Μετρικές σύζευξης:

CA (Afferent Coupling)
Δείχνει τον αριθμό των άλλων κλάσεων ή πακέτων που εξαρτώνται από κλάσεις εντός ενός συγκεκριμένου πακέτου. Αυτή η τιμή είναι καλή για να δούμε πως αλλαγές στις κλάσεις ενός πακέτου θα επηρεάσουν άλλα μέρη του προγράμματος.

Dao:	7
View:	4
Controller:	3
Model:	3
Controller.impl:	1
Dao.impl:	1
Main:	0

Στο πρόγραμμα μας βλέπουμε ότι, από τις κλάσεις του dao εξαρτώνται όλα τα πακέτα μας. Τόσο μεγάλος αριθμός είναι αποδεκτός αν οι κλάσεις του πακέτου είναι σχετικά αφηρημένες. Μεγάλος είναι και αριθμός των κλάσεων που εξαρτώνται από το πακέτο του Controller ο οποίος όμως και αυτός είναι αποδεκτός για τον ίδιο ακριβώς λόγο. Όπως είναι φυσικό η Main δεν εξαρτάται από καμία κλάση.

CE (Efferent Coupling)
Δείχνει τον αριθμό των άλλο πακέτων από τις οποίες εξαρτιούνται οι κλάσεις εντός ενός συγκεκριμένου πακέτου. Αυτή η τιμή είναι καλή για να δούμε πόσο ευαίσθητο είναι ένα πακέτο από τις αλλαγές σε άλλα πακέτα.

Controller.impl:	3
Dao.impl:	2
View:	2
Main:	1
Model:	1
Controller:	0
Dao:	0

Σε αντίθεση με το CA, στο CE βλέπουμε ότι από τις αφηρημένες κλάσεις (controller και dao) δεν εξαρτιούνται άλλες κλάσεις ή πακέτα. Ενώ τα πακέτα controller.impl, dao.impl και view έχουν εξαρτήσεις αφού είναι και είναι αυτά που κάνουν “όλοι την δουλειά”.

RMI (Instability)
Είναι προτιμότερη από τις προηγούμενες δύο μετρικές γιατί τις συνδυάζει. Είναι η αναλογία ανάμεσα στο CE και στο άθροισμα CE + CA δηλαδή RMI = (CE / (CE + CA)).
Αν το αποτέλεσμα πλησιάζει στο 0, δείχνει ένα σταθερό πακέτο εντελώς ανεξάρτητο.
Αν το αποτέλεσμα πλησιάζει στο 1, δείχνει ένα ασταθές πακέτο που εξαρτάται πλήρως από άλλα πακέτα.

Main:	1
Controller.impl:	0.75
Dao.impl:	0.667
View:	0.333
Model:	0.25
Controller:	0
Dao:	0

Το να σχεδιαστούν τα πακέτα να είναι τελείως ασταθή ή τελείως σταθερά δεν είναι και τόσο καλή πρακτική. Το σύνηθες είναι να τα σχεδιάσουμε ή σταθερά, με τιμές περίπου από 0.0 έως 0.3, ή ασταθή, με τιμές περίπου από 0.7 έως 1. Όπως θα περιμέναμε, τα πακέτα controller και dao είναι σταθερά και εντελώς ανεξάρτητα από άλλα πακέτα, σε αντίθεση με τα πακέτα controller.impl, dao.impl και την main τα οποία εξαρτώνται πλήρως από άλλα πακέτα και κλάσεις.

Μετρικές πολυπλοκότητας:
MCC (McCabe Cyclomatic Comlexity, άλλη συντομογραφία: VG)
Μετρά τον αριθμό των ανεξάρτητων διαδρομών μέσω μιας συγκεκριμένης μεθόδου.
Συχνά χρησιμοποιείται για να ελέγξει το πόσο εύκολα μπορούμε να τεστάρουμε μια μέθοδο.
Όσο υψηλότερη είναι η τιμή της μετρικής, τόσο πιο δύσκολα τεστάρετε αυτή η μέθοδος.
Φυσιολογική τιμή: 1-10

Μέτρια τιμή: 11-20
Μεγάλη τιμή: 21-50
Υπερβολική τιμή: 51+
Τον αριθμό της κυκλωματικής πολυπλοκότητας McCabe (M) τον αυξάνουν τα εξής:

- **`if`** statement
- **`while`** statement
- **`for`** statement
- **`case`** statement
- **`catch`** statement
- **`&&`** and **`||`** boolean operations
- **`?:`** ternary operator and **`?:`** *Elvis* operator.
- **`?.`** null-check operator

Αυτό που μας ενδιαφέρει κυρίως είναι οι μέσες τιμές των πακέτων αλλά και οι ανώτατες τιμές να μην ξεπερνούν τα όρια των μετρίων τιμών.

	Μέση	Ανώτατη
Controller.impl:	1.342	6
Main:	5	5
Dao.impl:	1.8	3
View:	1.113	3
Model:	1	1
Controller:	0	0
Dao:	0	0

Όπως βλέπουμε σε όλα τα πακέτα έχουμε ικανοποιητικές μέσες και ανώτατες τιμές. Στο πακέτο controller.impl τις μεγαλύτερες τιμές τις έχουν οι μέθοδοι itemStateChanged των viewByBoxListener και viewByOptionBoxListener στην κλάση ProductControllerImpl στα οποία υπάρχει μία if και μια switch case 5 περιπτώσεων.
Στην κλάση της main βλέπουμε ότι έχουμε μέση και ανώτατη τιμή ίδια γιατί υπάρχει μία μόνο μέθοδος στην οποία έχουμε try – catch, for και if.
Στο πακέτο dao.ipml έχουμε μεση τιμή 1,8 και ανώτατη 3 με επιμέρους μέσες τιμές των κλάσεων ProductDaoImpl και DaoImpl, 2,143 και 1 αντίστοιχα. Την υψηλότερη τιμή (3) την συναντούμε στο μέθοδο getItemDetails της κλάσης ProductDaoImpl λόγο της for και της if ενώ στις υπόλοιπες μεθόδους αυτής της κλάσης έχουμε τιμή 2 λόγο της for.
Στο πακέτο view, έχουμε μέση τιμή 1.113 και ανώτατη τιμή 3. Στην κλάση ProductView έχουμε μέση τιμή 1,174 και ανώτατη τιμή 3 στην μέθοδο showpart λόγο της for και του while. Στην κλάση ProductDetailsView έχουμε μέση τιμή 1,5 και ανώτατη τιμή 2 στην μέθοδο intiComponets λόγο του try - catch. Στην κλάση ManageProductView έχουμε μέση τιμή 1,05 και ανώτατη τιμή 2 στην μέθοδο populatedYears λόγο της for.
Στο πακέτο model, έχουμε μέση τιμή 1 και ανώτατη τιμή 1. Ενώ στα πακέτα controller και dao έχουμε από 0 και στην μέση και στην ανώτατη τιμή, πράγμα που το περιμέναμε διότι σε αυτά τα πακέτα, οι κλάσεις είναι αφηρημένες.

Μετρικές συνεκτικότητας:

LCOM (Lack of Cohesion Methods)

Ποσοτικοποίηση του βαθμού συνεκτικότητας μιας κλάσης
P: το σύνολο των μη συνεκτικών ζευγών μεθόδων
Q: το σύνολο των συνεκτικών ζευγών μεθόδων

$$LCOM = \begin{cases} |P| - |Q|, & \text{αν } |P| > |Q| \\ 0, & \text{σε κάθε άλλη περίπτωση} \end{cases}$$

Άρα, όσο μεγαλύτερη είναι η συνεκτικότητα τόσο μικρότερη είναι η τιμή της LCOM. Κλάσεις με LCOM κοντά στο 1 δείχνουν ότι μπορούν να σπάσουν σε επιμέρους κλάσεις.

Model:	0.95
View:	0.925

Dao.impl: 0.438
Controller.impl: 0.062
Main: 0
Controller: 0
Dao: 0

Στο πακέτο model, η κλάση Product έχει τιμή 0,95 (μικρή συνεκτικότητα) πράγμα που μας δείχνει ότι αυτή η κλάση μπορεί να διασπαστεί.
Στο πακέτο view, οι κλάσεις ProductView, ManageProductView και ProductDetailsView έχουν τιμές 0.939, 0.929 και 0.909 δηλαδή έχουν και αυτές μικρή συνεκτικότητα άρα μπορούν και αυτές να διασπαστούν.
Στο πακέτο dao.impl, η κλάση Dao.Impl έχει τιμή 0,875 δηλαδή μπορεί να διασπαστεί, ενώ η κλάση ProdustDaoImpl με τιμή 0 δεν μπορεί να δεχτεί περεταίρω διάσπαση.
Στο πακέτο του controller.impl, η κλάση ControllerImpl με τιμή 0,75 μπορεί να διασπαστεί, ενώ οι υπόλοιπες κλάσεις με τιμή 0 δεν μπορούν να διασπαστούν σε επιμέρους κλάσεις.
Στα πακέτα main, controller και dao η τιμή της LCOM είναι 0 άρα έχουν πολύ μεγάλη συνεκτικότητα.

Μετρικές μεγέθους:
TLOC (Συνολικές γραμμές κώδικα)
Ο συνολικός αριθμός γραμμών κώδικα είναι 1180.

Total Lines of Code	1180
view	576
ProductView.java	246
ManageProductView.java	216
ProductDetailsView.java	114
controller.impl	323
ProductControllerImpl.java	203
ManageProductControllerImpl.java	68
ControllerImpl.java	52
dao.impl	132
ProductDaoImpl.java	104
DaoImpl.java	28
model	84
Product.java	84
main	28
Main.java	28
controller	19
Controller.java	9
ProductController.java	7
ManageProductController.java	3
dao	18
ProductDao.java	9
Dao.java	9

Όπως βλέπουμε τα πακέτα view και controller έχουν τις περισσότερες γραμμές ενώ τα πακέτα που ουσιαστικά δεν κάνουν και πολλά (main, controller και dao) έχουν ελάχιστες γραμμές κώδικα.

MLOC (Γραμμές κώδικα μεθόδων)

Method Lines of Code (avg/max per method)	711
view	410
ManageProductView.java	150
ProductView.java	168
ProductDetailsView.java	92
controller.impl	174
ProductControllerImpl.java	115
ManageProductControllerImpl.java	34
ControllerImpl.java	25
main	20
Main.java	20
dao.impl	88
ProductDaoImpl.java	81
DaoImpl.java	7
model	19
Product.java	19
controller	0
ProductController.java	0
Controller.java	0
ManageProductController.java	0
dao	0

NOM (Αριθμός μεθόδων)

Number of Methods (avg/max per type)	112
view	45
ProductView.java	23
ManageProductView.java	20
ProductDetailsView.java	2
model	19
Product.java	19
controller.impl	38
ProductControllerImpl.java	23
ControllerImpl.java	7
ManageProductControllerImpl.java	8
dao.impl	10
ProductDaoImpl.java	7
DaoImpl.java	3
controller	0
ProductController.java	0
Controller.java	0
ManageProductController.java	0
dao	0
ProductDao.java	0
Dao.java	0
main	0
Main.java	0