά γράμματα** παραθέτω οδηγίες, επεξηγήσεις και σχόλια. **Στη δική σας αναφορά, αυτό το κομμάτι προφανώς θα πρέπει να το σβήσετε.**

Στο υπόλοιπο κείμενο, για να μη περιπλέξω το στυλ περαιτέρω**, οι οδηγίες παρατίθενται σε απλό Βασικό/Normal στυλ. Εσείς εκεί πρέπει να βάλετε το δικό σας κείμενο στη θέση του δικού μου.**

ΟΔΗΓΙΕΣ

Αναμένεται να τηρήσετε το παρόν πρότυπο ΠΙΣΤΑ. Όχι επειδή είναι το απαύγασμα της καλαισθησίας ή της λειτουργικότητας, αλλά επειδή πρέπει να μάθετε να τηρείτε πρότυπα με πειθαρχία και συνέπεια. Για το λόγο αυτό, καλείσθε να ΜΕΙΝΕΤΕ ΠΙΣΤΑ ΣΤΟ ΣΤΥΛ ΤΟΥ ΚΕΙΜΕΝΟΥ! Είναι ευκαιρία επίσης, να μάθετε να χρησιμοποιείτε με ΕΥΧΕΡΕΙΑ επεξ. κειμένου.

Κατά τα λοιπά:

1. Συμπληρώστε με τα σωστά στοιχεία το εξώφυλλο και το header, όπου υπάρχουν κόκκινα γράμματα. Αν δεν σας έχει δοθεί α/α ομάδας, είναι το concatenation των ΑΜ σας, με παύλες ανάμεσα, ταξινομημένα με αύξουσα σειρά.
2. Ο παραπάνω πίνακας «Ιστορικό Εκδόσεων» συμπληρώνεται κάθε φορά που αλλάζετε κάτι στην αναφορά σας. Οι καταχωρήσεις που υπάρχουν ήδη είναι απλά ενδεικτικές.
3. Στις Use Cases συμπληρώνετε ΟΛΕΣ τις Use Cases που έχετε εξάγει (εδώ παρατίθεται μόνο μία, ενδεικτικά). Το στυλ είναι από το υπόδειγμα για use cases που χρησιμοποιείται στο μάθημα.
4. Δώστε ιδιαίτερη σημασία στους ελέγχους, καθώς είναι από τις λίγες φορές στην εκπαίδευσή σας που εξετάζεστε για την απάντηση που δώσατε στο ερώτημα «Πώς θα επιβεβαιώσω ότι ο κώδικάς μου κάνει αυτό που πρέπει?» (θυμηθείτε ότι στις συνεντεύξεις για δουλειά, ερωτήσεις επί του testing είναι από τις πιο κλασικές επιλογές των interviewers).
5. Συμπληρώστε τα διαγράμματα με τις σχετικές εικόνες. Προσοχή: τα σχήματα πρέπει να είναι ΕΥΔΙΑΚΡΙΤΑ. Ο σκοπός των διαγραμμάτων είναι να μπορούν να μεταδώσουν ΣΤΟΥΣ ΑΛΛΟΥΣ τις σχεδιαστικές σας αποφάσεις.

1 ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ – USE CASES

Στην παρούσα ενότητα, παρατίθενται οι περιγραφές των use cases με βάση τις καταγεγραμμένες απαιτήσεις.

ΕΜΦΑΝΙΣΕΠΡΟΪΟΝΤΑΚΑΤΑΣΤΗΜΑΤΟΣ

ID: UC 1

DESCRIPTION AND GOAL

Η use case «ΕμφάνισεΠροϊόνταΚαταστήματος» εμφανίζει τα διαθέσιμα προϊόντα του καταστήματος στην οθόνη.

ACTORS (ESP. PRIMARY ACTOR)

Ο πελάτης του ηλ. καταστήματος.

PRECONDITIONS

Πρέπει να έχουν φορτωθεί και να υπάρχουν διαθέσιμα προϊόντα στο κατάστημα.

BASIC FLOW

1. Το use case ξεκινάει όταν ο πελάτης επιλέξει από μενού την επιλογή «Εμφάνιση προϊόντων».
2. Το σύστημα εμφανίζει τα προϊόντα στην οθόνη.

EXTENSIONS / VARIATIONS

1. Στην περίπτωση κατά την οποία δεν υπάρχουν προϊόντα στο κατάστημα εμφανίζεται ένα μήνυμα που ενημερώνει ότι δεν υπάρχουν διαθέσιμα προϊόντα.

POST CONDITIONS

-

2 ΣΧΕΔΙΑΣΗ ΕΛΕΓΧΩΝ

Οι έλεγχοι που σχεδιάσθηκαν και εντάχθηκαν στην υλοποίηση περιγράφονται παρακάτω. Εδώ, ως υπόδειγμα: το project με την διάσπαση χρονοσειράς σε φάσεις.

2.1 ΕΛΕΓΧΟΣ USE CASES VIA SYSTEM TESTS

Στην αρχική σχεδιαστική φάση, αρκεί να συμπληρώσετε την λεκτική περιγραφή με τις OREOS προδιαγραφές. Στην τελική φάση, συμπληρώστε και τις λεπτομέρειες σε σχέση με τις εμπλεκόμενες μεθόδους και το setup input, output, pre-post conditions, ...

2.1.1 USE CASE UC1: LOAD DATA

Test cases

Description	*ON*	*any context*
	RECEIVING	*Request to parse a specific txt file with a valid timeline*
	ENSURE	*That the System*
	OUTPUTS	*A timeline with the correct size and no offending <time,value> pairs*
	SUCH THAT	*state is intact*

ID	T1_V0_01	HappyDayScenario for SimpleTextParser.parse()
Pre-cond.		No specific precond constructed
Object creation		New `SimpleTextParser`
Input		input_test.txt, a small file with less than 10 entries, all valid
Method To test		`SimpleTextParser.parse(filename)`
Expected Output		a timeline with the same #entries as the contexts of input_test.txt and no offending values `assert:` all expected entries are equal to actual entries
Post-cond.		No state properties tested

ID	T1_V0_02	HappyDayScenario for MainEngine.setTimeLine()
		...identical setup with T1_V0_01
Method To test		`MainEngine.setTimeLine(filename)`

Involved methods

```
MainEngine.setTimeLine(),

IParser.parse --> SimpleTextParser.parse(filename)
```

Not designed yet: T1_V1: missing file, T1_V2: invalid values in input file

2.1.2 USE CASE UC2: ANALYZE TIMELINE (AND PRODUCE PHASES)

Test cases

Description	*ON*	*A time line having being loaded*
	RECEIVING	*Request to analyze a valid timeline into phases*
	ENSURE	*That the System*
	OUTPUTS	*a set of phases*
	SUCH THAT	*state is intact*

ID	T2_V0_01	HappyDayScenario for NaiveAnalyser.producePhases()
Pre-cond.		Load input_test.txt, a small file with less than 10 entries, all valid, for o(5) phases, and produce timeline
Object creation		New `SimpleTextParser, MainEngine, NaiveAnalyser`
Input		the abovementioned timeline
Method To test		`NaiveAnalyser.producePhasesFromTimeLine (TimeLine)`
Expected Output		A correct #phases, with the correct points inside `assert:` all expected phases are equal to actual phases
Post-cond.		No state properties tested

ID	T2_V0_02	HappyDayScenario for MainEngine.producePhases()
		…identical setup with T2_V0_01
Method To test		`MainEngine.producePhases()`

Involved methods

```
MainEngine.producePhases()

AnalyserFactory.createAnalyzer()

IAnalyzer --> NaiveAnalyser.producePhasesFromTimeLine (TimeLine)
```

Not designed yet: T2_V1: null time line, T2_V2: timeline with only one phase

2.1.3 USE CASE UC3: VISUALIZE TIMELINE

Test cases

Description	*ON*	*A time line having being loaded*
	RECEIVING	*Request to visualize a valid timeline via a specific visualizer*
	ENSURE	*That the System*
	OUTPUTS	*An appropriate visualization*
	SUCH THAT	*state is intact*

ID	T3_V0_01	HappyDayScenario for MainEngine.visualize()
Pre-cond.		Load input_test.txt, a small file with less than 10 entries, all valid, for o(5) phases, and produce timeline
Object creation		New `MainEngine`
Input		the abovementioned timeline, "`HtmlVisualizer`" as the tested visualizer
Method To test		`MainEngine.setVisualizer(String)` `MainEngine.visualize()`
Expected Output		A correct visualization, expressed as a 2D raster of chars `assert:` all raster cells are equal to actual cells
Post-cond.		No state properties tested

ID	T3_V0_02	HappyDayScenario for MainEngine.visualize()
		...identical setup with T2_V0_01
Input		"`ConsoleVisualizer`" as the tested visualizer

Involved methods

```
MainEngine.visualize()
```

2.1.4 USE CASE UC4: VISUALIZE PHASES (TO CONSOLE)

Test cases

Description	*ON*	*A time line having being loaded*
	RECEIVING	*Request to visualize the phases of the timeline*
	ENSURE	*That the System*
	OUTPUTS	*An appropriate visualization for the phases*
	SUCH THAT	*state is intact*

ID	T4_V0	HappyDayScenario for MainEngine.reportPhases()
Pre-cond.		Load input_test.txt, a small file with less than 10 entries, all valid, for o(5) phases, produce timeline and analyze it to phases
Object creation		New `MainEngine`
Input		the abovementioned timeline and its phases
Method To test		`MainEngine. reportPhases()`
Expected Output		A correct visualization (approximation: the size of the descriptions produced is the same with the number of phases) `assert:` expected description size equal to actual size
Post-cond.		No state properties tested

Involved methods

```
MainEngine.reportPhases()

NaiveAnalyser.reportToConsole()

Phase.consoleVerticalReport()
```

2.2 TRACEABILITY MATRIX

Η αντιστοίχιση use cases σε id's φαίνεται στον Πίνακα 1:

UC1	Load Data
UC2	Analyze TimeLine
UC3	Present TimeLine
UC4	Present Phases

Πίνακας 1 Σύνοψη use cases και των id's τους

Ο Πίνακας 2 είναι ο traceability matrix για τους ελέγχους μας. Στη συνέχεια, οι έλεγχοι επεξηγούνται πιο αναλυτικά.

	UC1	UC2	UC3	UC4
T1_V0_01	X			
T1_V0_02	X			
T2_V0_01		X		
T2_V0_02		X		
T3_V0_01			X	
T3_V0_02			X	
T4_V0				X

Πίνακας 2 Traceability matrix between use cases and tests

2.3 ΕΚΚΡΕΜΟΤΗΤΕΣ (TODO)

Εκκρεμούν μη υλοποιημένοι έλεγχοι ως ακολούθως *(αν υπάρχουν εκκρεμότητες, παραθέστε την TODO λίστα ελέγχων που πρέπει να ετοιμαστούν)*

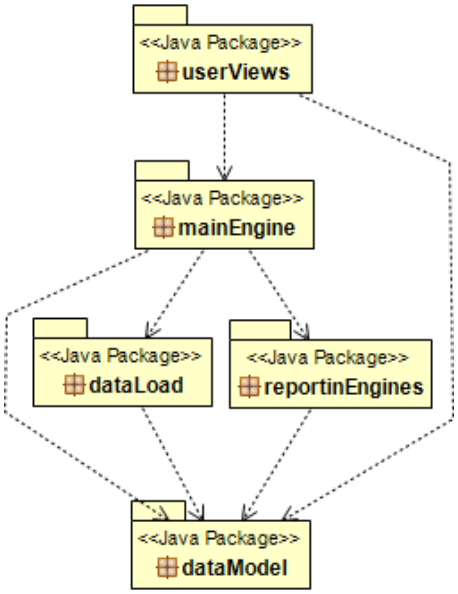
1. Unit tests are missing for several classes, both at the model and at the business logic level, specifically, class XXX, YYY, ZZZ

3 ΣΧΕΔΙΑΣΗ ΛΟΓΙΣΜΙΚΟΥ

3.1 ΔΙΑΓΡΑΜΜΑΤΑ ΠΑΚΕΤΩΝ / ΥΠΟΣΥΣΤΗΜΑΤΩΝ

Η ανάλυση του κώδικα σε υποσυστήματα και πακέτα έχει νόημα μόνο όταν το μέγεθος και η πολυπλοκότητα του κώδικα επιτάσσουν την εν λόγω διαίρεση.

Το διάγραμμα των πακέτων του συστήματος ακολουθεί στο Σχ. 1.



Σχήμα 1. Διάγραμμα πακέτων (εδώ: από την αξιολόγηση εστιατορίου)

Ακολουθεί μια συνοπτική περιγραφή των πακέτων του συστήματος.

ΠΑΚΕΤΑ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

`userViews`	Περιέχει τις boundary classes που είναι υπεύθυνες για την αλληλεπίδραση με το χρήστη
`mainEngine`	Κεντρική business logic engine, along with the necessary interface to export to the boundary classes
`dataLoad`	Υποσύστημα αλληλεπίδρασης με τα αρχεία δεδομένων, για την ανάκτησή τους από το σύστημα
`reportEngines`	Υποσύστημα παραγωγής αναφορών
`dataModel`	Domain classes of the system

Πίνακας 3. Συνοπτική περιγραφή πακέτων συστήματος (εδώ: από την αξιολόγηση εστιατορίου)

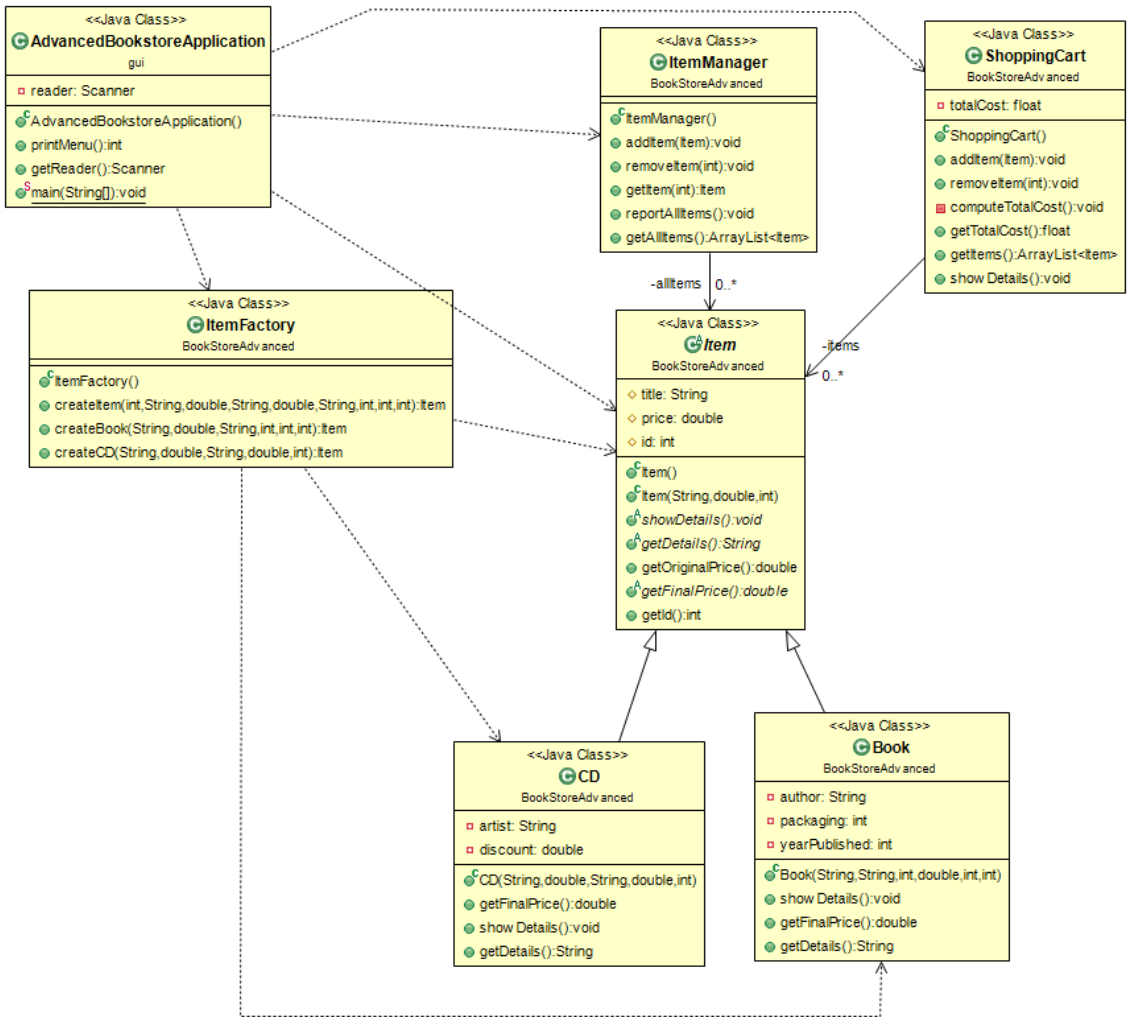
3.2 ΔΙΑΓΡΑΜΜΑΤΑ ΚΛΑΣΕΩΝ

Στην παρούσα υποενότητα, παρατίθενται τα διαγράμματα κλάσεων και ακολουθιών.

Π.χ., για το παράδειγμα με τον έλεγχο του βιβλιοπωλείου έχουμε περισσότερα του ενός διαγράμματα.

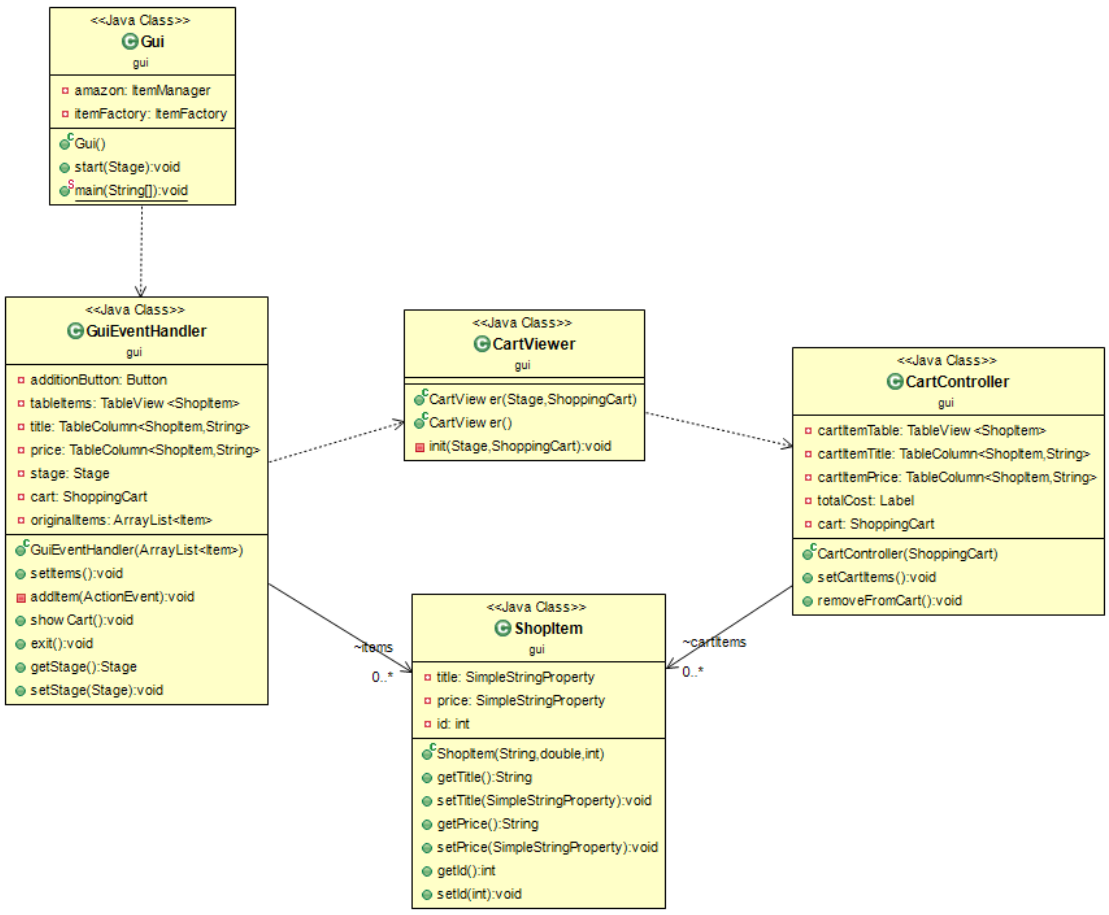
(ΠΡΟΣΟΧΗ: μόλις αλλάξαμε project απ' όπου δανειζόμαστε παραδείγματα!)

package booskstoreAdvanced;



Σχήμα 2. Διάγραμμα κλάσεων για το πακέτο booskstoreAdvanced

```
package gui;
```

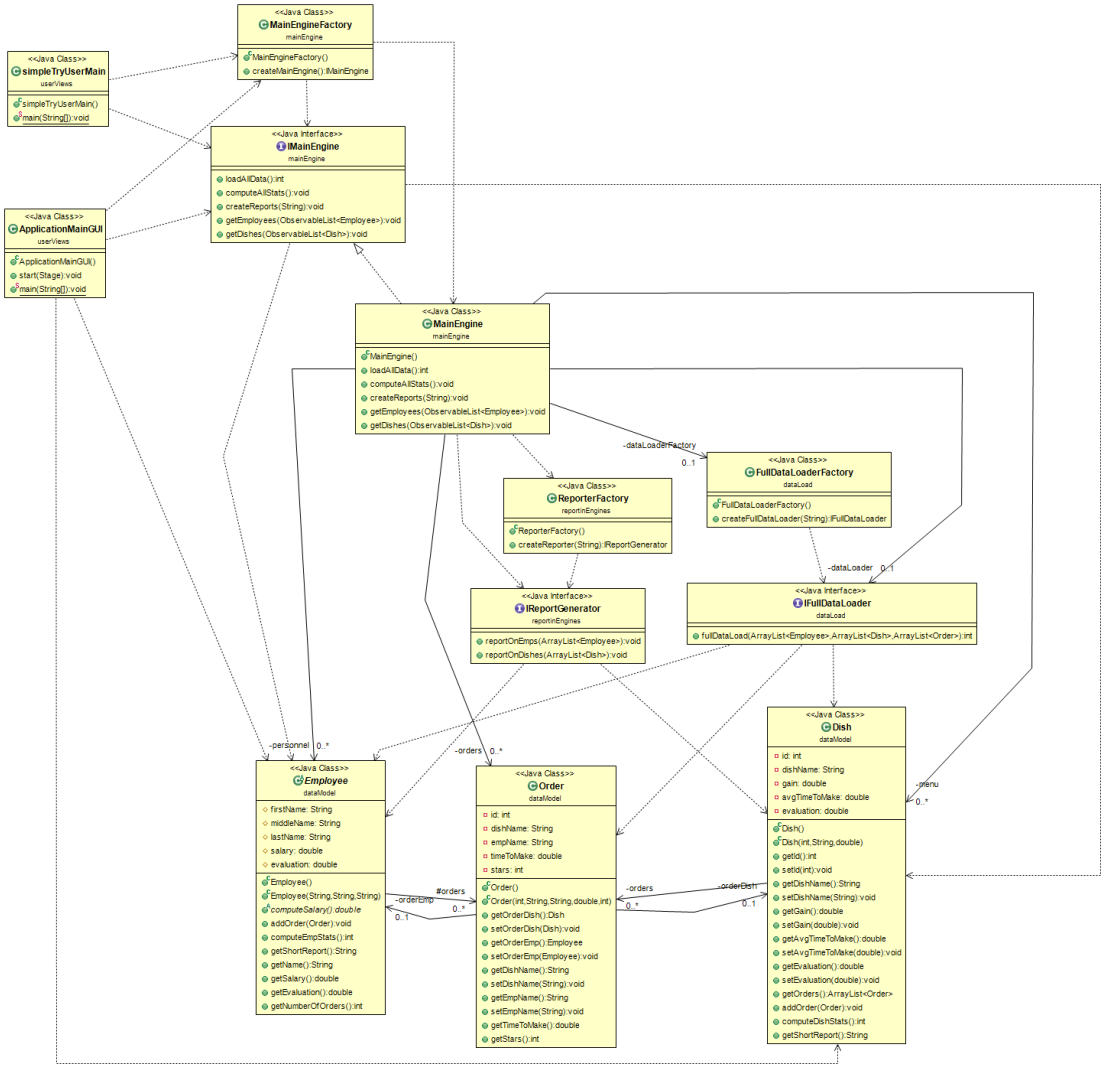


Σχήμα 3. Διάγραμμα κλάσεων για το πακέτο gui

Μπορείτε, επίσης, στα διαγράμματα που δίνετε, να βάζετε και συνεργαζόμενες κλάσεις από άλλα πακέτα. Το διάγραμμα κλάσεων του Σχ. 4 παρουσιάζει τις κεντρικές κλάσεις της εφαρμογής αξιολόγησης εστιατορίου, μαζί με τις συνεργαζόμενες κλάσεις.

- Πλεονεκτήματα: συνολική εποπτεία του συστήματος σε μία (1) απεικόνιση.
- Μειονεκτήματα: η απεικόνιση είναι στα όρια του οπτικά διακριτού. Από ένα σημείο κι έπειτα, είναι αδύνατο να διακρίνει κανείς τι είναι στο διάγραμμα (ακόμα κι αν έχει ακολουθήσει μια προσεγμένη διάταξη στο χώρο), οπότε το διάγραμμα γίνεται πρακτικά άχρηστο. (θυμηθείτε: το διάγραμμά σας, εγώ θα το δω σε χαρτί – θα έχουμε μπροστά μας μόνο ό,τι τυπώσετε)

Παρατηρήστε επίσης, ότι αν τυχόν χρειάζεται να τεκμηριώσουμε / εξηγήσουμε / ανακτήσουμε την συνεργασία ενός υποσυνόλου κλάσεων, είναι απολύτως νόμιμο να κατασκευάζουμε διαγράμματα με κλάσεις από διαφορετικά πακέτα -- δεν μας περιορίζουν τα όρια των πακέτων, δλδ. Τα διαγράμματα πρέπει να εξυπηρετούν μια ομάδα ανάπτυξης λογισμικού και όχι οι άνθρωποι τα διαγράμματα.
(Εννοείται πως αυτό δεν είναι δικαιολογία για να τα ισοπεδώσουμε όλα…)



Σχήμα 4. Διάγραμμα κλάσεων επεξήγησης ενός πακέτου με συνεργαζόμενες κλάσεις

3.3 ΑΝΑΛΥΣΗ ΚΛΑΣΕΩΝ ΚΑΙ ΣΥΝΕΠΕΙΑ ΠΡΟΣ ΤΙΣ ΑΠΑΙΤΗΣΕΙΣ

Στην παρούσα ενότητα παραθέτουμε μια ανάλυση των κλάσεων και μια τεκμηρίωση της κάλυψης των βασικών use cases του συστήματος .

Πρέπει ΥΠΟΧΡΕΩΤΙΚΑ να μου εξηγήσετε:

(α) Την ταξινόμηση των κλάσεων σε Domain/Business Logic/Boundary classes

(β) Τα interfaces between subsystems (emph., for Business Logic classes)

(γ) Την απεικόνιση των use cases σε μεθόδους (όχι σε κλάσεις, σε *μεθόδους*)

Αυτού του είδους η τεκμηρίωση δεν θα υπήρχε σε μια επαγγελματική αναφορά – όμως, επαληθεύει την οργάνωση και την πληρότητα της σχεδίασής σας.

3.3.1 DOMAIN CLASSES

Package `booskstoreAdvanced`	`CD`, `Book`, μια αφηρημένη κλάση `Item` (A) για αυτές τα δύο, και ένα factory, το `ItemFactory`, για την κατασκευή στιγμιοτύπων. `ShoppingCart`.

3.3.2 BUSINESS LOGIC CLASSES

Package `booskstoreAdvanced`	`ItemManager`, για την υλοποίηση όλων των use cases στο back-end. `ItemManager`: - Interfaces with domain classes via `Item` + `Factory`. - Has no interface to boundary classes (!!!)

3.3.3 BOUNDARY CLASSES

Package `gui`	`AdvancedBookstoreApplication`, μια client class για την αλληλεπίδραση με το χρήστη μέσω κονσόλας. `Gui`, `GuiEventHandler`, `CartViewer`, `CartController`, `ShoppingItem`, για το graphical user interface με το χρήστη (οι handler/controller classes χειρίζονται τα events από τη γραφική διαπροσωπεία).

3.3.4 ΑΠΕΙΚΟΝΙΣΗ ΑΠΑΙΤΗΣΕΩΝ ΣΕ ΜΕΘΟΔΟΥΣ

Υπάρχουν 4 use cases για το ηλ. βιβλιοπωλείο: εμφάνισε προϊόντα καταστήματος, πρόσθεσε προϊόν στο καλάθι, διέγραψε προϊόν από το καλάθι, εμφάνισε προϊόντα του καλαθιού.

ΑΠΕΙΚΟΝΙΣΗ USE CASES ΣΕ ΜΕΘΟΔΟΥΣ

Use case	Back-end methods	Front-end methods
εμφάνισε προϊόντα καταστήματος	`ItemManager.reportAllItems()`	`Gui.start()` `GuiEventHandler.setItems()`
πρόσθεσε προϊόν στο καλάθι	`ShoppingCart.addItem()`	`GuiEventHandler.addItem()`
διέγραψε προϊόν από το καλάθι	`ShoppingCart.removeItem()`	`CartController.removeFromCart()`
εμφάνισε προϊόντα του καλαθιού	`ShoppingCart.showDetails()`	`GuiEventHandler.showCart()`

Πίνακας 4 Επαλήθευση απεικόνισης use cases σε μεθόδους

3.4 ΔΙΑΓΡΑΜΜΑΤΑ ΑΚΟΛΟΥΘΙΩΝ

Αν ζητούνται / υπάρχουν

4 ΛΟΙΠΑ ΣΧΟΛΙΑ

Εδώ προστίθενται όποια σχόλια μπορεί να υπάρχουν (αν υπάρχουν) για σχεδιαστικές υποθέσεις, αποφάσεις, ελλείψεις και σημεία κινδύνου, ή για οτιδήποτε άλλο κρίνετε σημαντικό να καταγραφεί για τη μελλοντική συντήρηση του κώδικα.

4.1 ΣΧΕΔΙΑΣΤΙΚΕΣ ΑΠΟΦΑΣΕΙΣ

Αν υπάρχει λόγος να καταγραφούν εναλλακτικές σχεδιάσεις και γιατί αποφασίσθηκε να προκριθεί κάποια από αυτές.

4.2 ΣΗΜΕΙΑ ΚΙΝΔΥΝΟΥ

Αν υπάρχουν.

4.3 ΕΚΚΡΕΜΟΤΗΤΕΣ (TODO)

Αν υπάρχουν. Π.χ., εδώ:

- Missing interface(s) between `ItemManager` and boundary classes