

# Αυτοματοποιημένη ενσωμάτωση μοτίβων σχεδίασης σε πηγαίο κώδικα Java

Αναστάσιος Λιόντος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τμήμα Μηχανικών Η/Υ και Πληροφορικής  
Πολυτεχνική Σχολή  
Πανεπιστήμιο Ιωαννίνων

Ιούλιος 2023

# ΑΦΙΕΡΩΣΗ

---

Στην Οικογένεια μου

# ΠΕΡΙΕΧΟΜΕΝΑ

---

Κατάλογος Σχημάτων	iii
Κατάλογος Πινάκων	v
Παράδειγμα Κώδικα	vi
Περίληψη	vii
Abstract	viii
<b>1 Εισαγωγή</b>	<b>1</b>
1.1 Σχεδιαστικά μοτίβα . . . . .	1
1.2 Στόχοι . . . . .	1
1.3 Δομή της διπλωματικής εργασίας . . . . .	2
<b>2 Σχετική Δουλειά</b>	<b>3</b>
2.1 Σχετικά εργαλεία . . . . .	3
2.1.1 Σύγκριση . . . . .	4
2.2 Υπόβαθρο . . . . .	4
<b>3 Ανάλυση Απαιτήσεων</b>	<b>7</b>
3.1 Ιστορίες Χρήστη . . . . .	7
3.2 Περιπτώσεις χρήσης . . . . .	8
<b>4 Σχεδίαση και αρχιτεκτονική λογισμικού</b>	<b>18</b>
4.1 Αρχιτεκτονική . . . . .	22
4.2 dpb.wizards.mainWizard . . . . .	24
4.3 dpb.wizards.setupWizards . . . . .	25
4.4 dpb.controller . . . . .	27

4.5	dpb.io . . . . .	28
4.6	dpb.model . . . . .	29
<b>5</b>	<b>Έλεγχος</b>	<b>32</b>
5.1	Έλεγχος δομής σχεδιαστικών μοτίβων . . . . .	32
5.1.1	Έλεγχος μεθόδων φορτώματος μοτίβων . . . . .	32
5.1.2	Έλεγχος μεθόδου ανάκτησης περιγραφής μοτίβου . . . . .	34
5.2	Έλεγχος μεθόδων δημιουργίας πηγαίου κώδικα Java . . . . .	34
5.3	Λοιποί έλεγχοι . . . . .	38
5.4	Αναφορά αποτελεσμάτων ελέγχων . . . . .	39
<b>6</b>	<b>Οδηγός Χρήσης Design Pattern Builder</b>	<b>40</b>
6.1	Λειτουργίες χρήστη . . . . .	40
<b>7</b>	<b>Επίλογος</b>	<b>57</b>
7.1	Σύνοψη και συμπεράσματα . . . . .	57
7.2	Μελλοντικές επεκτάσεις . . . . .	58
	<b>Βιβλιογραφία</b>	<b>59</b>
<b>A</b>	<b>Κάρτες αρμοδιοτήτων και συνεργασιών κλάσεων</b>	<b>60</b>

# ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

---

3.1	Διάγραμμα UML Περιπτώσεων Χρήστη. . . . .	9
4.1	Σχήμα αρχείου xml. . . . .	20
4.2	Διάγραμμα UML Πακέτων συστήματος. . . . .	22
4.3	Διάγραμμα UML Πακέτου mainWizard. . . . .	24
4.4	Διάγραμμα UML Πακέτου setupWizards. . . . .	25
4.5	Διάγραμμα UML Πακέτου controller. . . . .	27
4.6	Διάγραμμα UML Πακέτου io. . . . .	28
4.7	Διάγραμμα UML Πακέτου model. . . . .	29
5.1	Αναφορά αποτελεσμάτων ελέγχων . . . . .	39
6.1	Άνοιγμα οδηγού. . . . .	41
6.2	Σελίδα επιλογής κατηγορίας & μοτίβου. . . . .	42
6.3	Επιλογή μοτίβου. . . . .	43
6.4	Οι κλάσεις & διεπαφές του μοτίβου Iterator. . . . .	44
6.5	Επεξεργασία ονόματος κλάσης. . . . .	45
6.6	Επεξεργασία πεδίων. . . . .	46
6.7	Προσθήκη πεδίου. . . . .	47
6.8	Επεξεργασία μεθόδων κλάσης. . . . .	48
6.9	Προσθήκη μεθόδου. . . . .	49
6.10	Επεξεργασία παραμέτρων μεθόδου. . . . .	50
6.11	Προσθήκη παραμέτρου. . . . .	51
6.12	Επεξεργασία ονόματος διεπαφής. . . . .	52
6.13	Επεξεργασία μεθόδων διεπαφής. . . . .	53
6.14	Διεπαφή Collection. . . . .	54
6.15	Κλάση ConcreteCollection. . . . .	55

6.16 Διεπαφή Iterator. . . . .	55
6.17 Κλάση ConcreteIterator. . . . .	56

# ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

---

3.1	Ιστορίες χρήστη. . . . .	8
3.2	Επιλογή κατηγορίας μοτίβου. . . . .	10
3.3	Επιλογή μοτίβου. . . . .	10
3.4	Καθορισμός μεθόδων κλάσης. . . . .	11
3.5	Ονοματοδοσία κλάσης. . . . .	12
3.6	Καθορισμός πεδίων κλάσης. . . . .	13
3.7	Ονοματοδοσία διεπαφής. . . . .	14
3.8	Καθορισμός μεθόδων διεπαφής. . . . .	15
3.9	Εξαγωγή σκελετού επιλεγμένου μοτίβου. . . . .	16
3.10	Προσθήκη νέας κλάσης. . . . .	16
3.11	Καθορισμός υλοποιημένης διεπαφής. . . . .	17
A.1	PatternClass κάρτα αρμοδιοτήτων. . . . .	60
A.2	PatternInterface κάρτα αρμοδιοτήτων. . . . .	61
A.3	Method κάρτα αρμοδιοτήτων. . . . .	61
A.4	Field κάρτα αρμοδιοτήτων. . . . .	62
A.5	FileParser κάρτα αρμοδιοτήτων. . . . .	63
A.6	PatternManager κάρτα αρμοδιοτήτων. . . . .	64
A.7	ClassGenerator κάρτα αρμοδιοτήτων . . . . .	65
A.8	InterfaceGenerator κάρτα αρμοδιοτήτων. . . . .	66

# ΚΑΤΑΛΟΓΟΣ ΠΑΡΑΔΕΙΓΜΑΤΩΝ ΚΩΔΙΚΑ

---

4.1	Περιγραφή μοτίβου Singleton. . . . .	21
5.1	Mock κλάση για την κλάση ClassGenerator . . . . .	36
5.2	Mock κλάση για την κλάση PackageFragment . . . . .	37



# ΠΕΡΙΛΗΨΗ

---

Αναστάσιος Λιόντος, Δίπλωμα, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, Ιούλιος 2023.

Αυτοματοποιημένη ενσωμάτωση μοτίβων σχεδίασης σε πηγαίο κώδικα Java.

Επιβλέπων: Απόστολος Ζάρρας, Καθηγητής.

Το εργαλείο αυτό αποτελεί μία επέκταση του Eclipse IDE, έχει ως στόχο να βοηθήσει κάποιον αρχάριο κυρίως, προγραμματιστή, να εισάγει εύκολα και γρήγορα κάποιο σχεδιαστικό μοτίβο. Μέσω μίας γραφικής διεπαφής ένας προγραμματιστής μπορεί να επιλέξει κατηγορία και το μοτίβο που επιθυμεί, να ρυθμίσει τις κλάσεις και τις διεπαφές σύμφωνα με τις απαιτήσεις του κώδικα του και τέλος να εξάγει τον σκελετό του μοτίβου στον κώδικα του.

**Λέξεις κλειδιά:** Επέκταση Eclipse, διπλωματική εργασία, σχεδιαστικά μοτίβα, προγραμματιστής

# ABSTRACT

---

Anastasios Lontos, Diploma, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, July 2023.

Automated incorporation of design patterns in Java source code.

Advisor: Apostolos Zarras, Professor.

This tool is an extension of the Eclipse IDE, it aims to help a beginner programmer, to easily and quickly introduce a design pattern. Through a graphical interface a developer can use a graphical user interface to help a programmer can select the category and pattern he wants, set up the classes, and set up the interfaces according to the requirements of his code and finally export the skeleton of pattern to code of.

**Keywords:** Eclipse plugin, tool, diploma thesis, design patterns, developer, programmer

# ΚΕΦΑΛΑΙΟ 1

## Εισαγωγή

### 1.1 Σχεδιαστικά μοτίβα

Ένα σχεδιαστικό μοτίβο [1] ορίζει μια γενική λύση σε ένα συχνά εμφανιζόμενο πρόβλημα, γραμμένο ως πρότυπο ή ως σύνολο σχέσεων. Στην επιστήμη των υπολογιστών, αυτό μεταφράζεται σε ένα προγραμματιστικό μοτίβο ή ένα διάγραμμα αντικειμένων που αναπαριστά μια γνωστή λύση σε ένα κοινό πρόβλημα.

Όπως οι αλγόριθμοι, έτσι και τα πρότυπα σχεδίασης είναι μια δοκιμασμένη και αποδεκτή λύση και ως εκ τούτου, οι άνθρωποι τα αναπτύσσουν και τα προσαρμόζουν αντί να τα ανακαλύπτουν. Και τα δύο αντιπροσωπεύουν μια προσέγγιση ενός προβλήματος περισσότερο από την ειδική υλοποίηση της λύσης. Ωστόσο, διαφέρουν ως προς τον σκοπό τους οι αλγόριθμοι βελτιστοποιούν το κόστος μιας λύσης, ενώ τα πρότυπα σχεδίασης βελτιστοποιούν τη σαφήνειά της.

### 1.2 Στόχοι

Η παρούσα διπλωματική εργασία επιδιώκει την ανάπτυξη ενός εργαλείου, το οποίο ονομάζεται Design Pattern Builder. Το εργαλείο αυτό επιτρέπει σε αρχάριους προγραμματιστές να εισάγουν σχεδιαστικά μοτίβα στον κώδικά τους σε Java με απλά βήματα. Με το Design Pattern Builder ο χρήστης έχει την δυνατότητα: να επιλέξει το επιθυμητό σχεδιαστικό μοτίβο, να κάνει τις απαραίτητες ρυθμίσεις και να εισάγει αυτόματα τον αντίστοιχο κώδικα στον δικό του κώδικα. Το εργαλείο περιλαμβάνει μια συλλογή από τα σχεδιαστικά μοτίβα που προτείνονται από την ομάδα

των GOF [1], όπως το Singleton, το Factory, το Observer και άλλα. Ο χρήστης έχει τη δυνατότητα να εξερευνήσει αυτήν τη συλλογή, να επιλέξει το επιθυμητό μοτίβο και να εισάγει αυτόματα τον απαιτούμενο κώδικα στον κώδικά του. Ο στόχος είναι να διευκολυνθεί ο αρχάριος προγραμματιστής στη χρήση σχεδιαστικών μοτίβων και να ενισχυθεί η προγραμματιστική του απόδοση και ακρίβεια, επιτρέποντάς του να εισάγει εύκολα τα απαραίτητα σχεδιαστικά μοτίβα στον κώδικά του, χωρίς την ανάγκη για χειροκίνητη υλοποίηση. Συνολικά, ο στόχος είναι να δημιουργηθεί ένα εύχρηστο και βοηθητικό εργαλείο που θα επιτρέπει στους αρχάριους προγραμματιστές να αξιοποιούν τα σχεδιαστικά μοτίβα.

### **1.3 Δομή της διπλωματικής εργασίας**

Το υπόλοιπο της διπλωματικής εργασίας περιλαμβάνει τα εξής κεφάλαια: Στο κεφάλαιο 2, γίνεται ανάλυση της παρελθοντικής δουλειάς που έχει γίνει και είναι σχετική με το εργαλείο που πραγματεύεται η παρούσα διπλωματική. Το κεφάλαιο 3, περιέχει τις ιστορίες χρήστη καθώς, και τις περιπτώσεις χρήσης του εργαλείου. Το κεφάλαιο 4, εξετάζει την αρχιτεκτονική του εργαλείου, καθώς παρουσιάζονται λεπτομερώς οι κλάσεις και τα πακέτα που αποτελούν το εργαλείο. Στο κεφάλαιο 5, περιέχονται λεπτομέρειες σχετικά με τον αυτοματοποιημένο έλεγχο του εργαλείου. Στο κεφάλαιο 6, παρουσιάζονται οι οδηγίες χρήσης του εργαλείου. Τέλος, στο κεφάλαιο 7 παρουσιάζονται οι μελλοντικές επεκτάσεις του εργαλείου.

# ΚΕΦΑΛΑΙΟ 2

## Σχετική Δουλειά

### 2.1 Σχετικά εργαλεία

**Pattern Wizard.** Το εργαλείο αυτό [2], προσφέρει ως λειτουργίες, την επιλογή κάποιου μοτίβου από τα Adapter, Abstract Factory και Observer, καθώς και την επιλογή γλώσσας προγραμματισμού, μέχρι στιγμής την Java. Στην συνέχεια ο χρήστης έχει την δυνατότητα να αντιστοιχίσει υπάρχων κώδικα στο μοτίβο που έχει επιλέξει και το εργαλείο μετατρέπει τον κώδικα σε κώδικα που είναι συμβατός με το μοτίβο. Τέλος, ο προγραμματιστής έχει την δυνατότητα να επιλέξει κάποιο κενό πηγαίο αρχείο και το εργαλείο εξάγει τον σκελετό του μοτίβου.

**Patternbox.** Το εργαλείο αυτό [2] υποστηρίζει 16 μόνο μοτίβα από αυτά που έχουν προτείνει οι GoF [1], δημιουργεί ένα ειδικό αρχείο το οποίο αναπαριστά το μοτίβο. Μέσα από το αρχείο μπορεί ο προγραμματιστής να επιλέξει κάποιο στοιχείο του μοτίβου και να επιλέξει την τοποθεσία όπου θα εξαχθεί είτε η κλάση είτε η διεπαφή που έχει επιλέξει ο χρήστης. Αφού ο χρήστης ολοκληρώσει την διαδικασία, το εργαλείο παράγει απλώς τον σκελετό του μοτίβου, χωρίς να τροποποιεί τον κώδικα.

**Design Pattern Automation Toolkit.** Το εργαλείο αυτό [2] υποστηρίζει και τα 23 μοτίβα των GoF [1], υποστηρίζει λειτουργίες όπως επιλογή μοτίβου και γλώσσας προγραμματισμού. Επίσης κάποιος προγραμματιστής μπορεί να εισάγει νέα μοτίβα που μπορεί να υλοποιήσει το εργαλείο. Τέλος δεν τροποποιεί τον υπάρχων κώδικα παρά μόνο εξάγει τον σκελετό του μοτίβου.

**Alphaworks Design Pattern Toolkit.** Αυτό το εργαλείο [2] είναι αντίστοιχο του Patternbox, με την διαφορά ότι το Alphaworks μετασχηματίζει τον κώδικα του προ-

γραμματιστή σε ένα ενδιάμεσο αρχείο τύπου xml, αφού ο προγραμματιστής έχει εισάγει στον κώδικά του κατάλληλα tags. Στη συνέχεια, ο προγραμματιστής κάνει τις αλλαγές που επιθυμεί στο αρχείο xml και το εργαλείο μετασχηματίζει πίσω σε κώδικα συμβατό με το μοτίβο.

### 2.1.1 Σύγκριση

To Design Pattern Builder, το εργαλείο που αποτελεί αντικείμενο της διπλωματικής αυτής, παρουσιάζει πλεονεκτήματα έναντι των παραπάνω εργαλείων. Ένα από τα πλεονεκτήματα του εργαλείου είναι ότι ο προγραμματιστής μπορεί να τροποποιήσει οποιοδήποτε μοτίβο, ώστε το μοτίβο να ταιριάζει με τις ανάγκες του έργου του προγραμματιστή. Επίσης, εξάγει annotations -τα οποία αποδίδουν τον ρόλο που έχει η κλάση ή η διεπαφή- στον σκελετό του μοτίβου ώστε να διευκολύνει τον προγραμματιστή στην εφαρμογή του μοτίβου. Τέλος, το εργαλείο που προτείνουμε προσφέρει και τα 23 μοτίβα των GoF [1].

Από την άλλη, το εργαλείο που προτείνουμε υστερεί ως προς την δυνατότητα αντιστοίχισης ενός στοιχείου, δηλαδή μίας κλάσης ή μίας διεπαφής, του μοτίβου με κάποιο στοιχείο του έργου του προγραμματιστή.

## 2.2 Υπόβαθρο

Τα σχεδιαστικά μοτίβα GoF (Gang of Four) είναι ένα σύνολο σχεδιαστικών προτύπων που περιγράφονται στο βιβλίο [1]. Αυτά τα πρότυπα προέκυψαν από την εμπειρία και την εμπειρογνωμοσύνη των τεσσάρων συγγραφέων, και αποτελούν γενικές λύσεις για συχνά προβλήματα στον σχεδιασμό λογισμικού. Οι GoF πρότειναν 23 μοτίβα στο πλαίσιο της γλώσσας προγραμματισμού C++, αλλά μπορούν να εφαρμοστούν και σε άλλες γλώσσες αντικειμενοστραφούς προγραμματισμού όπως η Java.

Για να περιγράψουμε ένα σχεδιαστικό μοτίβο, δεν αρκεί μία απλή γραφική αναπαράσταση, καθώς απλώς αποτυπώνει τις σχέσεις μεταξύ κλάσεων και αντικειμένων. Για να μπορέσουμε να επαναχρησιμοποιήσουμε την σχεδίαση πρέπει να μπορούμε να καταγράψουμε τις αποφάσεις, τους συμβιβασμούς που χρειάζεται να κάνουμε για να εφαρμόσουμε την σχεδίαση αυτή, καθώς και τις εναλλακτικές λύσεις. Χρειάζονται επίσης, παραδείγματα, καθώς αναδεικνύουν την εφαρμογή της

σχεδίασης αυτής. Για την περιγραφή των μοτίβων χρησιμοποιείται μία συνεπής μορφή. Κάθε μοτίβο διαιρείται σε ενότητες σύμφωνα με το πρότυπο που θα αναλυθεί παρακάτω. Το πρότυπο προσδίδει μια ομοιόμορφη δομή στις πληροφορίες, διευκολύνοντας την εκμάθηση, τη σύγκριση, καθώς και την χρήση των σχεδιαστικών μοτίβων.

- **Όνομα και κατηγορία μοτίβου:** Το όνομα αποδίδει συνοπτικά την ουσία του μοτίβου.
- **Πρόθεση:** Τι κάνει το μοτίβο ποία είναι η λογική και η πρόθεση του. Ποιο σχεδιαστικό πρόβλημα αντιμετωπίζει το μοτίβο.
- **Γνωστό και ως:** Άλλα γνωστά ονόματα για το μοτίβο, εάν υπάρχουν.
- **Κίνητρα:** Ένα σενάριο που απεικονίζει ένα σχεδιαστικό πρόβλημα και τον τρόπο με τον οποίο οι δομές κλάσεων και αντικειμένων στο μοτίβο λύνουν το πρόβλημα. Το σενάριο βοηθάει να κατανοηθεί αφηρημένη περιγραφή του προτύπου που ακολουθεί.
- **Εφαρμογές:** Σε ποιες περιπτώσεις μπορεί να εφαρμοστεί το πρότυπο σχεδίασης. Ποια είναι τα παραδείγματα κακής σχεδίασης που μπορεί να αντιμετωπίσει το πρότυπο.
- **Δομή:** Ένα διάγραμμα OMT [3], των κλάσεων του προτύπου και διαγράμματα αλληλεπίδρασης [4] για την απεικόνιση ακολουθίες αιτημάτων και συνεργασίες μεταξύ αντικειμένων.
- **Συμμετέχοντες:** Οι κλάσεις και/ή τα αντικείμενα που συμμετέχουν στο πρότυπο σχεδίασης και οι αρμοδιότητές τους.
- **Συνεργασίες:** Πώς οι συμμετέχοντες συνεργάζονται για την εκτέλεση των αρμοδιοτήτων τους.
- **Συνέπειες:** Πώς η σχεδίαση υποστηρίζει τους στόχους της. Ποια είναι τα αντισταθμιστικά οφέλη και τα αποτελέσματα της χρήσης του προτύπου.
- **Υλοποίηση:** Ποιες παγίδες, τεχνικές θα πρέπει να γνωρίζει κάποιος κατά την εφαρμογή του προτύπου. Υπάρχουν θέματα που αφορούν συγκεκριμένες γλώσσες.

- **Παραδείγματα:** Τμήματα κώδικα που απεικονίζουν τον τρόπο με τον οποίο μπορεί να υλοποιηθεί το πρότυπο σε C++ ή Smalltalk.
- **Γνωστές χρήσεις:** Παραδείγματα του μοτίβου που συναντώνται σε πραγματικά συστήματα. Περιλαμβάνονται τουλάχιστον δύο παραδείγματα από διαφορετικούς τομείς.
- **Σχετικά μοτίβα:** Ποια πρότυπα σχεδίασης σχετίζονται με αυτό. Ποιες είναι οι σημαντικές διαφορές. Με ποια άλλα πρότυπα θα πρέπει να χρησιμοποιείται αυτό το πρότυπο.

Τα μοτίβα που πρότειναν οι GoF, χωρίζονται σε τρεις κατηγορίες,

- **Δημιουργικά πρότυπα:** Για την δημιουργία αντικειμένων.
- **Δομικά πρότυπα:** Για την δημιουργία σχέσεων μεταξύ αντικειμένων.
- **Πρότυπα συμπεριφοράς:** Για τον καθορισμό του τρόπου αλληλεπίδρασης μεταξύ αντικειμένων.

Τέλος, για να επιλέξει κάποιος προγραμματιστής το μοτίβο ανάμεσα σε παραπάνω από 20 μοτίβα που ταιριάζει στο πρόβλημα του, προτείνεται:

- Να εξετάσει πως ένα μοτίβο επιλύει ένα σχεδιαστικό πρόβλημα.
- Να διαβάσει τις ενότητες πρόθεσης.
- Να μελετήσει το πως τα σχεδιαστικά πρότυπα αλληλεπιδρούν.
- Να μελετήσει μοτίβα με παρόμοιο σκοπό.
- Να εξετάσει τις αιτίες επανασχεδιασμού του έργου του.
- Να σκεφτεί τι πρέπει να αλλάξει στο έργο του.



## ΚΕΦΑΛΑΙΟ 3

### Ανάλυση Απαιτήσεων

Στο κεφάλαιο αυτό θα δοθούν οι ιστορίες χρήστη [5] που αφορούν το εργαλείο, σε μορφή πινάκων, καθώς και οι περιπτώσεις χρήσης του [5].

#### 3.1 Ιστορίες Χρήστη

Οι ιστορίες χρήστη αποτελούν άτυπες περιγραφές των χαρακτηριστικών του εργαλείου μας και των δυνατοτήτων του σε φυσική γλώσσα. Γράφονται από την πλευρά του χρήστη του εργαλείου σε μορφή καρτών [5].

Ιστορία Χρήστη	Σαν [τύπος χρήστη]	Θέλω να [πραγματοποιήσω ένα έργο]	Ώστε να μπορώ [να πετύχω έναν στόχο]
IX1	Προγραμματιστής	Να μπορώ να επιλέξω κατηγορία μοτίβων.	Έτσι ώστε να εισάγω αυτόματα στον κώδικά μου το σκελετό ενός μοτίβου της κατηγορίας αυτής.
IX2	Προγραμματιστής	Να μπορώ να επιλέγω ένα μοτίβο μιας κατηγορίας.	Έτσι ώστε να εισάγω αυτόματα στον κώδικά μου το σκελετό του μοτίβου αυτού.
IX3	Προγραμματιστής	Να μπορώ να καθορίσω τα ονόματα των κλάσεων που θα δημιουργηθούν αυτόματα.	Έτσι ώστε να προσαρμόσω τις κλάσεις αυτές στον κώδικά μου και στις ανάγκες του μοτίβου.
IX4	Προγραμματιστής	Να μπορώ να καθορίσω πεδία που θα προστεθούν στις νέες κλάσεις.	Έτσι ώστε να προσαρμόσω τις κλάσεις αυτές στον κώδικά μου και στις ανάγκες του μοτίβου.
IX5	Προγραμματιστής	Να μπορώ να καθορίσω μεθόδους που θα προστεθούν στις νέες κλάσεις.	Έτσι ώστε να προσαρμόσω τις μεθόδους αυτές στον κώδικά μου και στις ανάγκες του μοτίβου.
IX6	Προγραμματιστής	Να μπορώ να δημιουργήσω αυτόματα τον κώδικα του μοτίβου με βάση τις όποιες παραμετροποιήσεις έχουν γίνει.	Έτσι ώστε να εισάγω αυτόματα στον κώδικά μου το σκελετό του μοτίβου.
IX7	Προγραμματιστής	Να μπορώ να καθορίσω τα ονόματα των διεπαφών που θα δημιουργηθούν αυτόματα.	Έτσι ώστε να προσαρμόσω τις διεπαφές αυτές στον κώδικά μου και στις ανάγκες του μοτίβου.
IX8	Προγραμματιστής	Να μπορώ να καθορίσω μεθόδους που θα προστεθούν στις νέες διεπαφές.	Έτσι ώστε να προσαρμόσω τις διεπαφές αυτές στον κώδικά μου και στις ανάγκες του μοτίβου.
IX9	Προγραμματιστής	Να μπορώ να ακυρώσω την διαδικασία.	Έτσι ώστε να επιστρέψω σε αυτό που έκανα χωρίς να αλλάξω τον κώδικά μου.
IX10	Προγραμματιστής	Να μπορώ να προσθέσω νέες κλάσεις.	Έτσι ώστε να προσαρμόσω το μοτίβο στον κώδικά μου.
IX11	Προγραμματιστής	Να μπορώ να καθορίσω ποιες διεπαφές θα υλοποιούν οι νέες κλάσεις.	Έτσι ώστε να προσαρμόσω τις κλάσεις αυτές στον κώδικά μου και στις ανάγκες του μοτίβου.

Πίνακας 3.1: Ιστορίες χρήστη.

## 3.2 Περιπτώσεις χρήσης

Οι Περιπτώσεις Χρήσης [5] αφορούν σύνολα διαδοχικών ενεργειών που προσδιορίζουν τη συμπεριφορά του συστήματος και τις λειτουργικές του απαιτήσεις. Αποτελούν μία πιο λεπτομερειακή προσέγγιση των ιστοριών χρήστη [5]. Κάθε περίπτωση χρήσης πρέπει να διαθέτει τουλάχιστον έναν Actor, κάποιον δηλαδή, που παίζει έναν ρόλο και αλληλεπιδρά με το σύστημα με τον τρόπο που ορίζει το περιεχόμενο

της περίπτωσης χρήσης.



Σχήμα 3.1: Διάγραμμα UML Περιπτώσεων Χρήστη.

<b>Περίπτωση χρήσης:</b> Επιλογή κατηγορίας μοτίβου.
<b>Αναγνωριστικό:</b> ΠΧ1
<b>Προϋποθέσεις:</b> <ol style="list-style-type: none"> <li>1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει το έργο που θα εργαστεί.</li> </ol>
<b>Ροή γεγονότων:</b> <ol style="list-style-type: none"> <li>1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής επιλέξει Import pattern, κάτω από το μενού Design pattern builder στο παράθυρο New του eclipse.</li> <li>2. Το σύστημα εμφανίζει έναν οδηγό.</li> <li>3. Ο προγραμματιστής επιλέγει την κατηγορία μοτίβου που επιθυμεί.</li> </ol>
<b>Μετα-συνθήκες:</b> Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.2: Επιλογή κατηγορίας μοτίβου.

<b>Περίπτωση χρήσης:</b> Επιλογή μοτίβου
<b>Αναγνωριστικό:</b> ΠΧ2
<b>Προϋποθέσεις:</b> <ol style="list-style-type: none"> <li>1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει κατηγορία μοτίβου.</li> </ol>
<b>Ροή γεγονότων:</b> <ol style="list-style-type: none"> <li>1. Η περίπτωση χρήσης ξεκινάει όταν ο προγραμματιστής κάνει κλικ στην πτυσσόμενη λίστα.</li> <li>2. Το σύστημα εμφανίζει τα διαθέσιμα μοτίβα.</li> <li>3. Ο προγραμματιστής επιλέγει το μοτίβο που επιθυμεί.</li> </ol>
<b>Μετα-συνθήκες:</b> Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.3: Επιλογή μοτίβου.

<b>Περίπτωση χρήσης:</b> Καθορισμός μεθόδων κλάσης
<b>Αναγνωριστικό:</b> ΠΧ3
<p><b>Προϋποθέσεις:</b></p> <ol style="list-style-type: none"> <li>1. Ο προγραμματιστής χρειάζεται να είναι στο παράθυρο επεξεργασίας της κλάσης.</li> </ol>
<p><b>Ροή γεγονότων:</b></p> <ol style="list-style-type: none"> <li>1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επεξεργασίας πεδίων της κλάσης.</li> <li>2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις μεθόδους της τρέχουσας κλάσης.</li> <li>3. Για κάθε μέθοδο: <ol style="list-style-type: none"> <li>(α) Ο προγραμματιστής επεξεργάζεται το όνομα της μεθόδου.</li> <li>(β) Ο προγραμματιστής επεξεργάζεται τον επιστρεφόμενο τύπο της μεθόδου.</li> <li>(γ) Ο προγραμματιστής επεξεργάζεται την ορατότητα της μεθόδου.</li> </ol> </li> <li>4. Ο προγραμματιστής κάνει κλικ στο κουμπί finish.</li> <li>5. Το σύστημα κλείνει το παράθυρο.</li> </ol>
<p><b>Μετα-συνθήκες:</b></p> <ol style="list-style-type: none"> <li>1. Το σύστημα ρυθμίζει το όνομα της κλάσης.</li> <li>2. Το σύστημα ρυθμίζει τα πεδία της κλάσης.</li> <li>3. Το σύστημα ρυθμίζει τις μεθόδους της κλάσης.</li> </ol>

Πίνακας 3.4: Καθορισμός μεθόδων κλάσης.

Περίπτωση χρήσης: Ονοματοδοσία κλάσης.
Αναγνωριστικό: ΠΧ4
<p>Προϋποθέσεις:</p> <ol style="list-style-type: none"> <li>1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει μοτίβο.</li> </ol>
<p>Ροή γεγονότων:</p> <ol style="list-style-type: none"> <li>1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επιλογής μοτίβου.</li> <li>2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις κλάσεις και τις διεπαφές του μοτίβου.</li> <li>3. Ο προγραμματιστής επιλέγει την κλάση που επιθυμεί.</li> <li>4. Ο προγραμματιστής κάνει κλικ στο κουμπί edit class.</li> <li>5. Το σύστημα εμφανίζει ένα νέο παράθυρο όπου ο χρήστης μπορεί να επεξεργαστεί το όνομα της κλάσης.</li> <li>6. Ο προγραμματιστής επεξεργάζεται το όνομα της κλάσης.</li> <li>7. Ο προγραμματιστής κάνει κλικ στο κουμπί Next.</li> <li>8. Το σύστημα εμφανίζει ένα νέο παράθυρο.</li> </ol>
<p>Μετα-συνθήκες:</p> <p>Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.</p>

Πίνακας 3.5: Ονοματοδοσία κλάσης.

Περίπτωση χρήσης: Καθορισμός πεδίων κλάσης
Αναγνωριστικό: ΠΧ5
<p>Προϋποθέσεις:</p> <ol style="list-style-type: none"> <li>1. Ο προγραμματιστής χρειάζεται να έχει επεξεργαστεί το όνομα της κλάσης</li> </ol>
<p>Ροή γεγονότων:</p> <ol style="list-style-type: none"> <li>1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επεξεργασίας ονόματος της κλάσης.</li> <li>2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τα πεδία της τρέχουσας κλάσης.</li> <li>3. Για κάθε μέθοδο: <ol style="list-style-type: none"> <li>(α) Ο προγραμματιστής επεξεργάζεται το όνομα του πεδίου.</li> <li>(β) Ο προγραμματιστής επεξεργάζεται τον τύπο του πεδίου.</li> <li>(γ) Ο προγραμματιστής επεξεργάζεται την ορατότητα του πεδίου.</li> </ol> </li> <li>4. Ο προγραμματιστής κάνει κλικ στο κουμπί Next.</li> <li>5. Το σύστημα εμφανίζει ένα νέο παράθυρο.</li> </ol>
Μετα-συνθήκες:

Πίνακας 3.6: Καθορισμός πεδίων κλάσης.

Περίπτωση χρήσης: Ονοματοδοσία διεπαφής
Αναγνωριστικό: ΠΧ6
<p>Προϋποθέσεις:</p> <ol style="list-style-type: none"> <li>1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει μοτίβο.</li> </ol>
<p>Ροή γεγονότων:</p> <ol style="list-style-type: none"> <li>1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επιλογής μοτίβου.</li> <li>2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις κλάσεις και τις διεπαφές του μοτίβου.</li> <li>3. Ο προγραμματιστής επιλέγει την διεπαφή που επιθυμεί.</li> <li>4. Ο προγραμματιστής κάνει κλικ στο κουμπί edit interface.</li> <li>5. Το σύστημα εμφανίζει ένα νέο παράθυρο όπου ο χρήστης μπορεί να επεξεργαστεί το όνομα της διεπαφής.</li> <li>6. Ο προγραμματιστής επεξεργάζεται το όνομα της διεπαφής.</li> <li>7. Ο προγραμματιστής κάνει κλικ στο κουμπί Next.</li> <li>8. Το σύστημα εμφανίζει ένα νέο παράθυρο.</li> </ol>
Μετα-συνθήκες:

Πίνακας 3.7: Ονοματοδοσία διεπαφής.



<b>Περίπτωση χρήσης:</b> Καθορισμός μεθόδων διεπαφής
<b>Αναγνωριστικό:</b> ΠΧ7
<p><b>Προϋποθέσεις:</b></p> <ol style="list-style-type: none"> <li>1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επεξεργασίας ονόματος της διεπαφής.</li> <li>2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις μεθόδους της τρέχουσας διεπαφής.</li> <li>3. Για κάθε μέθοδο: <ol style="list-style-type: none"> <li>(α) Ο προγραμματιστής επεξεργάζεται το όνομα της μεθόδου.</li> <li>(β) Ο προγραμματιστής επεξεργάζεται τον επιστρεφόμενο τύπο της μεθόδου.</li> <li>(γ) Ο προγραμματιστής επεξεργάζεται την ορατότητα της μεθόδου.</li> </ol> </li> <li>4. Ο προγραμματιστής κάνει κλικ στο κουμπί finish.</li> <li>5. Το σύστημα κλείνει το παράθυρο.</li> </ol>
<p><b>Μετα-συνθήκες:</b></p> <ol style="list-style-type: none"> <li>1. Το σύστημα ρυθμίζει το όνομα της διεπαφής.</li> <li>2. Το σύστημα ρυθμίζει τις μεθόδους της διεπαφής.</li> </ol>

Πίνακας 3.8: Καθορισμός μεθόδων διεπαφής.

<b>Περίπτωση χρήσης:</b> Εξαγωγή σκελετού επιλεγμένου μοτίβου.
<b>Αναγνωριστικό:</b> PX8
<b>Προϋποθέσεις:</b> <ol style="list-style-type: none"> <li>1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει κατηγορία μοτίβου.</li> </ol>
<b>Ροή γεγονότων:</b> <ol style="list-style-type: none"> <li>1. Η Περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί finish στο παράθυρο επιλογής κλάσης ή διεπαφής.</li> <li>2. Το σύστημα δημιουργεί τα απαραίτητα πηγαία αρχεία java στο επιλεγμένο πακέτο.</li> <li>3. Το σύστημα τερματίζει.</li> </ol>
<b>Μετα-συνθήκες:</b> Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.9: Εξαγωγή σκελετού επιλεγμένου μοτίβου.

<b>Περίπτωση χρήσης:</b> Προσθήκη νέας κλάσης
<b>Αναγνωριστικό:</b> PX10
<b>Προϋποθέσεις:</b> <ol style="list-style-type: none"> <li>1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει μοτίβο.</li> <li>2. Το μοτίβο πρέπει να επιτρέπει την εισαγωγή καινούργιας κλάσης.</li> </ol>
<b>Ροή γεγονότων:</b> <ol style="list-style-type: none"> <li>1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επιλογής μοτίβου.</li> <li>2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις κλάσεις και τις διεπαφές του μοτίβου.</li> <li>3. Ο προγραμματιστής Κάνει κλικ στο κουμπί Add Class.</li> <li>4. Το σύστημα προσθέτει μία νέα κλάση.</li> </ol>
<b>Μετα-συνθήκες:</b> Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.10: Προσθήκη νέας κλάσης.

<b>Περίπτωση χρήσης:</b> Καθορισμός υλοποιημένης διεπαφής
<b>Αναγνωριστικό:</b> ΠΧ11
<b>Προϋποθέσεις:</b> <ol style="list-style-type: none"> <li>1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει μοτίβο.</li> </ol>
<b>Ροή γεγονότων:</b> <ol style="list-style-type: none"> <li>1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επιλογής μοτίβου.</li> <li>2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις κλάσεις και τις διεπαφές του μοτίβου.</li> <li>3. Ο προγραμματιστής επιλέγει μία κλάση που έχει προσθέσει ο ίδιος.</li> <li>4. Ο προγραμματιστής κάνει κλικ στο κουμπί edit class.</li> <li>5. Το σύστημα εμφανίζει ένα νέο παράθυρο όπου ο χρήστης μπορεί να επεξεργαστεί την διεπαφή που μπορεί να υλοποιεί η κλάση αυτή.</li> <li>6. Ο προγραμματιστής επεξεργάζεται την διεπαφή που μπορεί να υλοποιεί η κλάση αυτή.</li> <li>7. Ο προγραμματιστής κάνει κλικ στο κουμπί Next.</li> <li>8. Το σύστημα εμφανίζει ένα νέο παράθυρο.</li> </ol>
<b>Μετα-συνθήκες:</b> Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.11: Καθορισμός υλοποιημένης διεπαφής.

## ΚΕΦΑΛΑΙΟ 4

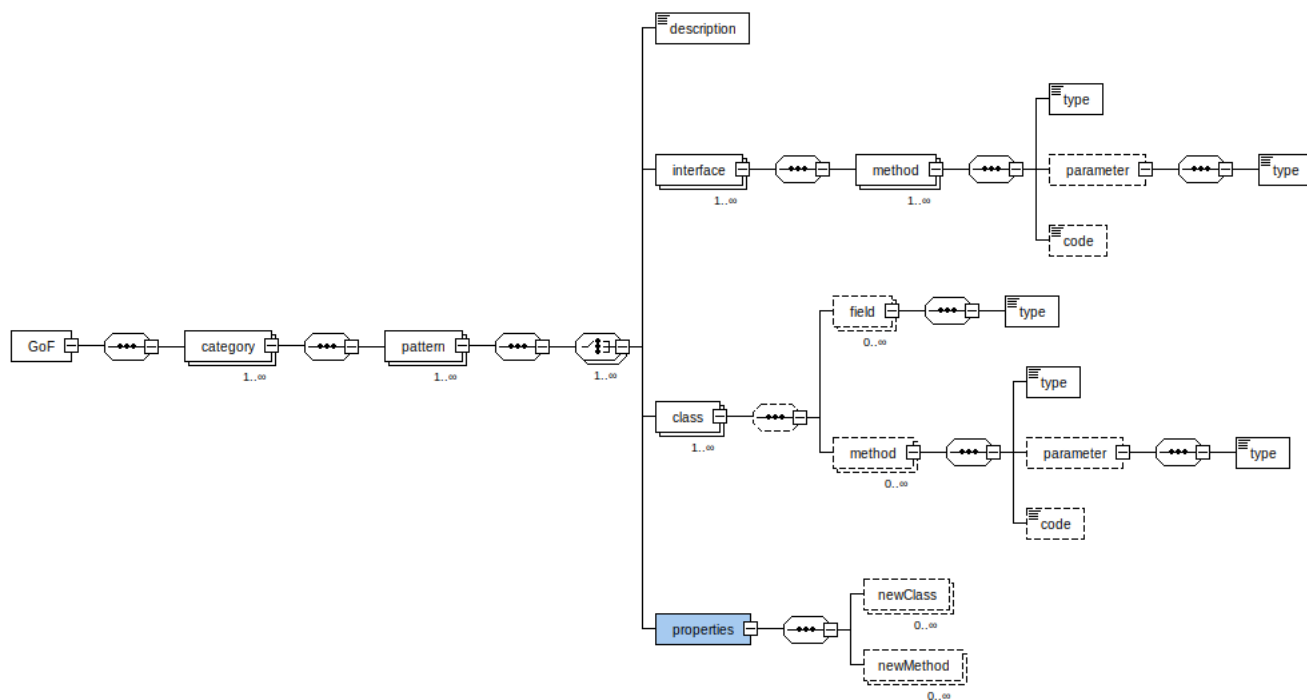
### Σχεδίαση και αρχιτεκτονική λογισμικού

Στη ενότητα αυτή, θα περιγραφούν οι βασικές αρχές λειτουργίας του Design Pattern Builder, το οποίο μπορεί κάποιος να βρει στο [github](#). Πρόκειται για ένα εργαλείο το οποίο αποτελεί επέκταση του Eclipse και έχει υλοποιηθεί σε γλώσσα Java. Αποτελείται από δύο τμήματα, το ένα κομμάτι αποτελεί τον πηγαίο κώδικα του εργαλείου. Το άλλο τμήμα του Design Pattern Builder είναι ένα αρχείο στο οποίο περιγράφουμε με δομημένο τρόπο κάθε σχεδιαστικό μοτίβο των GoF [1]. Ουσιαστικά, δημιουργήσαμε ένα εργαλείο, το οποίο έχει την δυνατότητα να παράξει πηγαίο κώδικα Java, διαβάζοντας ένα αρχείο στο οποίο υπάρχουν οι περιγραφές κάθε κλάσης και διεπαφής για κάθε μοτίβο των GoF [1]. Καθώς επιθυμούσαμε να υπάρχει η δυνατότητα δομημένης περιγραφής για την κάθε κατηγορία και κάθε μοτίβο, επιλέξαμε να χρησιμοποιήσουμε την γλώσσα περιγραφής xml. Για την περιγραφή της δομής κάθε μοτίβου, δημιουργήσαμε ένα αρχείο xml, το οποίο απεικονίζεται στην εικόνα 4.1, όπως και η περιγραφή του μοτίβου Singleton [1] το οποίο απεικονίζεται επίσης στο παράδειγμα 4.1. Έτσι, μπορεί κάποιος πολύ εύκολα να επεκτείνει τα διαθέσιμα μοτίβα, απλά τροποποιώντας το αρχείο αυτό, χωρίς να κάνει αλλαγές στον κώδικα, αλλά ούτε μεταγλώττιση του κώδικα και ελέγχους. Το εργαλείο διαβάζει το αρχείο xml, δημιουργεί εσωτερικά κατάλληλες δομές, ώστε στη συνέχεια, η γραφική διεπαφή να έχει την δυνατότητα να διαχειριστεί τις κλάσεις και διεπαφές του μοτίβου, όπως και τις μεθόδους και τα πεδία. Με αυτόν τον τρόπο, προσφέρεται η δυνατότητα στον χρήστη να αλληλεπιδρά με τα διάφορα μοτίβα. Καθώς ο προγραμματιστής αλληλεπιδρά με την γραφική διεπαφή, το σύστημα ενημερώνει συνεχώς τις δομές αυτές, ώστε όταν ο προγραμματιστής έχει παραμετροποιήσει κατάλληλα το μοτίβο, το σύστημα να είναι σε θέση να εξάγει τα πηγαία αρχεία, ώστε το τελικό

αποτέλεσμα να ανταποκρίνεται στις απαιτήσεις του προγραμματιστή.

Όταν ο χρήστης ανοίξει τον οδηγό, δηλαδή την γραφική διεπαφή του εργαλείου, το σύστημα διαβάζει το αρχείο xml και εμφανίζει στην γραφική διεπαφή τις διαθέσιμες κατηγορίες. Αφού ο προγραμματιστής διαλέξει κάποια κατηγορία, τότε το σύστημα εμφανίζει στον προγραμματιστή τα διαθέσιμα μοτίβα αυτής της κατηγορίας. Στη συνέχεια, ο προγραμματιστής μπορεί να προσθέσει νέα κλάση, εάν αυτό επιτρέπεται, να τροποποιήσει τις κλάσεις, δηλαδή να αλλάξει το όνομα τους, να επιλέξει κάποια διεπαφή για να υλοποιήσει, εάν είναι νέα κλάση και όχι κλάση του μοτίβου, τέλος να επεξεργαστεί τα υπάρχοντα, καθώς και να προσθέσει νέα πεδία και μεθόδους. Αφού ο προγραμματιστής κάνει τις αλλαγές που επιθυμεί και πατήσει finish, τότε το σύστημα θα ενημερώσει κατάλληλα τις δομές. Αντίστοιχα, ο προγραμματιστής μπορεί να τροποποιήσει και τις διεπαφές του μοτίβου και στη συνέχεια, το σύστημα με ανάλογο τρόπο ενημερώνει τις δομές του πεδίου εφαρμογής.

Το υποσύστημα της γραφικής διεπαφής είναι πλήρως εξαρτημένο από το API του Eclipse, καθώς είναι βασισμένο στην κλάση Wizard που προσφέρει το Eclipse και η οποία παρέχει τη λειτουργικότητα για τη δημιουργία προσαρμοσμένων οδηγών. Ο οδηγός είναι μια σειρά σελίδων που καθοδηγούν τον χρήστη σε μια σύνθετη εργασία. Τα κουμπιά Πίσω και Επόμενο επιτρέπουν στο χρήστη να μετακινείται προς τα εμπρός και προς τα πίσω στις σελίδες. Συνήθως, κάθε σελίδα συλλέγει μια πληροφορία. Όταν ο χρήστης κάνει κλικ στο κουμπί Τέλος, οι πληροφορίες χρησιμοποιούνται για την εκτέλεση μιας εργασίας. Ανά πάσα στιγμή, πριν από το κλικ στο κουμπί Finish, ο χρήστης μπορεί να ακυρώσει την εργασία, γεγονός που θα πρέπει να αναιρέσει τυχόν παρενέργειες των βημάτων που έχουν ολοκληρωθεί μέχρι στιγμής. Τέλος, ένα τμήμα του εργαλείου, το οποίο είναι υπεύθυνο για την δημιουργία των πηγαίων αρχείων, εξαρτάται από το API του Eclipse, καθώς χρησιμοποιεί, την κλάση PackageFragment. Η κλάση PackageFragment παρέχει διάφορες λειτουργίες που είναι απαραίτητες για την σωστή εξαγωγή των μοτίβων, όπως τον εντοπισμό του classpath για την προσθήκη των annotations, την εύρεση της τοποθεσίας όπου θα δημιουργηθούν τα πηγαία αρχεία, καθώς και την ίδια την δημιουργία ενός πηγαίου αρχείου.

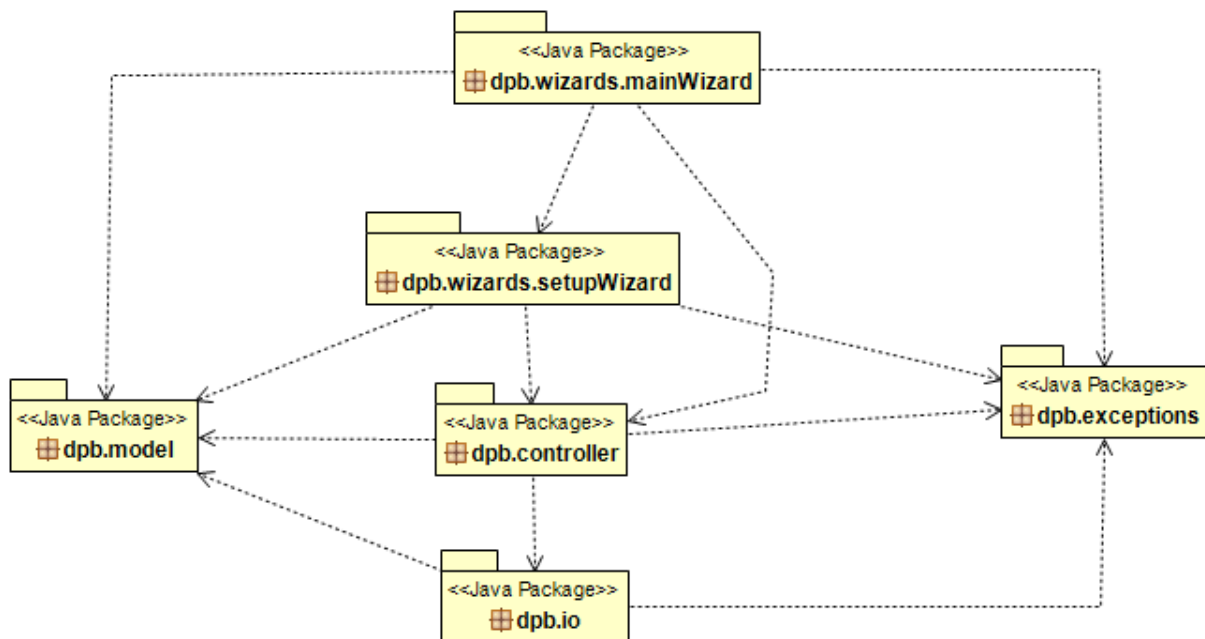


Σχήμα 4.1: Σχήμα αρχείου xml.

```
1 <pattern id="Singleton">
2   <description>
3     Ensure a class only has one instance,
4     and provide a global point of access to it.
5   </description>
6   <class id="Singleton" annotation="Singleton">
7     <field id="instance" isStatic="true">
8       <type>Singleton</type>
9     </field>
10    <method id="Singleton" visibility="private">
11      <type></type>
12    </method>
13    <method id="getInstance" isStatic="true">
14      <type>Object</type>
15      <code>
16        if (instance == null) {
17          instance = new Singleton();
18        }
19        return instance;
20      </code>
21    </method>
22  </class>
23 </pattern>
```

---

## 4.1 Αρχιτεκτονική



Σχήμα 4.2: Διάγραμμα UML Πακέτων συστήματος.

Το λογισμικό αποτελείται από τα παρακάτω πακέτα,

- **dpb.wizards.mainWizard:** Σε αυτό το πακέτο, περιέχονται οι κλάσεις που αφορούν το γραφικό περιβάλλον του Design Pattern Builder, πιο συγκεκριμένα, οι κλάσεις αυτού του πακέτου έχουν να κάνουν με την επιλογή του μοτίβου και την διαχείριση των κλάσεων και διεπαφών του επιλεγμένου μοτίβου.
- **dpb.wizards.setupWizards:** Σε αυτό το πακέτο, υπάρχουν οι κλάσεις που έχουν σχέση με την γραφική διεπαφή και την τροποποίηση των κλάσεων και διεπαφών, όπως και των μεθόδων και πεδίων.
- **dpb.controller:** Αποτελεί το πακέτο μέσω του οποίου η γραφική διεπαφή επικοινωνεί με το back-end. Χρησιμοποιεί τις λειτουργίες που παρέχει το back-end, ώστε να μετασχηματίζει τα ακατέργαστα δεδομένα των μοτίβων που παίρνει από το υποσύστημα io σε αντικείμενα κλάσεων που παρέχει το πακέτο model.
- **dpb.io:** Αποτελεί το κομμάτι του συστήματος το οποίο χαρακτηρίζεται ως back-end. και είναι υπεύθυνο για το διάβασμα του αρχείου xml στο οποίο

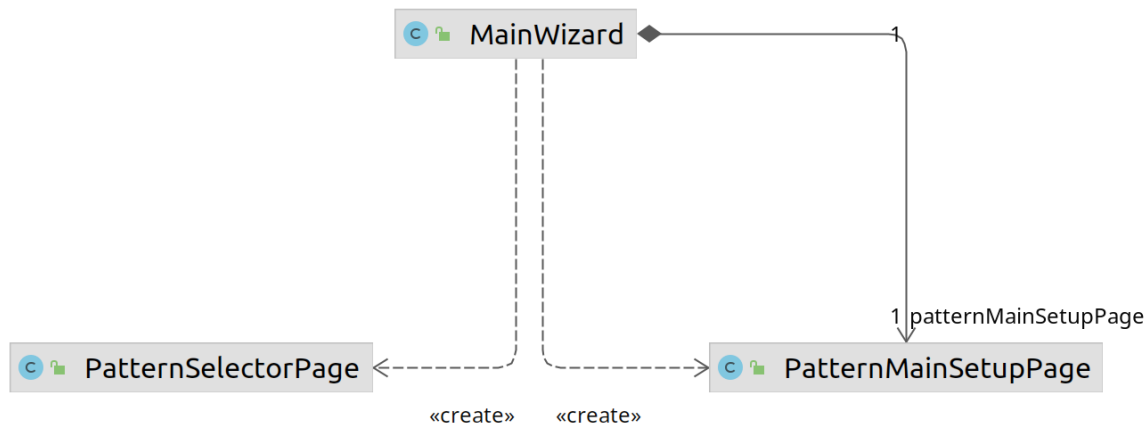


περιγράφονται τα μοτίβα. Παρέχει λειτουργίες για την εισαγωγή των μοτίβων στο σύστημα.

- **dpb.model:** Το πακέτο αυτό, περιέχει τις κλάσεις οι οποίες αναπαριστούν τα αντικείμενα του πεδίου του προβλήματος.
- **dpb.exceptions:** Τέλος, στο πακέτο αυτό υπάρχουν κάποιες κλάσεις οι οποίες αναπαριστούν εξαιρέσεις οι οποίες είναι κάποια γεγονότα που συμβαίνουν κατά την εκτέλεση του Design Pattern Builder και διακόπτουν την κανονική ροή του προγράμματος.

Το υποσύστημα `io` είναι υπεύθυνο για να διαβάζει το αρχείο `xml`. Δημιουργεί πίνακες από `Strings` με τις απαιτούμενες πληροφορίες. Αυτό σημαίνει ότι σε περίπτωση που ζητούνται οι κλάσεις ενός μοτίβου θα δημιουργηθεί ένας πίνακας με τα ονόματα των κλάσεων. Εάν, όμως, ζητούνται οι μέθοδοι μίας κλάσης θα δημιουργηθεί ένας πίνακας ο οποίος περιέχει για κάθε μέθοδο έναν πίνακα: με την ορατότητα της μεθόδου, τον επιστρεφόμενο τύπο της μεθόδου και το όνομα της. Αντίστοιχα δημιουργούνται πίνακες για τις διεπαφές και τα πεδία. Στη συνέχεια, το υποσύστημα του `controller`, χρησιμοποιεί τον `parser` για να αντλήσει τα ωμά δεδομένα και να τα μετασχηματίσει σε κατάλληλες δομές ώστε να μπορεί η γραφική διεπαφή να τις διαχειριστεί. Οι δομές αυτές είναι οι κλάσεις που αναπαριστούν το πεδίο της εφαρμογής μας, δηλαδή οι κλάσεις του πακέτου `model`.

## 4.2 dpb.wizards.mainWizard



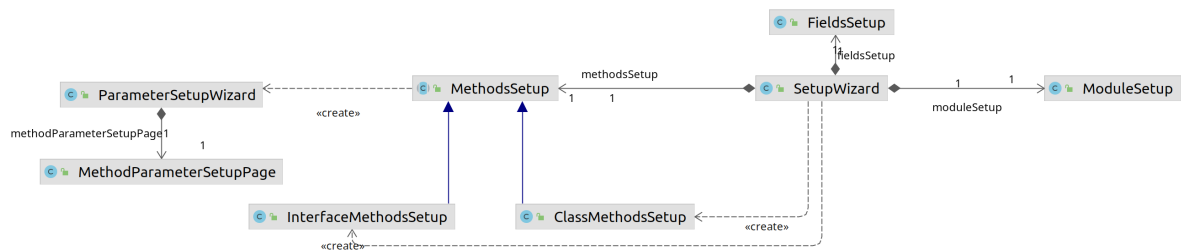
Σχήμα 4.3: Διάγραμμα UML Πακέτου mainWizard.

Το υποσύστημα αυτό, έχει να κάνει με τον κύριο οδηγό του Design Pattern Builder, ο οποίος αποτελείται από δύο σελίδες. Η πρώτη σελίδα, δίνει στον προγραμματιστή την δυνατότητα να επιλέξει κατηγορία και μοτίβο. Στην δεύτερη σελίδα ο προγραμματιστής μπορεί να περιηγηθεί ανάμεσα στις διαθέσιμες κλάσεις και διεπαφές, καθώς και να προσθέσει κάποια καινούργια κλάση, εάν αυτό είναι δυνατό από τους περιορισμούς που ορίζονται στην περιγραφή των μοτίβων, όπως και να επεξεργαστεί τις κλάσεις και τις διεπαφές. Η κλάση που υλοποιεί τον οδηγό, είναι η κλάση MainWizard, επεκτείνει την κλάση Wizard που παρέχει το Eclipse και υλοποιεί την διεπαφή INewWizard, η αρμοδιότητά της είναι να προσθέτει τις σελίδες του οδηγού αυτού. Χειρίζεται το γεγονός κατά το οποίο ο προγραμματιστής έχει τελειώσει με την ρύθμιση του μοτίβου και κάνει κλικ στο κουμπί Τέλος. Τότε καλείται η μέθοδος generate της κλάσης PatternGenerator, για κάθε κλάση και διεπαφή του μοτίβου, η οποία θα αναλυθεί στην ενότητα 4.4 και είναι υπεύθυνη για την παραγωγή των πηγαίων αρχείων.

Η κλάση PatternSelectonPage, η οποία επεκτείνει την κλάση WizardPage και υλοποιεί την διεπαφή IWizardPage, είναι η πρώτη σελίδα του κύριου μας οδηγού, αυτό που κάνει είναι να χρησιμοποιεί την κλάση PatternManager, η οποία θα αναλυθεί στην ενότητα 4.4, ώστε να αντλήσει τις κατηγορίες των μοτίβων και να τις εμφανίσει σε μία λίστα. Όταν επιλεγθεί μία κατηγορία από τον προγραμματιστή τότε ενεργοποιεί μία δεύτερη λίστα στην οποία προσθέτει τα μοτίβα αυτής της κατηγορίας.

Τέλος, η τελευταία κλάση του πακέτου αυτού είναι η κλάση `PatternMainSetupPage`, η οποία και αυτή υλοποιεί και επεκτείνει αντίστοιχα τα `IWizardPage` και `WizardPage`. Κύρια της δουλειά είναι η εμφάνιση των κλάσεων και διεπαφών του επιλεγμένου μοτίβου τα οποία αντλεί και αυτή με την χρήση του υποσυστήματος controller. Παρέχει κουμπιά για την προσθήκη νέας κλάσης, καθώς και την επεξεργασία των κλάσεων και διεπαφών. Όταν ο χρήστης κάνει κλικ σε κάποιο κουμπί επεξεργασίας, τότε δημιουργείται ο δεύτερος οδηγός της γραφικής διεπαφής ο οποίος θα αναλυθεί στην ενότητα 4.3.

### 4.3 dpb.wizards.setupWizards



Σχήμα 4.4: Διάγραμμα UML Πακέτου `setupWizards`.

Το υποσύστημα αυτό, αποτελείται από δύο οδηγούς, ο πρώτος οδηγός είναι υπεύθυνος για την ρύθμιση των ονομάτων κλάσεων και διεπαφών, για την επιλογή κάποιας διεπαφής που θα υλοποιεί μία νέα κλάση, καθώς και για την ρύθμιση των πεδίων και μεθόδων. Η κλάση που υλοποιεί τον οδηγό αυτό, είναι η κλάση `SetupWizard`, προσθέτει τις κατάλληλες σελίδες στον οδηγό, δηλαδή εάν ο προγραμματιστής θέλει να επεξεργαστεί μία κλάση τότε θα προσθέσει τρεις σελίδες, αλλιώς εάν επεξεργάζεται διεπαφή δύο σελίδες. Η τελευταία σελίδα είναι παρόμοια και για τις δύο περιπτώσεις, αφορά την επεξεργασία μεθόδων και υλοποιείται από την κλάση `MethodsSetup`. Εμφανίζει στον προγραμματιστή όλες τις μεθόδους που ανήκουν στην κλάση ή διεπαφή που επεξεργάζεται αυτήν την στιγμή. Ο προγραμματιστής έχει την δυνατότητα να αλλάξει το όνομα, την ορατότητα, καθώς και τον επιστρεφόμενο τύπο κάθε μεθόδου, επίσης έχει την δυνατότητα να προσθέσει ή διαγράψει κάποια μέθοδο. Ακόμα μπορεί να επεξεργαστεί τις παραμέτρους κάποιας μεθόδου. Για την λειτουργία αυτή, αναπτύχθηκε ακόμα ένας οδηγός ο οποίος

υλοποιείται από την κλάση `ParameterSetupWizard`, έχει μία σελίδα, την `MethodParameterSetup`, η οποία εμφανίζει μία λίστα με τις παραμέτρους της επιλεγμένης μεθόδου. Ο προγραμματιστής, έχει την δυνατότητα να αλλάξει το όνομα και τύπο κάθε παραμέτρου, καθώς και να προσθέσει ή διαγράψει κάποια παράμετρο. Μόλις ο χρήστης αλλάξει κάποια παράμετρο αυτομάτως ενημερώνεται και το αντικείμενο τύπου `Parameter`. Αντίστοιχα μόλις ο προγραμματιστής παραμετροποιεί κάποια μέθοδο τότε αυτομάτως ενημερώνεται και η δομή `Method`, που αντιπροσωπεύει την μέθοδο αυτήν.

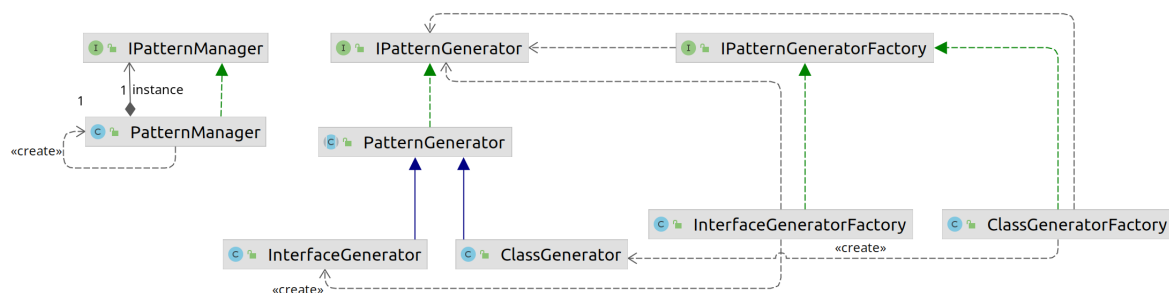
Ακόμα, η πρώτη σελίδα είναι κοινή, υλοποιείται από την κλάση `ModuleSetup`. Εάν πρόκειται για κλάση τότε περιέχει μία λίστα με τις διαθέσιμες διεπαφές που μπορεί να υλοποιήσει. Η δυνατότητα αυτή, είναι διαθέσιμη μόνο εάν πρόκειται για νέα κλάση που πρόσθεσε ο χρήστης στο μοτίβο. Η κλάση `ModuleSetup`, είναι υπεύθυνη και για τον χειρισμό του τερματισμού του οδηγού αυτού, από το κουμπί Τέλος ή την μετάβαση σε επόμενη σελίδα. Μόλις ο χρήστης πατήσει Τέλος ή Επόμενο, αντλούνται τα απαραίτητα δεδομένα από τα γραφικά στοιχεία εισόδου, που υπάρχουν στην σελίδα αυτή, όπως το όνομα, καθώς και το η διεπαφή που υλοποιεί η κλάση που επεξεργάζεται ο προγραμματιστής, εάν πρόκειται για κλάση και ενημερώνονται οι κατάλληλες δομές (`PatternClass` ή `PatternInterface`), τέλος, αναθέτουμε και το κατάλληλο annotation.

Η διαφοροποιημένη συμπεριφορά στην επεξεργασία των μεθόδων για τις κλάσεις και τις διεπαφές κάποιου μοτίβου, είναι στην λειτουργία προσθήκης νέας μεθόδου, αναλυτικότερα στην δομή `Method`, η οποία δομή θα αναλυθεί στην ενότητα 4.6, πιο συγκεκριμένα, αφορά την παράμετρο του constructor της δομής αυτής, η οποία ορίζει εάν αυτή η μέθοδος ανήκει σε διεπαφή ή κλάση. Οι μέθοδοι των διεπαφών έχουν το annotation της Java, `@Override`, όταν υλοποιούνται από κάποια κλάση και έτσι όταν ο χρήστης επεξεργάζεται κάποια διεπαφή, για να μην έχουμε διαφορετική κλάση που αναπαριστά σελίδα για την επεξεργασία μεθόδων διεπαφής, καθώς η υπόλοιπη λογική είναι ακριβώς ίδια, δημιουργήσαμε μία ιεραρχία κλάσεων. Στην ιεραρχία αυτή, η κλάση `ModuleSetup` είναι μία αφηρημένη κλάση, η οποία περιέχει την αφηρημένη μέθοδο για την προσθήκη νέας μεθόδου στην κλάση ή διεπαφή του μοτίβου που μεταβάλλει αυτήν την στιγμή ο προγραμματιστής, την κλάση αυτήν την επεκτείνουν, οι κλάσεις `InterfaceMethodsSetup` και `ClassMethodsSetup`, οι οποίες απλώς υλοποιούν την αφηρημένη μέθοδο με κατάλληλο τρόπο.

Τέλος, η δεύτερη σελίδα αφορά την επεξεργασία των πεδίων, υλοποιείται από

την κλάση FieldsSetup και είναι όμοια της σελίδας με τις μεθόδους. Εμφανίζεται στον οδηγό, μόνο στην περίπτωση που ο προγραμματιστής επεξεργάζεται κάποια κλάση του μοτίβου. Εμφανίζονται σε μία λίστα όλα τα πεδία και ο χρήστης μπορεί να προσθέσει/διαγράψει κάποιο πεδίο, όπως και να αλλάξει την ορατότητα, το όνομα και τον τύπο του πεδίου. Η ενημέρωση της σωστής δομής γίνεται και εδώ αμέσως μετά την επεξεργασία του πεδίου.

#### 4.4 dpb.controller



Σχήμα 4.5: Διάγραμμα UML Πακέτου controller.

Το υποσύστημα του controller, είναι αρμόδιο για τις βασικές λειτουργίες του Design Pattern Builder. Περιέχει την κλάση PatternManager, η οποία έχει ως σκοπό την παροχή των απαραίτητων πληροφοριών για κάποιο μοτίβο, καθώς και τα διαθέσιμα μοτίβα και κατηγορίες. Πιο συγκεκριμένα, δια μέσου αυτής της κλάσης η γραφική διεπαφή έχει διαθέσιμες τις δομές του πακέτου model, που περιγράφεται στην ενότητα 4.6, δηλαδή, παρέχει ως πίνακες από Strings τις κατηγορίες και τα μοτίβα, τα οποία παίρνει από το υποσύστημα io της ενότητας 4.5. Δημιουργεί αντικείμενα τύπου PatternClass αντλώντας το όνομα της κλάσης, την διεπαφή που υλοποιεί, τις μεθόδους και πεδία, καθώς και το εάν είναι αφηρημένη. Επίσης δημιουργεί αντικείμενα τύπου Method και Field, αντλώντας τις πληροφορίες από τον FileParser. Ακόμα, δημιουργεί και αντικείμενα τύπου PatternInterface, με όμοιο τρόπο, δηλαδή ανακτώντας το όνομα και τις μεθόδους από τον FileParser. Τέλος, προσφέρει στο γραφικό περιβάλλον, μεθόδους για την σωστή ενημέρωση των ονομάτων.

Στο υποσύστημα αυτό ανήκει μία ιεραρχία κλάσεων για την δημιουργία των πηγίων αρχείων Java. δημιουργήσαμε αυτήν την ιεραρχία, καθώς, ήταν απαραίτητο να εφαρμοστεί το μοτίβο Template Method [1], διότι η λειτουργία εξαγωγής των

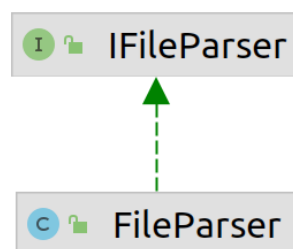
πηγαίων αρχείων χωρίστηκε σε κάποια βήματα, τα οποία είναι,

1. Η προσθήκη του πακέτου που βρίσκεται η συγκεκριμένη κλάση ή διεπαφή του μοτίβου.
2. Η προσθήκη των πεδίων μόνο στις κλάσεις.
3. Η προσθήκη των μεθόδων.
4. Η εξαγωγή του πηγαίου αρχείου που αφορά συγκεκριμένη κλάση ή διεπαφή του μοτίβου, με την χρήση του Eclipse API.

Τα βήματα 2 & 3, είναι διαφορετικά για την περίπτωση που δημιουργούμε πηγαίο αρχείο κλάσης και διαφορετικό για την περίπτωση που δημιουργούμε πηγαίο αρχείο διεπαφής, καθώς οι διεπαφές δεν έχουν πεδία και οι μέθοδοι των διεπαφών δεν έχουν σώμα. Ακόμα διαφορά παρουσιάζει και ο ορισμός της διεπαφής και της κλάσης. Άρα κρίνεται απαραίτητη η χρήση του μοτίβου Template Method [1]. Τέλος, η κλάση PatternGenerator, είναι υπεύθυνη και για την προσθήκη του πακέτου με τα annotations στο classpath του έργου του προγραμματιστή.

Τέλος, για την μείωση της σύζε ernGenerator ανάλογα την περίπτωση.

## 4.5 dpb.io

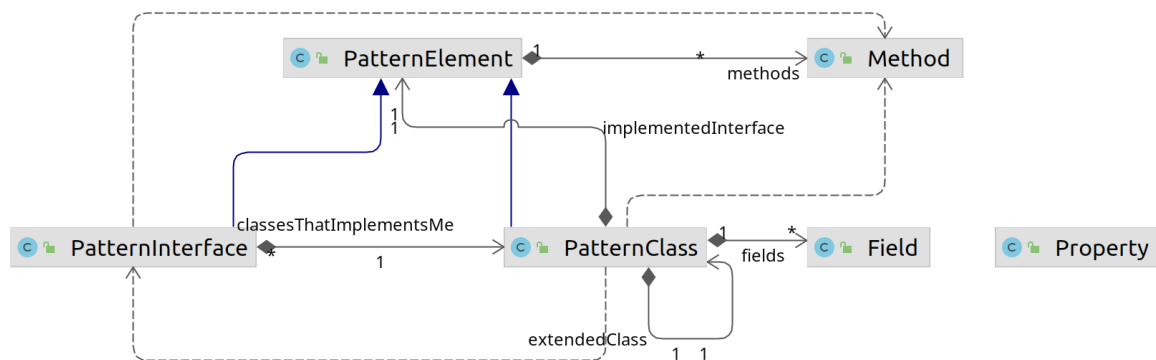


Σχήμα 4.6: Διάγραμμα UML Πακέτου io.

Το υποσύστημα αυτό, συνδέει το υπόλοιπο σύστημα με το αρχείο που περιγράφονται τα διάφορα μοτίβα. Η αρμοδιότητα του είναι να διαβάζει το αρχείο xml και να μεταφέρει τις απαιτούμενες πληροφορίες στον controller. Αυτό που κάνει είναι

να ανοίγει το αρχείο όπου περιγράφονται τα μοτίβα και να το διαβάσει. Είναι υπεύθυνο για την ανάκτηση των κατηγοριών και των μοτίβων, καθώς και την περιγραφή κάθε μοτίβου. Επίσης διαβάσει και μεταβιβάζει με την μορφή συμβολοσειρών τα ονόματα των κλάσεων και διεπαφών του μοτίβου, καθώς και τις ιδιότητες αυτών. Ακόμα, επιστρέφει το όνομα της διεπαφής που ενδέχεται να υλοποιεί μία κλάση, όπως και το όνομα κάποιας κλάσης που ενδέχεται να επεκτείνει και το αν μία κλάση είναι αφηρημένη. Επιστρέφει το όνομα, την ορατότητα και τον επιστρεφόμενο τύπο, εάν είναι στατική η μέθοδος, όπως και τις παραμέτρους κάθε μεθόδου μίας διεπαφής ή κλάσης του μοτίβου, επιπροσθέτως, εάν πρόκειται για κλάση επιστρέφει εάν η μέθοδος είναι αφηρημένη, καθώς και τον κώδικα που μπορεί να ορίζει το μοτίβο για την μέθοδο αυτή. Επιπλέον, διαβάσει το όνομα και τον τύπο κάθε πεδίου μίας κλάσης, καθώς και εάν κάποιο πεδίο είναι στατικό. Συμπληρωματικά, είναι υπεύθυνο για την διαβίβαση κάποιων ιδιοτήτων των μοτίβων, όπως ποιες διεπαφές είναι πιθανό να υλοποιεί κάποια νέα κλάση, καθώς και το εάν επιτρέπεται η προσθήκη νέας κλάσης όπως και το εάν επιτρέπεται η προσθήκη νέας μεθόδου και τέλος, τα annotations των νέων κλάσεων σύμφωνα με την διεπαφή που υλοποιούν. Τέλος, μεταβιβάζει στον controller τα annotations που χαρακτηρίζουν μία κλάση ή διεπαφή του μοτίβου.

## 4.6 dpb.model



Σχήμα 4.7: Διάγραμμα UML Πακέτου model.

Στο πακέτο αυτό, ορίζονται οι κλάσεις που αναπαριστούν τα αντικείμενα του πεδίου εφαρμογής.

Η κλάση `Method` αναπαριστά μία μέθοδο που ανήκει σε μία κλάση ή διεπαφή, έχει ως πεδία,

- Το όνομα της μεθόδου.
- Τον επιστρεφόμενο τύπο της μεθόδου.
- Τον τροποποιητή της μεθόδου.
- Την λίστα των παραμέτρων της μεθόδου.
- Εάν είναι στατική μέθοδος.
- Εάν είναι αφηρημένη μέθοδος.
- Που ανήκει η μέθοδος.
- Τον κώδικα της μεθόδου.
- Εάν γίνεται `Override` η μέθοδος.

Ακόμα περιέχει `getters` & `setters` και μία μέθοδο `equal`, για τον έλεγχο ισότητας με κάποιο άλλο αντικείμενο, καθώς χρειάζεται σε κάποιες περιπτώσεις.

Η κλάση `Field` αναπαριστά ένα πεδίο που ανήκει σε μία κλάση, έχει ως πεδία,

- Το όνομα του πεδίου.
- Τον τύπο του πεδίου.
- Τον τροποποιητή του πεδίου.
- Εάν είναι στατικό πεδίο.

Αντίστοιχα και εδώ περιέχει `getters` & `setters` και μία μέθοδο `equal`, για τον έλεγχο ισότητας με κάποιο άλλο αντικείμενο, καθώς χρειάζεται σε κάποιες περιπτώσεις.

Για την αναπαράσταση κλάσεων και διεπαφών, δημιουργήσαμε μία ιεραρχία, καθώς οι κλάσεις και οι διεπαφές έχουν πολλά κοινά χαρακτηριστικά, όπως, την κατηγορία μοτίβου και το μοτίβο που ανήκει μία κλάση/διεπαφή, το όνομα και την ορατότητα, καθώς και την λίστα με τις μεθόδους. Εκτός των κοινών χαρακτηριστικών έχουν και κάποιες διαφορές, για αυτόν τον λόγο προκύπτει η ανάγκη, να



δημιουργήσουμε μία μαμά κλάση που θα έχει τα κοινά χαρακτηριστικά τους, και δύο κλάσεις που κληρονομούν την μαμά κλάση PatternElement, την κλάση PatternClass, η οποία αντιπροσωπεύει μία κλάση του μοτίβου και την PatternInterface που αντιπροσωπεύει μία διεπαφή του μοτίβου. Η κλάση PatternClass, περιέχει ως πεδία, την διεπαφή που υλοποιεί, την κλάση που επεκτείνει, εάν είναι αφηρημένη και μία λίστα με τα πεδία που έχει κάποια κλάση του μοτίβου. Ενώ η κλάση PatternInterface έχει ως μοναδικό πεδίο μία λίστα με τις κλάσεις που υλοποιούν την συγκεκριμένη διεπαφή, ώστε να μπορούμε να προσθέσουμε τις μεθόδους της διεπαφής στις κλάσεις που την υλοποιούν. Τέλος, εκτός από τις μεθόδους get & set, έχουμε δημιουργήσει και μεθόδους οι οποίες προσθέτουν μία μέθοδο σε μία διεπαφή ή κλάση του μοτίβου, όπως και μεθόδους για την αφαίρεση κάποιας μεθόδου.

Τέλος, η κλάση Property, αναπαριστά τις ιδιότητες κάθε μοτίβου που έχουμε παρουσιάσει παραπάνω. Η κλάση αυτή, έχει δύο πεδία, το ένα πεδίο αφορά το annotation που θα έχει κάποια καινούργια κλάση/διεπαφή που υλοποιεί/επεκτείνει, κάποια διεπαφή ή άλλη κλάση και το δεύτερο πεδίο αφορά την κλάση/διεπαφή που μπορεί να επεκτείνει/υλοποιεί η νέα κλάση. Σαν μεθόδους έχει απλώς, getters & setters.

# ΚΕΦΑΛΑΙΟ 5

## Έλεγχος

Στο κεφάλαιο αυτό θα αναλυθούν οι έλεγχοι που υλοποιήθηκαν για τον κώδικα της εφαρμογής. Ο έλεγχος χωρίστηκε σε δύο επιμέρους κατηγορίες, στον έλεγχο του εργαλείου και στον έλεγχο των μοτίβων που υλοποιεί το εργαλείο. Για την υλοποίησή τους χρησιμοποιήθηκε η βιβλιοθήκη Junit 4.

### 5.1 Έλεγχος δομής σχεδιαστικών μοτίβων

#### 5.1.1 Έλεγχος μεθόδων φορτώματος μοτίβων

Σε αυτήν την ενότητα, θα παρουσιαστούν οι έλεγχοι για την δομή των μοτίβων τα οποία περιγράφονται στο αρχείο xml που δέχεται ως είσοδο το εργαλείο μας, Design Pattern Builder. Για να επιβεβαιώσουμε την σωστή δομή κάθε μοτίβου, το μόνο που χρειάζεται είναι να ελέγξουμε την κλάση, η οποία δημιουργεί τα απαραίτητα αντικείμενα αντλώντας δεδομένα από το αρχείο περιγραφής των μοτίβων, η κλάση αυτή είναι η PatternManager. Έχουμε δημιουργήσει τεστ τα οποία ελέγχουν εάν οι κλάσεις κάθε μοτίβου έχουν το σωστό όνομα, υλοποιούν ή επεκτείνουν την σωστή διεπαφή ή κλάση αντίστοιχα και περιέχουν τις σωστές μεθόδους και πεδία. Επίσης υπάρχουν τεστ τα οποία επιβεβαιώνουν την δομή των διεπαφών, ελέγχοντας το όνομα και τις μεθόδους κάθε διεπαφής. Οι περιπτώσεις ελέγχων χωρίστηκαν σε δύο πακέτα, στο ένα πακέτο υπάρχουν οι περιπτώσεις ελέγχου των κλάσεων κάθε μοτίβου και στο άλλο πακέτο υπάρχουν οι περιπτώσεις ελέγχου των διεπαφών κάθε μοτίβου. Πιο συγκεκριμένα, οι περιπτώσεις ελέγχου κατηγοριοποιούνται

σύμφωνα με την κατηγορία των μοτίβων που έχουν προτείνει οι GoF [1]. Στο πακέτο `dpb.patternManagerTests.getClassTests`, υπάρχουν οι εξής περιπτώσεις ελέγχου:

- **TestGetClassCreational:** Τα τεστ αυτής της κλάσης ελέγχουν όλα τα μοτίβα της κατηγορίας `creational`.
- **TestGetClassStructural:** Τα τεστ αυτής της κλάσης ελέγχουν όλα τα μοτίβα της κατηγορίας `structural`.
- **TestGetClassBehavioral:** Τα τεστ αυτής της κλάσης ελέγχουν όλα τα μοτίβα της κατηγορίας `behavioral`.

Για να επιβεβαιώσουμε την δομή κάθε κλάσης για κάθε μοτίβο καλούμε την μέθοδο `getClass(String, String)`, η οποία μας επιστρέφει μία λίστα με τις κλάσεις του συγκεκριμένου μοτίβου. Για κάθε κλάση ελέγχουμε το όνομα της και αν υλοποιεί κάποια διεπαφή, το όνομα της διεπαφής και αντίστοιχα εάν επεκτείνει κάποια άλλη κλάση. Στη συνέχεια, ελέγχουμε για κάθε πεδίο το όνομα του και τον τύπο του. Τέλος, για κάθε μέθοδο ελέγχουμε το όνομα της, τον επιστρεφόμενο τύπο, το όνομα και τον τύπο κάθε παραμέτρου και το σώμα της μεθόδου εάν υπάρχει.

Στο πακέτο `dpb.patternManagerTests.getInterfaceTests`, υπάρχουν οι εξής περιπτώσεις ελέγχου:

- **TestGetInterfaceCreational:** Τα τεστ αυτής της κλάσης ελέγχουν όλα τα μοτίβα της κατηγορίας `creational`.
- **TestGetInterfaceStructural:** Τα τεστ αυτής της κλάσης ελέγχουν όλα τα μοτίβα της κατηγορίας `structural`.
- **TestGetInterfaceBehavioral:** Τα τεστ αυτής της κλάσης ελέγχουν όλα τα μοτίβα της κατηγορίας `behavioral`.

Για να επιβεβαιώσουμε την δομή κάθε διεπαφής για κάθε μοτίβο καλούμε την μέθοδο `getClass(String, String)`, η οποία μας επιστρέφει μία λίστα με τις κλάσεις του συγκεκριμένου μοτίβου και στην συνέχεια την μέθοδο `getInterfaces()`. Για κάθε διεπαφή ελέγχουμε το όνομα της. Στη συνέχεια για κάθε μέθοδο ελέγχουμε το όνομα της, τον επιστρεφόμενο τύπο και το όνομα και τον τύπο κάθε παραμέτρου.

### 5.1.2 Έλεγχος μεθόδου ανάκτησης περιγραφής μοτίβου

Για τον έλεγχο της μεθόδου `getPatternDescription(String)` που ανήκει στην κλάση `dpb.controller.PatternManager`, υλοποιήσαμε την τεστ κλάση `TestGetDescription`, η οποία έχει τρεις περιπτώσεις ελέγχου, μία για κάθε κατηγορία. Για κάθε μοτίβο κάθε κατηγορίας, απλώς επιβεβαιώνουμε την περιγραφή του μοτίβου καλώντας την `getPatternDescription` και ελέγχοντας εάν είναι η αναμενόμενη περιγραφή.

## 5.2 Έλεγχος μεθόδων δημιουργίας πηγαίου κώδικα Java

Η ενότητα αυτή, έχει σκοπό να παρουσιάσει τους ελέγχους της μεθόδου, η οποία είναι υπεύθυνη για την δημιουργία των πηγαίων αρχείων Java που περιέχουν τις κλάσεις και τις διεπαφές του μοτίβου μαζί με τις διάφορες παραμετροποιήσεις που ενδέχεται να έχει κάνει ο προγραμματιστής. Η μέθοδος η οποία είναι υπεύθυνη για την εξαγωγή του πηγαίου κώδικα είναι η μέθοδος `generate(PatternElement)`, που ορίζεται στην διεπαφή `dpb.controller.IPatternGenerator`. Δεν είναι αναγκαίο να ελέγξουμε την μέθοδο `generate` εξονυχιστικά για κάθε μοτίβο, καθώς γνωρίζουμε την ορθότητα των μοτίβων από τα τεστ που περιγράφηκαν στην προηγούμενη ενότητα 5.1.1 (παρά μόνο για ένα τυχαίο μοτίβο). Διαλέξαμε το μοτίβο Singleton [1] της κατηγορίας `creational` καθώς έχει απλή δομή. Για την υλοποίησή των ελέγχων χρειάστηκε να δημιουργήσουμε μερικά mock αντικείμενα [5], διότι η μέθοδος `generate` έχει πεδίο ένα αντικείμενο τύπου `IPackageFragment`, το οποίο κατά την φάση του ελέγχου έχει τιμή `null`, καθώς αυτό το αντικείμενο αναπαριστά το πακέτο που έχει επιλέξει ο προγραμματιστής για την εξαγωγή όλων των πηγαίων αρχείων. Επίσης, είναι απαραίτητο, να κάνουμε `faking` [5] και την κλάση που υλοποιεί την διεπαφή `IPatternGenerator`. Κατά την δημιουργία ενός αντικειμένου της κλάσης αυτής καλείται η μέθοδος `addAnnotationsToClassPath()`, η οποία κατά τον έλεγχο δεν είναι απαραίτητη, καθώς προσθέτει στο `classpath` του εκάστοτε έργου τα `annotations`.

Για την κατασκευή των fake αντικειμένων [5], απλά κάναμε `override` τις απαραίτητες μεθόδους. Για την περίπτωση της κλάσης `ClassGenerator`, υλοποιήσαμε μία νέα κλάση στο πακέτο που έχουμε τους ελέγχους, η οποία επεκτείνει την κλάση `ClassGenerator`, και απλώς υλοποιεί την μέθοδο `addAnnotationsToClassPath()`, όπως φαίνεται στο παράδειγμα 5.1. Για την κατασκευή του fake αντικειμένου [5] `IPackageFragment`, δημιουργήσαμε μία κλάση, η οποία υλοποιεί την διεπαφή

IPackageFragment. Η κλάση αυτή έχει τρία πεδία τα οποία αναπαριστούν το όνομα του παραγόμενου πηγαίου αρχείου, τον κώδικα του αρχείου αυτού και το όνομα του πακέτου που θα εξαχθεί το πηγαίο αρχείο. Επίσης, τροποποιούμε την μέθοδο getElementName ώστε να επιστρέφει το πεδίο με το όνομα του πακέτου την createCompilationUnit, ώστε να θέτει το όνομα και τον κώδικα του πηγαίου αρχείου. Τέλος, προσθέσαμε δύο getters για το όνομα και τον κώδικα, ώστε να μπορούμε να επαληθεύσουμε αργότερα την ορθή λειτουργία της παραγωγής κώδικα όπως φαίνεται στο παράδειγμα 5.2.

Τέλος, αφού έχουμε δημιουργήσει τα απαραίτητα mock αντικείμενα [5], καλούμε την μέθοδο generate με παράμετρο την μοναδική κλάση του μοτίβου Singleton και επιβεβαιώνουμε την ορθή λειτουργία της μεθόδου generate ελέγχοντας τι επιστρέφουν οι getters.

```
1 package dpb.patternGeneratorTests;
2
3 import java.io.IOException;
4 import java.net.URISyntaxException;
5
6 import org.eclipse.core.runtime.CoreException;
7 import org.eclipse.jdt.core.JavaModelException;
8
9 import dpb.controller.ClassGenerator;
10
11 public class MockClassGenerator extends ClassGenerator {
12
13     public MockClassGenerator() throws CoreException,
14         URISyntaxException, IOException {
15         super();
16         // TODO Auto-generated constructor stub
17     }
18
19     @Override
20     protected void addAnnotationsToClassPath() throws
21         JavaModelException, URISyntaxException, IOException {
22         // TODO Auto-generated method stub
23     }
24 }
```

---

Παράδειγμα Κώδικα 5.2: Mock κλάση για την κλάση PackageFragment

```
1 public class MockPackageFragment implements IPackageFragment {
2     private final String elementName;
3     private String arg0;
4     private String arg1;
5
6     public MockPackageFragment(String elementName) {
7         this.elementName = elementName;
8     }
9
10    @Override
11    public String getElementName() {
12        return elementName;
13    }
14
15    @Override
16    public ICompilationUnit createCompilationUnit(String arg0,
17        String arg1, boolean arg2, IProgressMonitor arg3)
18        throws JavaModelException {
19        this.arg0 = arg0;
20        this.arg1 = arg1;
21        return null;
22    }
23
24    public String getArg0() {
25        return arg0;
26    }
27
28    public String getArg1() {
29        return arg1;
30    }
```

---

### 5.3 Λοιποί έλεγχοι

Τέλος, δημιουργήθηκαν μερικά ακόμα τεστ, για την επαλήθευση κάποιων μεθόδων τα οποία τεστ περιγράφουμε παρακάτω:

- **testGetPatternCategories:** Το τεστ αυτό επαληθεύει τις διαθέσιμες κατηγορίες μοτίβων που εμφανίζονται στον χρήστη.
- **testGetPatternsOfCategory:** Το τεστ αυτό επαληθεύει τα μοτίβα κάποιας κατηγορίας που είναι διαθέσιμα στον προγραμματιστή.
- **testUpdatePatternElementName:** Το τεστ αυτό επαληθεύει την λειτουργία αλλαγής ονόματος μίας κλάσης ή διεπαφής
- **testUpdateFieldName:** Το τεστ αυτό επαληθεύει την λειτουργία αλλαγής ονόματος ενός πεδίου.
- **testUpdateMethodName:** Το τεστ αυτό επαληθεύει την λειτουργία αλλαγής ονόματος κάποιας μεθόδου.
- **testGetProperties:** Το τεστ αυτό επαληθεύει τις ιδιότητες ενός μοτίβου, οι οποίες είναι, εάν το μοτίβο επιτρέπει την προσθήκη νέας κλάσης, ποιες διεπαφές μπορεί να υλοποιήσει μία νέα κλάση και τέλος το annotation κάθε κλάσης ή διεπαφής.



## 5.4 Αναφορά αποτελεσμάτων ελέγχων

### Test Report : AllTests (2) 20230626-201202.xml

- dpb.patternGeneratorTests.GenerateTests
  - [testGenerate](#)

#### Test Suite: dpb.patternManagerTests.PatternManagerTests

##### Results

Duration	0.003 sec
Tests	0
Failures	0

##### Tests

#### Test Suite: dpb.patternManagerTests.getInterfacesTest.GetInterfacesTests

##### Results

Duration	0.18 sec
Tests	0
Failures	0

##### Tests

#### Test Suite: dpb.patternManagerTests.getClassesTests.GetClassTests

##### Results

Duration	0.15 sec
Tests	0
Failures	0

##### Tests

#### Test Suite: dpb.patternGeneratorTests.GenerateTests

##### Results

Duration	0.006 sec
Tests	1
Failures	0

##### Tests

##### dpb.patternGeneratorTests.GenerateTests

Test case:	testGenerate
Outcome:	Passed
Duration:	0.006 sec
Failed	None

None

Σχήμα 5.1: Αναφορά αποτελεσμάτων ελέγχων

## ΚΕΦΑΛΑΙΟ 6

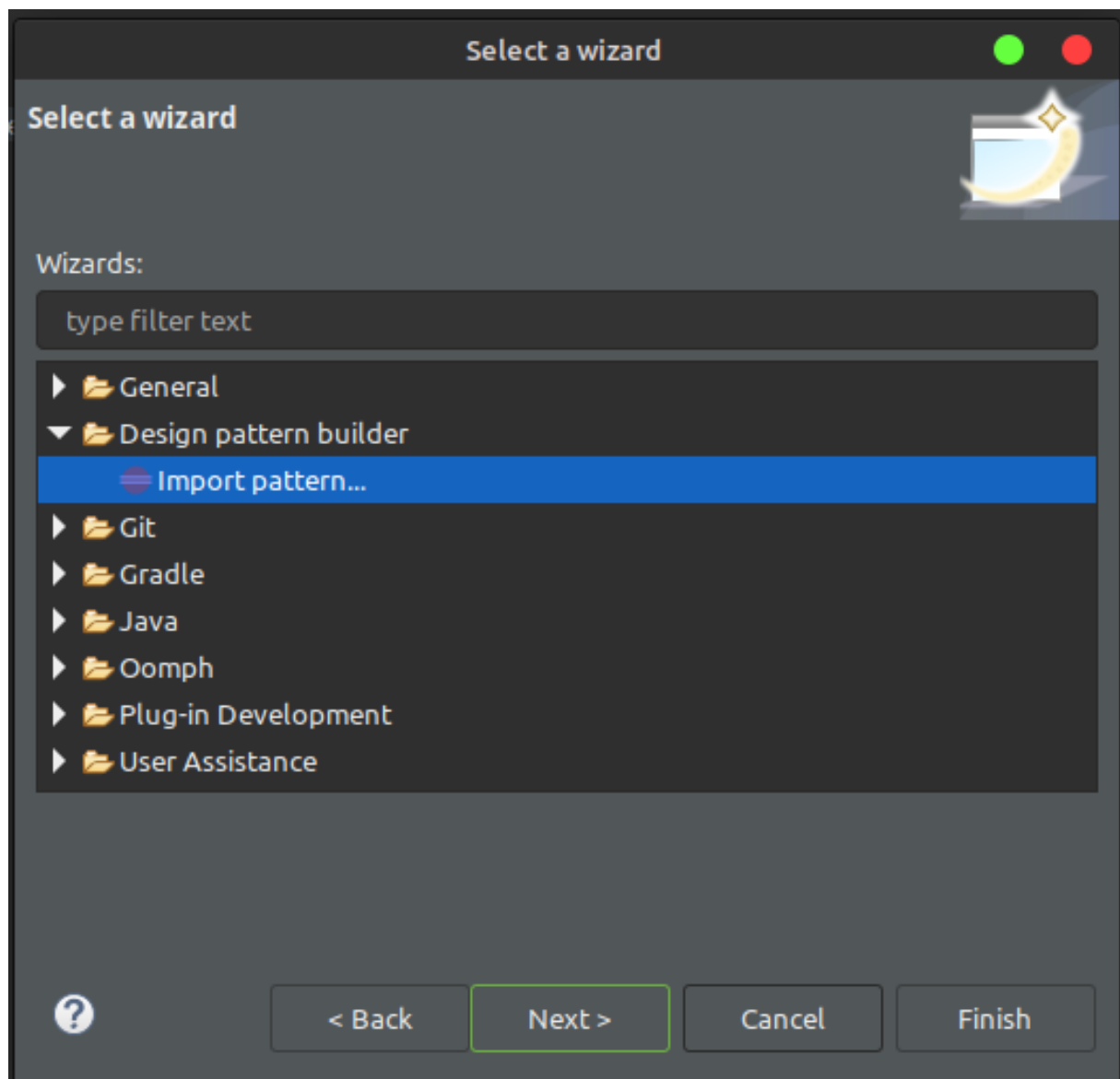
# Οδηγός Χρήσης Design Pattern Builder

### 6.1 Λειτουργίες χρήστη

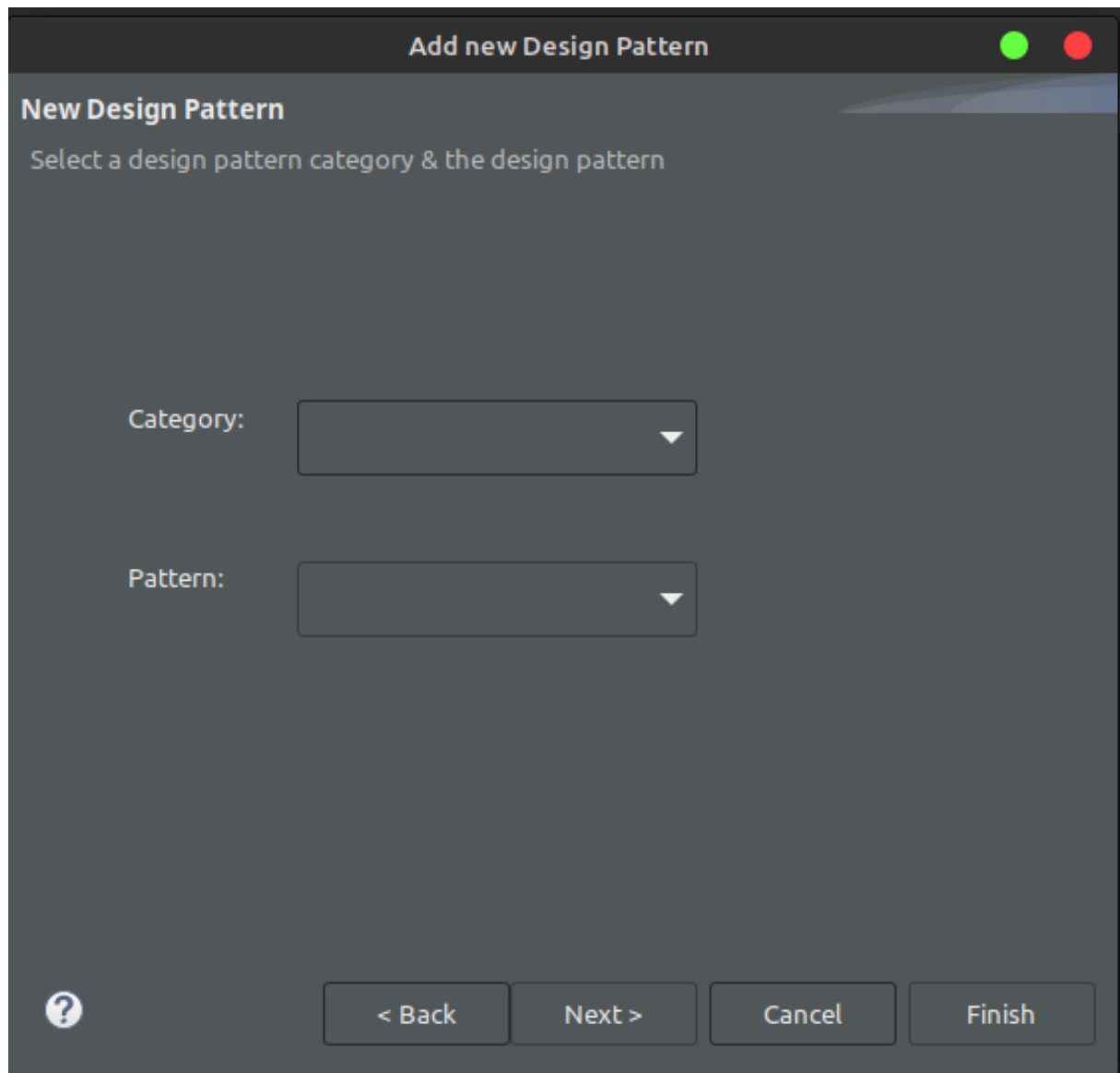
Για να έχει τη δυνατότητα ένας προγραμματιστής να χρησιμοποιήσει το Design Pattern Builder, θα χρειαστεί:

1. Να επιλέξει το πακέτο που επιθυμεί ή το τρέχων έργο στο οποίο εργάζεται.
2. Να κάνει δεξί κλικ.
3. Να επιλέξει New και στην καρτέλα Other θα βρει την επιλογή Design Pattern Builder.

όπως φαίνεται στην εικόνα 6.1, αφού επιλέξει Import Pattern θα εμφανιστεί ο οδηγός της εικόνας 6.2.



Σχήμα 6.1: Άνοιγμα οδηγού.




**Add new Design Pattern**

**New Design Pattern**

Select a design pattern category & the design pattern

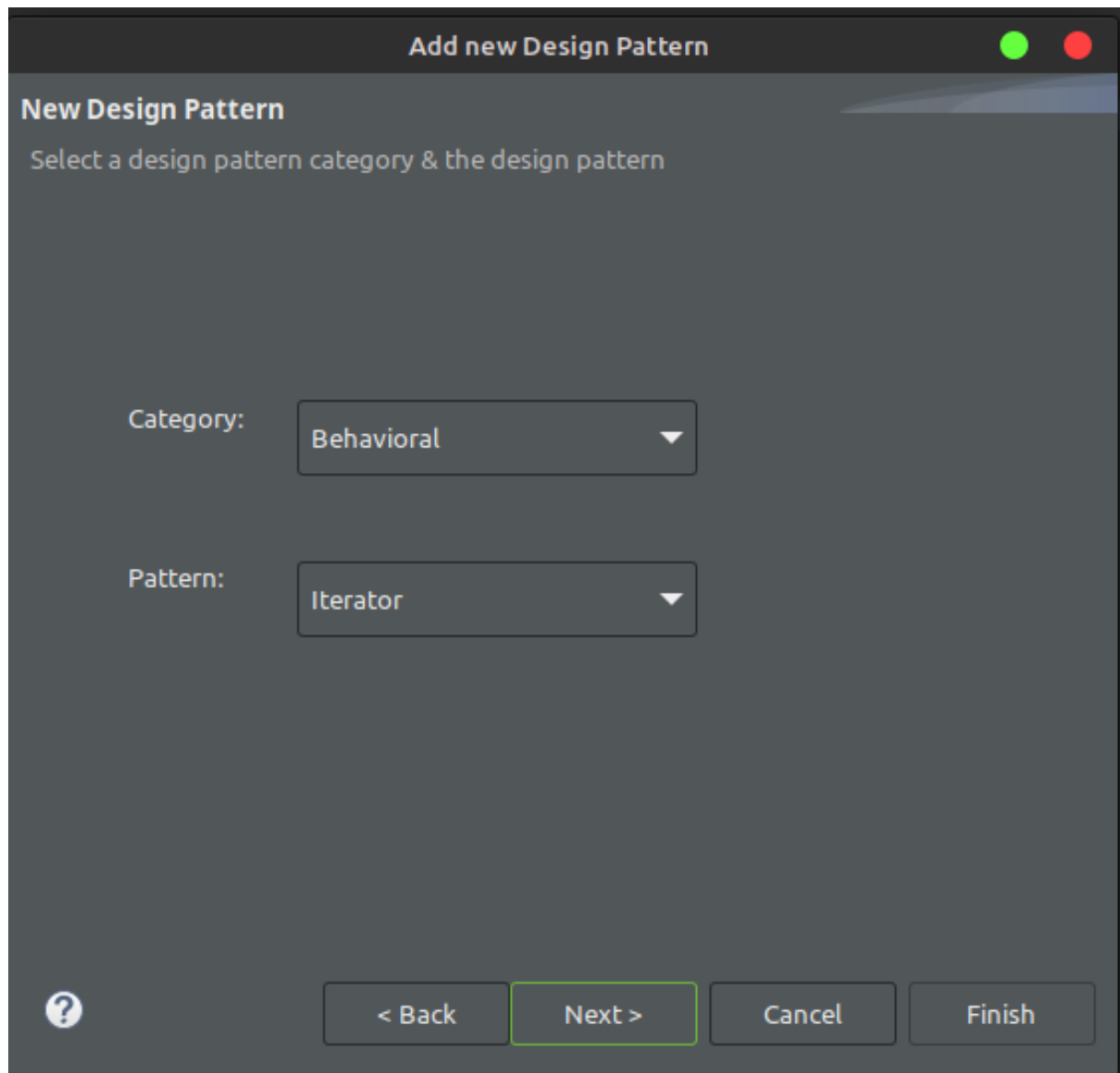
Category:

Pattern:



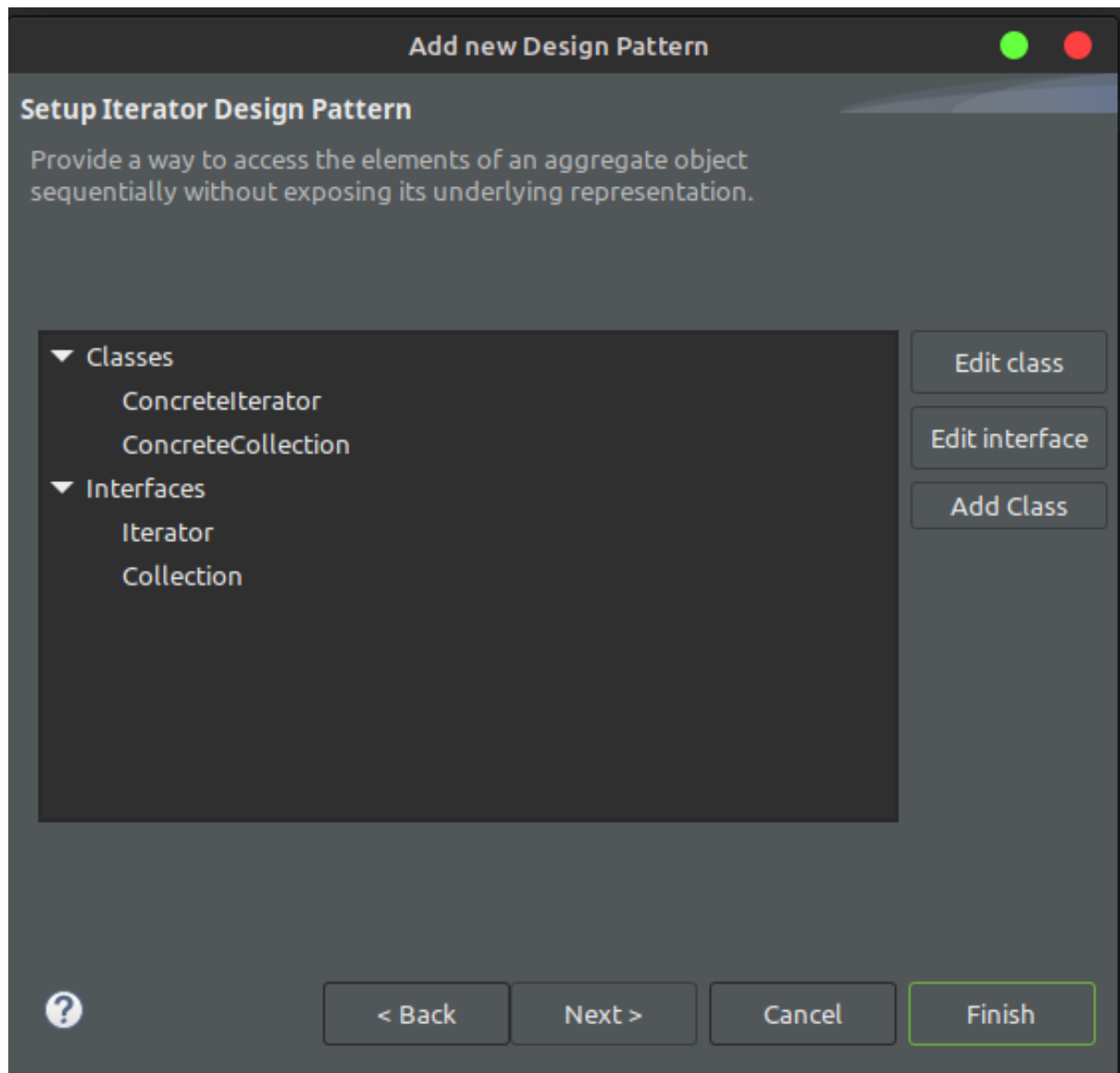
Σχήμα 6.2: Σελίδα επιλογής κατηγορίας & μοτίβου.

Στη συνέχεια, ο προγραμματιστής έχει την δυνατότητα να επιλέξει κατηγορία μοτίβου. Αφού επιλέξει κατηγορία, τότε θα γίνει διαθέσιμη και η επιλογή μοτίβου. Μόλις επιλέξει και μοτίβο, τότε θα μπορεί να πλοηγηθεί στην επόμενη σελίδα του οδηγού. Όπως φαίνεται στην εικόνα 6.3.



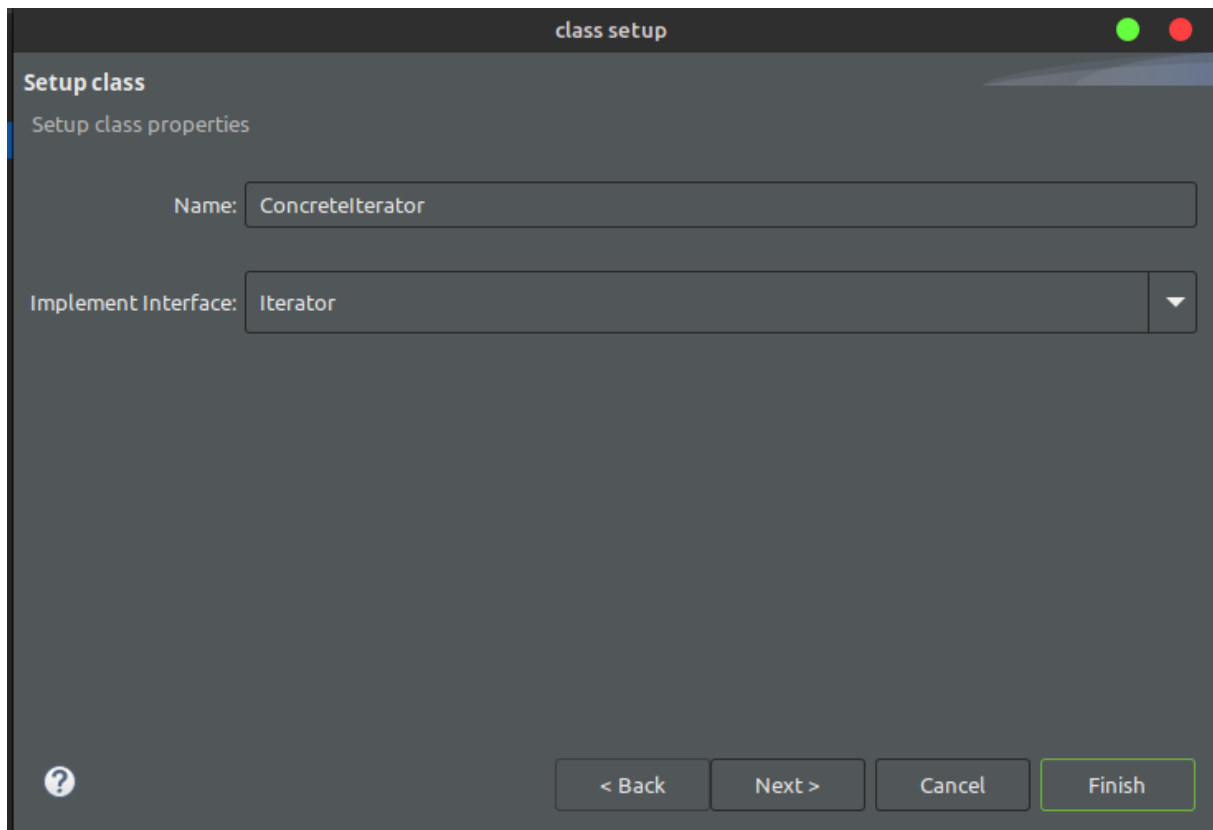
Σχήμα 6.3: Επιλογή μοτίβου.

Μόλις επιλέξει μοτίβο, ο οδηγός εμφανίζει τις κλάσεις και τις διεπαφές του μοτίβου. Στην εικόνα 6.4, βλέπουμε ένα παράδειγμα για το μοτίβο Iterator [1]. Στη συνέχεια, ο προγραμματιστής έχει την δυνατότητα, είτε να επεξεργαστεί κάποια κλάση ή διεπαφή, επιλέγοντας την από την λίστα και κάνοντας κλικ στο αντίστοιχο κουμπί, είτε να προσθέσει νέα κλάση, εάν αυτό είναι επιτρεπτό, είτε να ολοκληρώσει την διαδικασία χωρίς καμία τροποποίηση του μοτίβου κάνοντας κλικ στο κουμπί Finish.



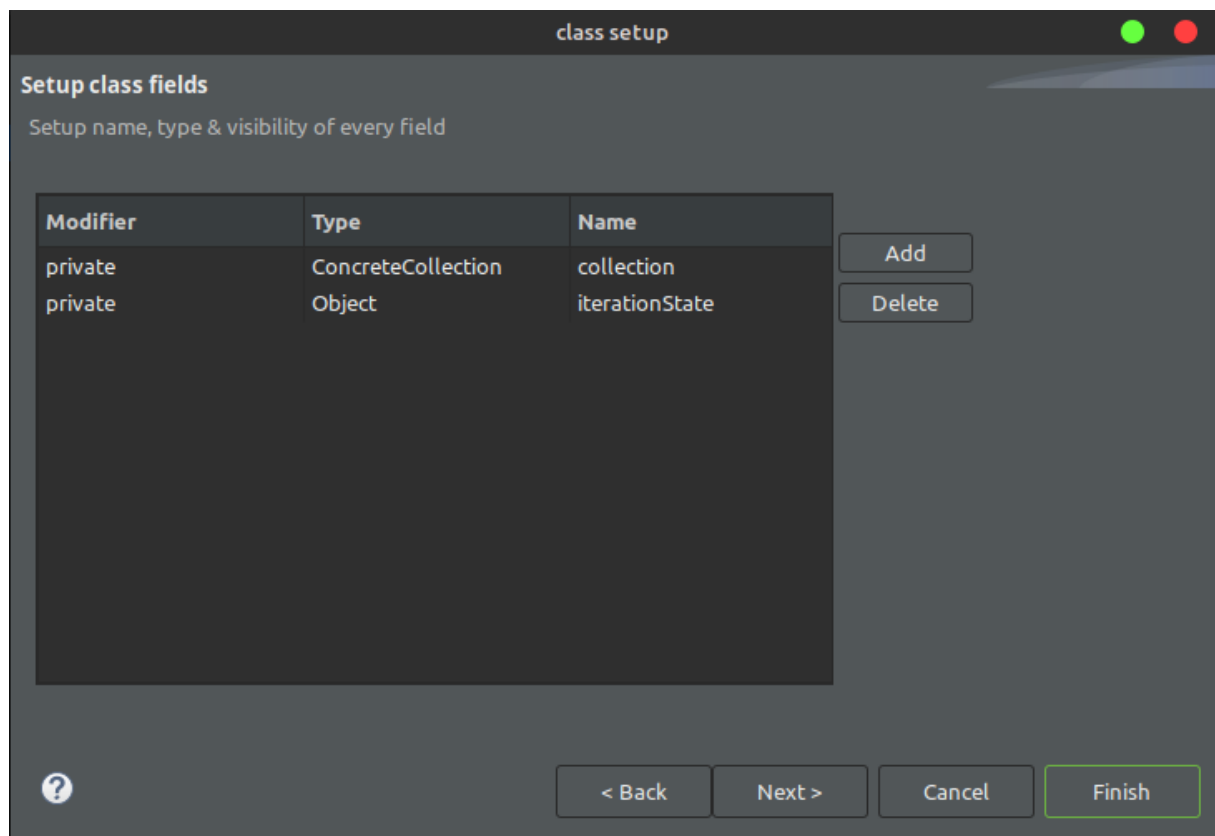
Σχήμα 6.4: Οι κλάσεις & διεπαφές του μοτίβου Iterator.

Αφού ο προγραμματιστής επιλέξει κάποια κλάση για να επεξεργαστεί, το σύστημα θα του εμφανίσει τη σελίδα 6.5, όπου μπορεί να αλλάξει το όνομα της κλάσης. Εάν πρόκειται για κλάση που έχει εισάγει ο χρήστης, να επιλέξει κάποια διεπαφή από τις διαθέσιμες για να υλοποιεί αυτή η κλάση. Επίσης, μπορεί να σταματήσει εδώ την επεξεργασία της κλάσης, κάνοντας κλικ στο κουμπί Finish.



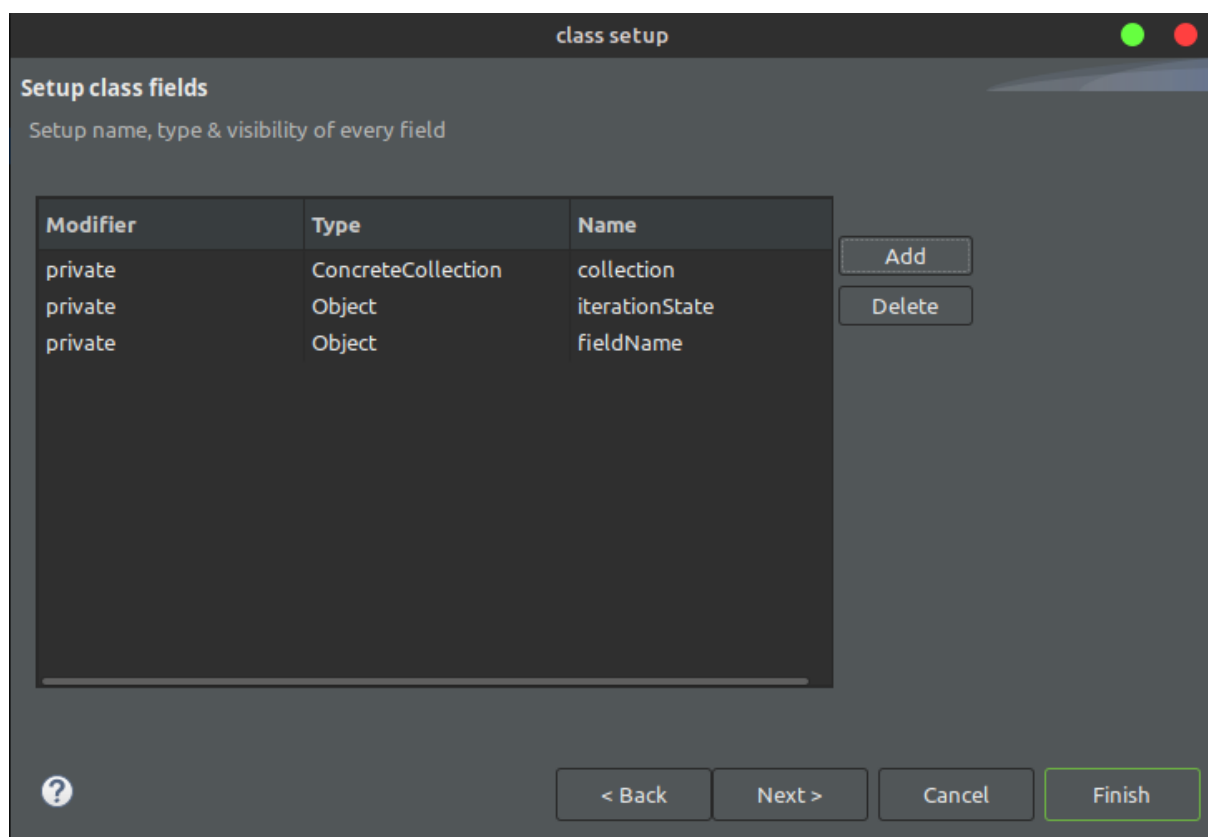
Σχήμα 6.5: Επεξεργασία ονόματος κλάσης.

Αφού έχει τροποποιήσει το όνομα της κλάσης, τότε στην επόμενη σελίδα 6.6, έχει την δυνατότητα να τροποποιήσει τα πεδία της κλάσης. Αυτό επιτυγχάνεται αλλάζοντας τις τιμές στα πεδία εισόδου. Επίσης, μπορεί να προσθέσει ή να διαγράψει κάποιο πεδίο όπως φαίνεται στην εικόνα 6.7.



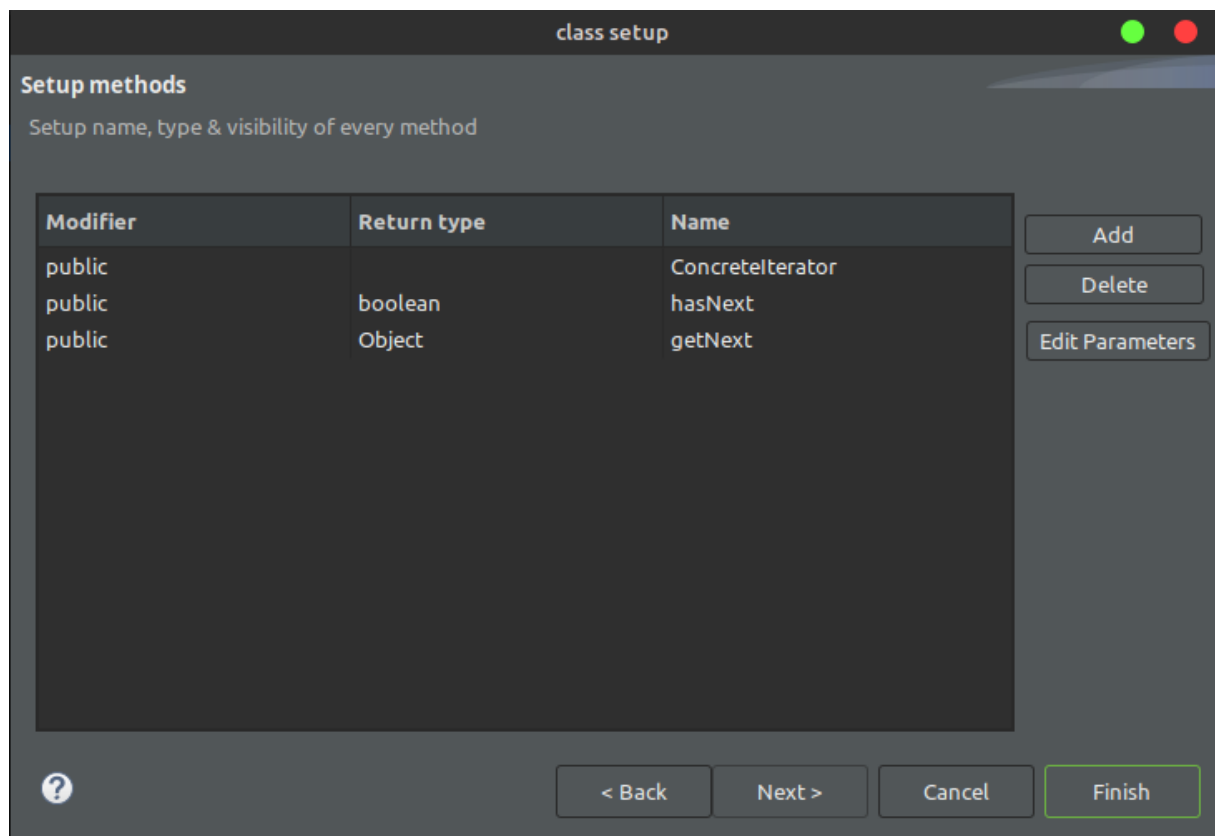
Σχήμα 6.6: Επεξεργασία πεδίων.



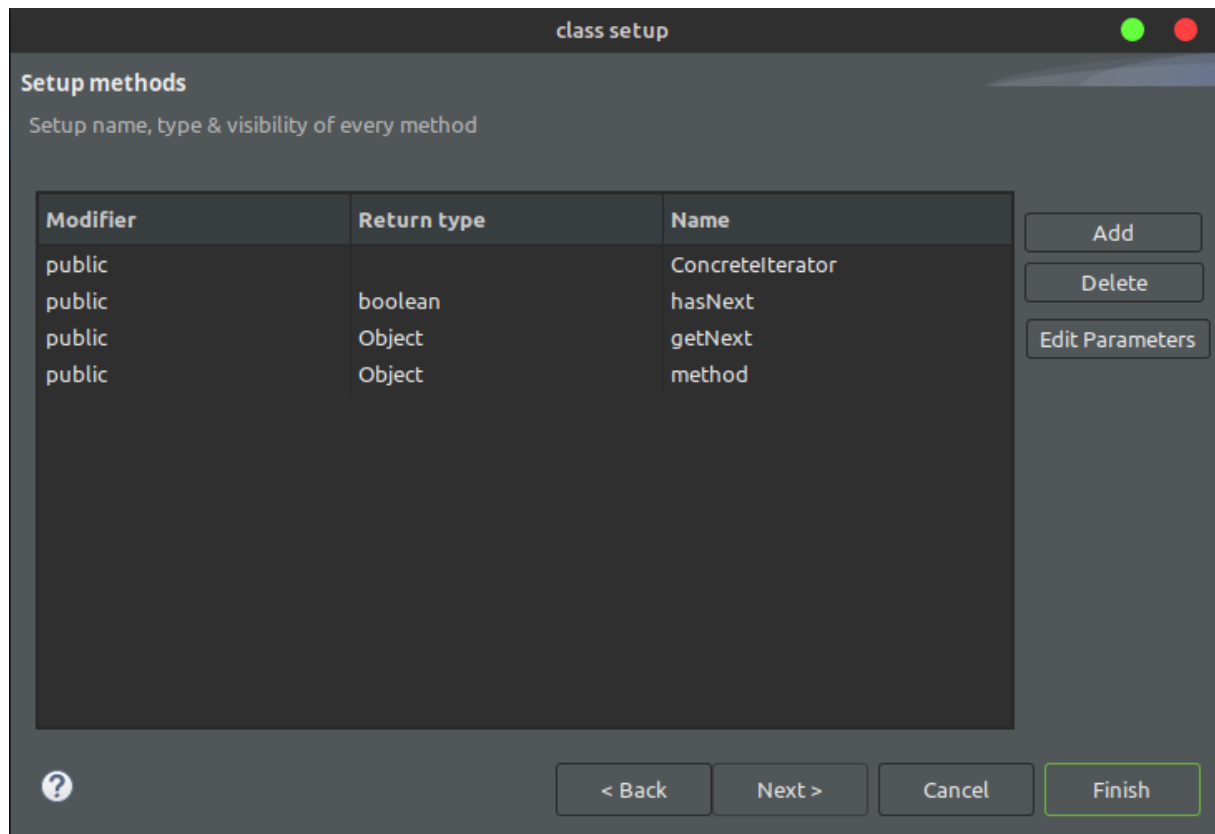


Σχήμα 6.7: Προσθήκη πεδίου.

Αφότου ο προγραμματιστής ολοκληρώσει την επεξεργασία των πεδίων, έχει την δυνατότητα να σταματήσει την επεξεργασία της κλάσης κάνοντας κλικ στο κουμπί finish ή να συνεχίσει με την επεξεργασία των μεθόδων της κλάσης. Το σύστημα θα εμφανίσει την σελίδα 6.8, όπου φαίνονται οι μέθοδοι της κλάσης που επεξεργάζεται ο προγραμματιστής. Δίνεται η δυνατότητα να τροποποιήσει κάποια μέθοδο, συμπληρώνοντας τα πεδία κειμένου. Επίσης, μπορεί να προσθαφαιρέσει κάποια μέθοδο, όπως φαίνεται στην εικόνα 6.9. Τέλος, μπορεί ο προγραμματιστής να επεξεργαστεί τις παραμέτρους κάποιας μεθόδου κάνοντας κλικ στο κουμπί Edit Parameters.

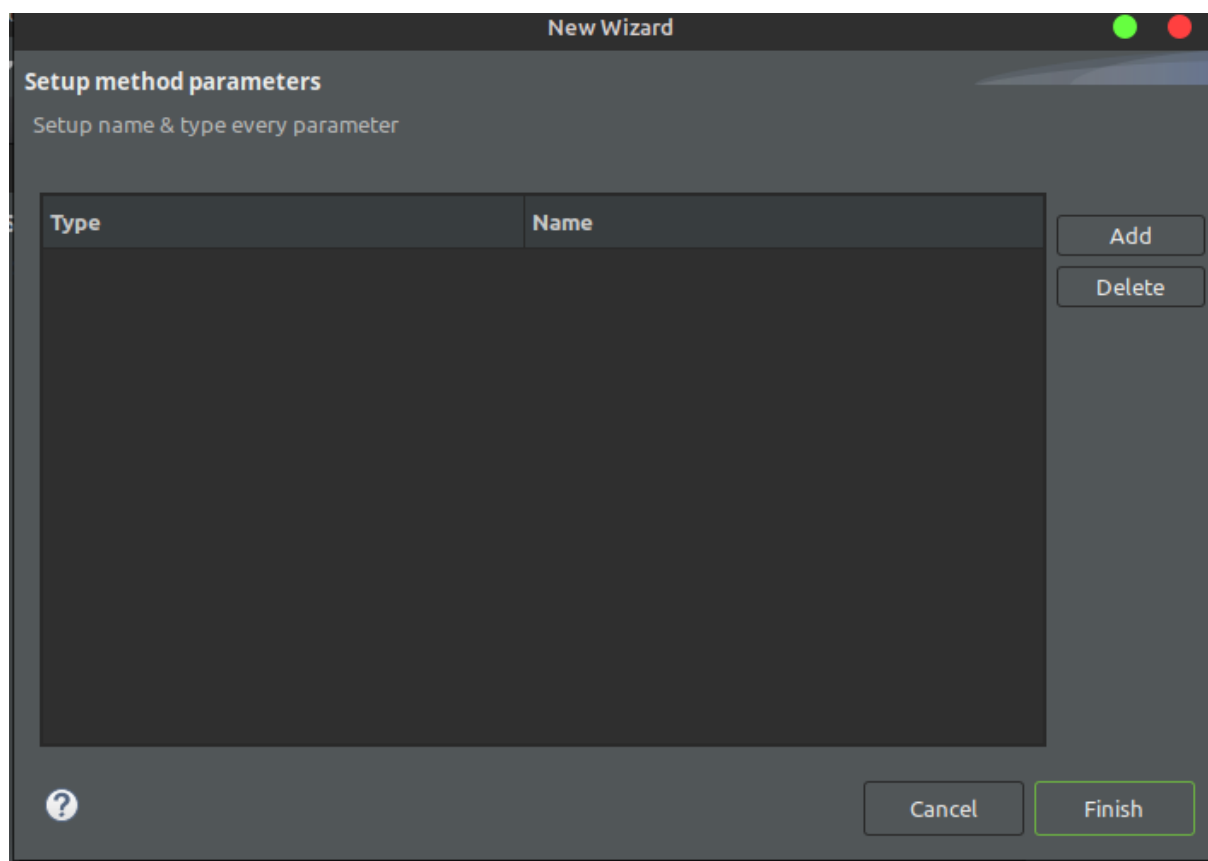


Σχήμα 6.8: Επεξεργασία μεθόδων κλάσης.

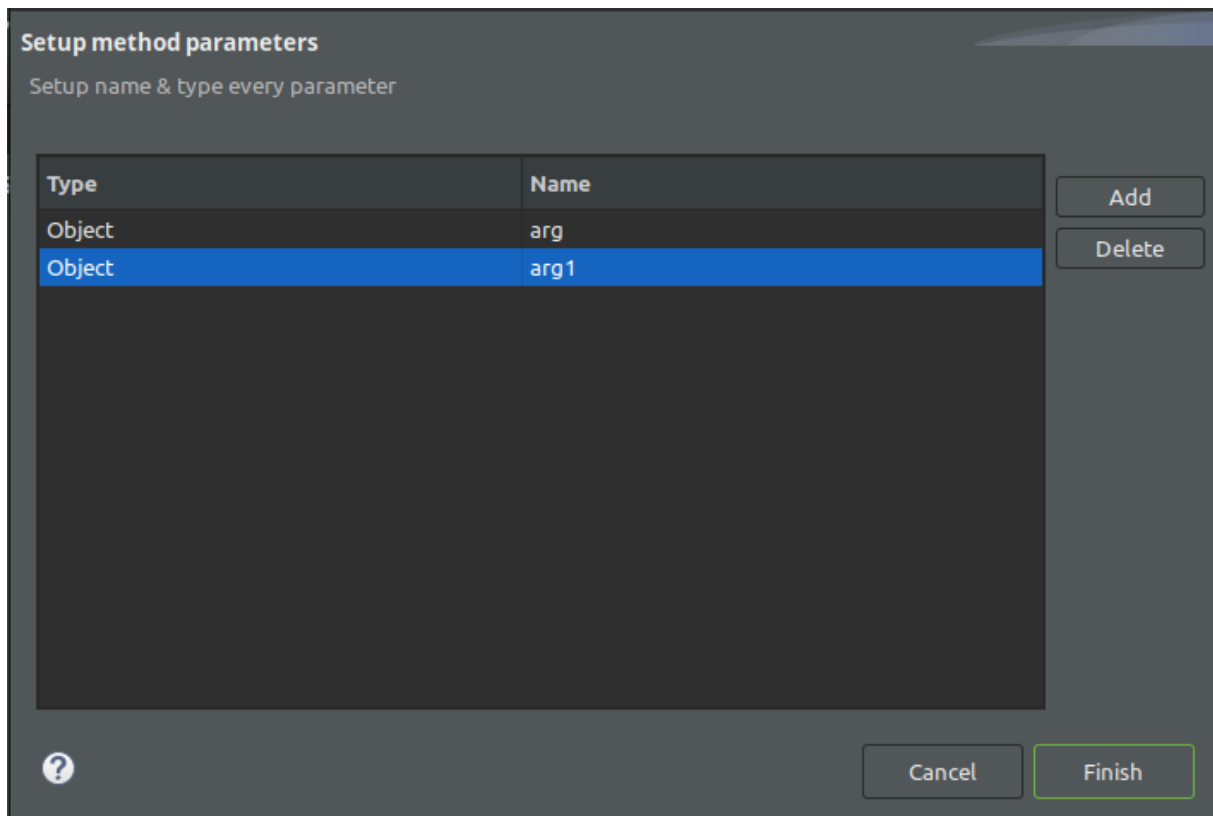


Σχήμα 6.9: Προσθήκη μεθόδου.

Μόλις ο προγραμματιστής πατήσει το κουμπί Edit Parameters, εμφανίζεται η σελίδα της εικόνας 6.10. Ο χρήστης μπορεί να προσθαφαιρέσει παραμέτρους, όπως στην εικόνα 6.11, καθώς και να τις επεξεργαστεί.

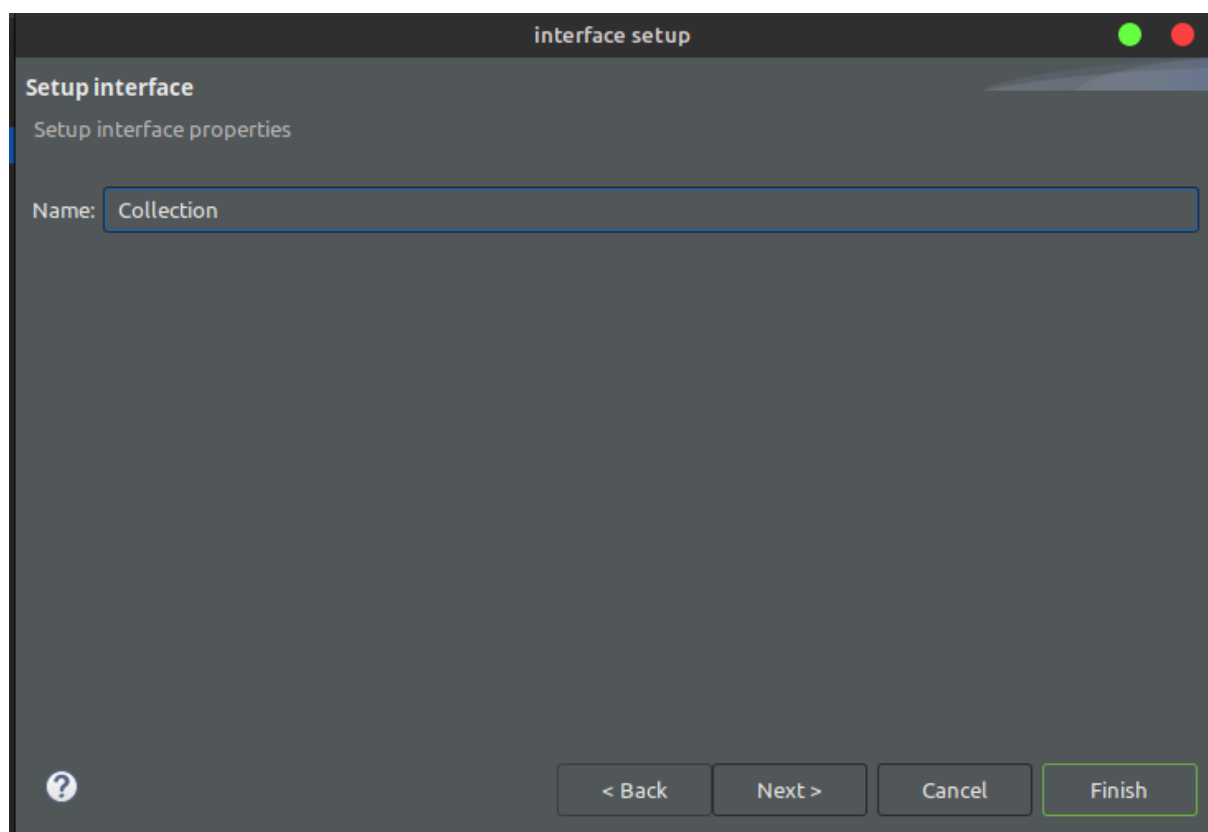


Σχήμα 6.10: Επεξεργασία παραμέτρων μεθόδου.

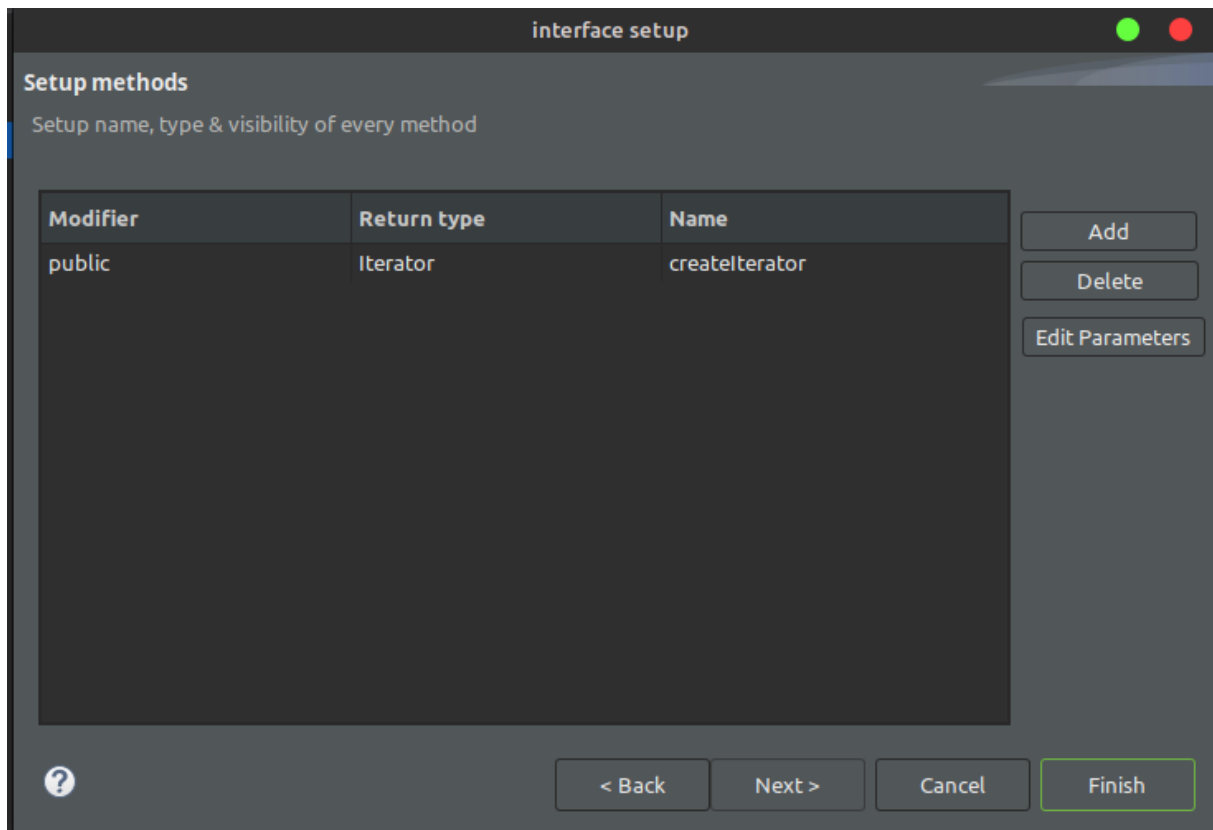


Σχήμα 6.11: Προσθήκη παραμέτρου.

Με αντίστοιχο τρόπο, ο προγραμματιστής έχει την δυνατότητα να τροποποιήσει κάποια διεπαφή του μοτίβου, επιλέγοντας την διεπαφή που επιθυμεί και πατώντας το κουμπί Edit interface. Εμφανίζεται ένας αντίστοιχος οδηγός, ο οποίος φαίνεται στην εικόνα 6.12. Εδώ, ο προγραμματιστής δύναται να τροποποιήσει το όνομα της διεπαφής και να επιλέξει να τερματίσει την υπόλοιπη διαδικασία για την επεξεργασία της διεπαφής πατώντας το Finish ή να συνεχίσει στην επόμενη σελίδα. Η επόμενη σελίδα απεικονίζεται στην εικόνα 6.13, στην οποία μπορεί να επεξεργαστεί τις μεθόδους της διεπαφής, όπως ακριβώς έκανε και με κάποια κλάση.

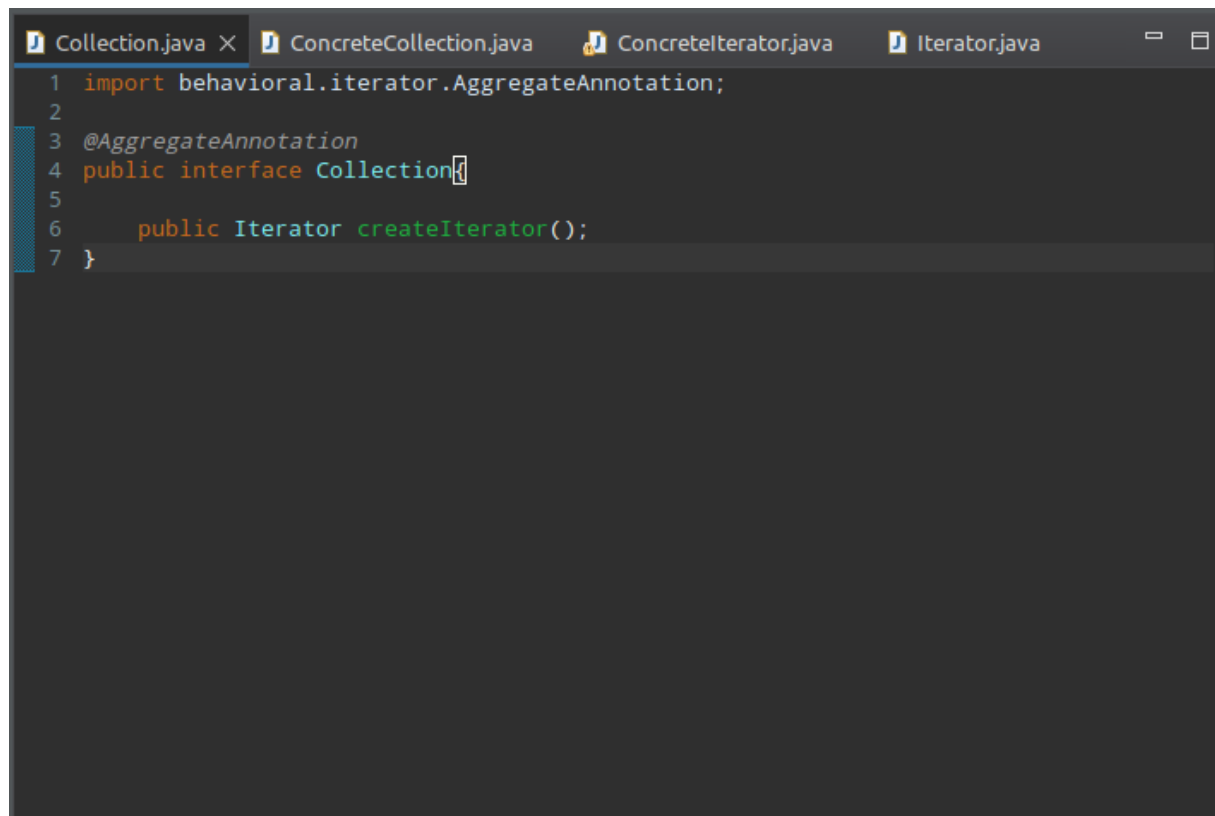


Σχήμα 6.12: Επεξεργασία ονόματος διεπαφής.



Σχήμα 6.13: Επεξεργασία μεθόδων διεπαφής.

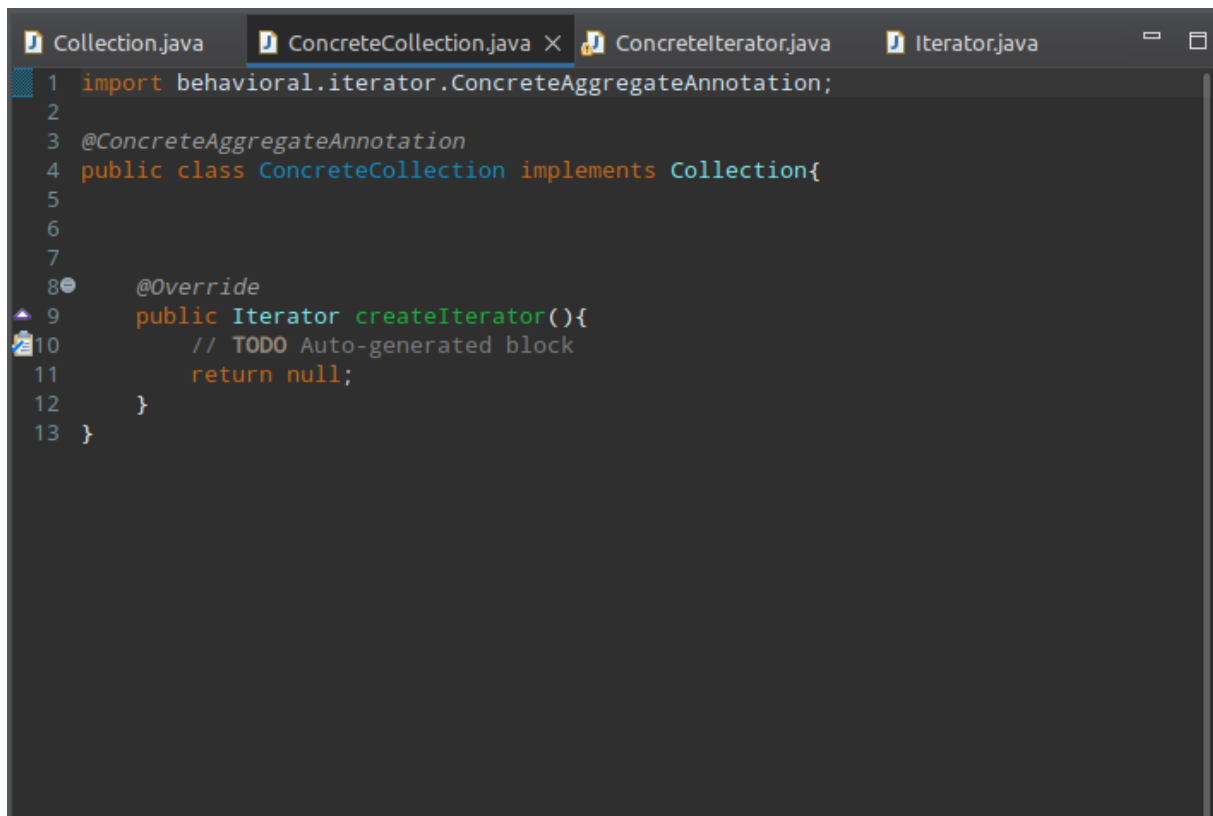
Τέλος, μόλις ο προγραμματιστής κάνει τις αλλαγές που επιθυμεί, χρειάζεται να πατήσει το κουμπί Finish στην οθόνη 6.4 και το εργαλείο θα παράξει τα πηγαία αρχεία (6.14, 6.15, 6.16, 6.17) που αντιστοιχούν στο μοτίβο. Ο προγραμματιστής έχει την δυνατότητα να ακυρώσει και να τερματίσει τον οδηγό, οποιαδήποτε στιγμή επιθυμεί, πατώντας τα κουμπιά Cancel.



```
1 import behavioral.iterator.AggregateAnnotation;
2
3 @AggregateAnnotation
4 public interface Collection {
5
6     public Iterator createIterator();
7 }
```

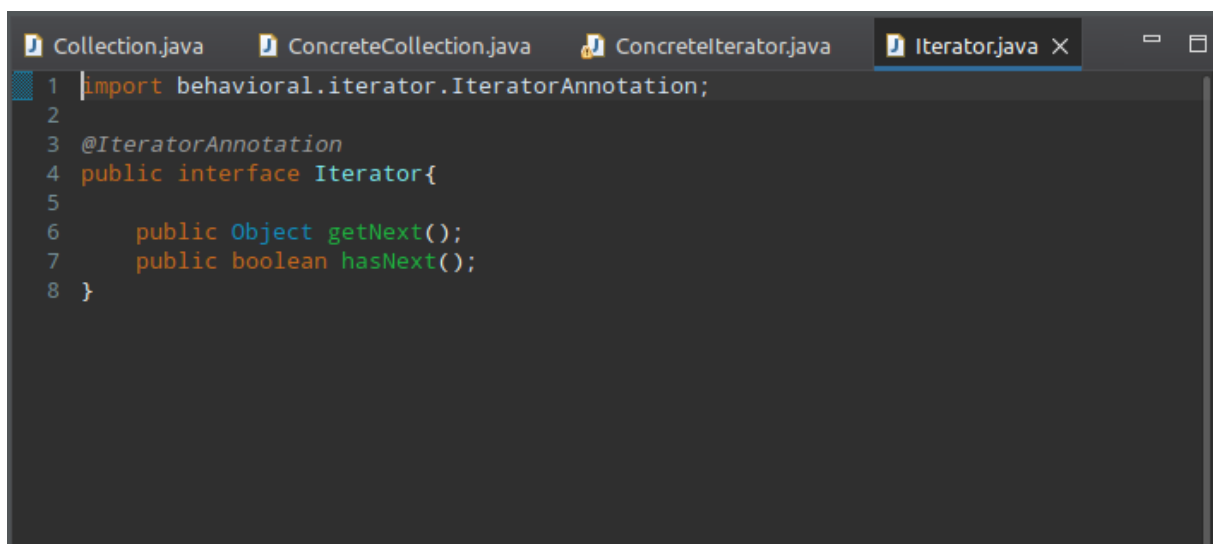
Σχήμα 6.14: Διεπαφή Collection.





```
1 import behavioral.iterator.ConcreteAggregateAnnotation;
2
3 @ConcreteAggregateAnnotation
4 public class ConcreteCollection implements Collection{
5
6
7
8     @Override
9     public Iterator createIterator(){
10         // TODO Auto-generated block
11         return null;
12     }
13 }
```

Σχήμα 6.15: Κλάση ConcreteCollection.



```
1 import behavioral.iterator.IteratorAnnotation;
2
3 @IteratorAnnotation
4 public interface Iterator{
5
6     public Object getNext();
7     public boolean hasNext();
8 }
```

Σχήμα 6.16: Διεπαφή Iterator.

Σχήμα 6.17: Κλάση ConcreteIterator.

# ΚΕΦΑΛΑΙΟ 7

## Επίλογος

### 7.1 Σύνοψη και συμπεράσματα

Στη διπλωματική αυτή δημιουργήθηκε ένα καινούριο εργαλείο με βασική λειτουργικότητα την ενσωμάτωση σχεδιαστικών μοτίβων σε πηγαίο κώδικα Java. Το εργαλείο βασίστηκε στα ήδη υπάρχοντα εργαλεία, Pattern Wizard, Patternbox, Design Pattern Automation Toolkit, καθώς και Alphaworks Design Pattern Toolkit [2], εξαλείφοντας και συνεισφέροντας σε βασικές τους ελλείψεις όπως η πληθώρα μοτίβων, την παραμετροποίηση των μοτίβων από τον χρήστη και τέλος, τον χαρακτηρισμό κάθε παραγόμενης κλάσης και διεπαφής, με την χρήση των annotations.

Επιπλέον, κάνοντας μία περιήγηση στο εργαλείο παρατηρούμε τα εξής: όλα τοποθετημένα στο ίδιο παράθυρο με τον κώδικα του χρήστη, καθώς πρόκειται για ένα εργαλείο το οποίο είναι επέκταση του περιβάλλοντος Eclipse. Χάρη στον σχεδιασμό του, μπορεί κάποιος πολύ εύκολα να επεκτείνει τα διαθέσιμα μοτίβα και με άλλα μοτίβα εκτός από αυτά των GoF [1].

Τέλος, η δυνατότητα των χρηστών να ελέγχουν ανά πάσα ώρα και στιγμή την πληροφορία που βρίσκεται μέσα στο εργαλείο. Καθώς, οι προσθήκες νέων μοτίβων από τον υπολογιστή του χρήστη όπως και η διαγραφή τους επιτρέπει, την εξατομικευμένη λειτουργία του. Έτσι, καλύπτονται πλήρως οι ανάγκες ενός προγραμματιστή ο οποίος έχει καλή γνώση των σχεδιαστικών μοτίβων.

## 7.2 Μελλοντικές επεκτάσεις

Στην ενότητα, αυτή θα προταθούν μερικές επεκτάσεις του εργαλείου. Στο μέλλον, θα μπορούσε να προστεθεί η δυνατότητα να μπορεί κάποιος προγραμματιστής να αντιστοιχίσει κλάσεις και διεπαφές του μοτίβου με υπάρχουσες, οι οποίες βρίσκονται στο τρέχον έργο του προγραμματιστή. Έτσι, μπορεί να προσαρμόσει τις ήδη υπάρχουσες κλάσεις και διεπαφές στις ανάγκες του μοτίβου. Με αυτό τον τρόπο, δίνεται η δυνατότητα στον χρήστη να εφαρμόσει διάφορα μοτίβα σε έργα τα οποία συντηρεί, αναδομώντας τα έργα αυτά και βελτιώνοντας τα.

# ΒΙΒΛΙΟΓΡΑΦΙΑ

---

- [1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.
- [2] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, “A design pattern generation tool,” Tech. Rep. GFP 0801, Faculty of Worcester Polytechnic Insitute, 2009.
- [3] J. R. Rumbaugh, M. R. Blaha, W. Lorensen, F. Eddy, and W. Premerlani, *Object-Oriented Modeling and Design*. Prentice Hall, October 1990.
- [4] I. Jacobson, *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.
- [5] P. Bourque and R. E. Fairley, eds., *SWEBOK: Guide to the Software Engineering Body of Knowledge*. Los Alamitos, CA: IEEE Computer Society, version 3.0 ed., 2014.

## ΠΑΡΑΡΤΗΜΑ Α

### Κάρτες αρμοδιοτήτων και συνεργασιών κλάσεων

Όνομα κλάσης: PatternClass	
Αρμοδιότητες	Συνεργασίες
<ul style="list-style-type: none"><li>Αυτή η κλάση είναι υπεύθυνη για την μοντελοποίηση μίας κλάσης ενός μοτίβου.</li></ul>	<ul style="list-style-type: none"><li>Method</li><li>Field</li></ul>

Πίνακας Α.1: PatternClass κάρτα αρμοδιοτήτων.

Όνομα κλάσης: PatternInterface	
Αρμοδιότητες	Συνεργασίες
<ul style="list-style-type: none"> <li>Αυτή η κλάση είναι υπεύθυνη για την μοντελοποίηση μίας διεπαφής ενός μοτίβου.</li> </ul>	<ul style="list-style-type: none"> <li>Method</li> </ul>

Πίνακας Α.2: PatternInterface κάρτα αρμοδιοτήτων.

Όνομα κλάσης: Method	
Αρμοδιότητες	Συνεργασίες
<ul style="list-style-type: none"> <li>Η κλάση αυτή διατηρεί τα χαρακτηριστικά μίας μεθόδου που ανήκει σε κάποια κλάση ή διεπαφή.</li> </ul>	

Πίνακας Α.3: Method κάρτα αρμοδιοτήτων.

Όνομα κλάσης: Field	
Αρμοδιότητες	Συνεργασίες
<ul style="list-style-type: none"> <li>• Η κλάση αυτή διατηρεί τα χαρακτηριστικά ενός πεδίου που ανήκει σε κάποια κλάση.</li> </ul>	

Πίνακας Α.4: Field κάρτα αρμοδιοτήτων.



Όνομα κλάσης: FileParser	
Αρμοδιότητες	Συνεργασίες
<ul style="list-style-type: none"> <li>• Η κλάση αυτή είναι υπεύθυνη για την άντληση των κατηγοριών που ορίζουν οι GoF [1].</li> <li>• Η κλάση αυτή αρμόδια για την άντληση των μοτίβων κάποιας κατηγορίας.</li> <li>• Η κλάση αυτή είναι αρμόδια για την άντληση των κλάσεων ενός μοτίβου, όπως και την ορατότητα της κλάσης.</li> <li>• Η κλάση αυτή είναι αρμόδια για την άντληση των διεπαφών ενός μοτίβου όπως και την ορατότητα της διεπαφής.</li> <li>• Η κλάση αυτή είναι αρμόδια για την άντληση των μεθόδων κάποιας κλάσης, καθώς τους διάφορους τροποποιητές που μπορεί να έχει μία μέθοδος, όπως και τον επιστρεφόμενο τύπο της μεθόδου και τέλος τον κώδικα της μεθόδου.</li> <li>• Η κλάση αυτή είναι αρμόδια για την άντληση των πεδίων κάποιας κλάσης ενός μοτίβου.</li> <li>• Η κλάση αυτή είναι αρμόδια για την άντληση των μεθόδων κάποιας διεπαφής μαζί με τους τροποποιητές της μεθόδου και τον επιστρεφόμενο τύπο της.</li> <li>• Η κλάση αυτή είναι αρμόδια για την άντληση των annotations ενός στοιχείου του μοτίβου.</li> </ul>	

Πίνακας A.5: FileParser κάρτα αρμοδιοτήτων.

Όνομα κλάσης: PatternManager	
Αρμοδιότητες	Συνεργασίες
<ul style="list-style-type: none"> <li>• Η κλάση αυτή είναι αρμόδια για την διάθεση των κατηγοριών των μοτίβων στην γραφική διεπαφή.</li> <li>• Η κλάση αυτή είναι αρμόδια για την διάθεση των μοτίβων κάποιας κατηγορίας στην γραφική διεπαφή.</li> <li>• Η κλάση αυτή είναι αρμόδια για την δημιουργία αντικειμένων τύπου PatternClass και την διαθέση τους στο front-end τμήμα του εργαλείου, ώστε να μπορεί να διαχειριστεί ο χρήστης τις διάφορες κλάσεις ενός μοτίβου.</li> <li>• Η κλάση αυτή είναι αρμόδια για την δημιουργία αντικειμένων τύπου PatternInterface και την διαθέση τους στο front-end τμήμα του εργαλείου, ώστε να μπορεί να διαχειριστεί ο χρήστης τις διάφορες διεπαφές ενός μοτίβου.</li> <li>• Η κλάση αυτή είναι αρμόδια για την ενημέρωση του ονόματος μίας κλάσης.</li> <li>• Η κλάση αυτή είναι αρμόδια για την ενημέρωση του ονόματος μίας διεπαφής.</li> <li>• Η κλάση αυτή είναι αρμόδια για την ενημέρωση του ονόματος μίας μεθόδου.</li> <li>• Η κλάση αυτή είναι αρμόδια για την ενημέρωση του ονόματος ενός πεδίου.</li> </ul>	<ul style="list-style-type: none"> <li>• FileParser</li> <li>• PatternClass</li> <li>• PatternInterface</li> <li>• Method</li> <li>• Field</li> <li>• Property</li> </ul>

Πίνακας Α.6: PatternManager κάρτα αρμοδιοτήτων.

Όνομα κλάσης: ClassGenerator	
Αρμοδιότητες	Συνεργασίες
<ul style="list-style-type: none"> <li>• Η κλάση αυτή είναι αρμόδια για την προσθήκη των annotations στο classpath του έργου που έχει επιλέξει ο προγραμματιστής.</li> <li>• Η κλάση αυτή είναι αρμόδια για την παραγωγή των πηγαιών αρχείων java μίας κλάσης.</li> </ul>	<ul style="list-style-type: none"> <li>• PatternManager</li> <li>• PatternClass</li> <li>• PatternInterface</li> <li>• Method</li> <li>• Field</li> <li>• Property</li> </ul>

Πίνακας A.7: ClassGenerator κάρτα αρμοδιοτήτων

Όνομα κλάσης: InterfaceGenerator	
Αρμοδιότητες	Συνεργασίες
<ul style="list-style-type: none"> <li>• Η κλάση αυτή είναι αρμόδια για την προσθήκη των annotations στο classpath του έργου που έχει επιλέξει ο προγραμματιστής.</li> <li>• Η κλάση αυτή είναι αρμόδια για την παραγωγή των πηγαιών αρχείων java μίας διεπαφής.</li> </ul>	<ul style="list-style-type: none"> <li>• PatternManager</li> <li>• PatternClass</li> <li>• PatternInterface</li> <li>• Method</li> <li>• Field</li> <li>• Property</li> </ul>

Πίνακας A.8: InterfaceGenerator κάρτα αρμοδιοτήτων.