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

ΕΥΧΑΡΙΣΤΙΕΣ

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

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 κλειδία: Επέκταση Eclipse, διπλωματική εργασία, σχεδιαστικά μοτίβα, προγραμματιστής

ABSTRACT

Anastasios Liontos, 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 μια κοινή εργασία.
Όπως οι αλγόριθμοι, έτσι και τα πρότυπα σχεδίασης είναι μια δοκιμασμένη και αποδεκτή λύση- 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

Αυτή η διπλωματική επιδιώκει την ανάπτυξη ενός εργαλείου που επιτρέπει που επιτρέπει σε αρχάριους προγραμματιστές να εισάγουν σχεδιαστικά μοτίβα στον κώδικά τους σε Java με ένα απλό βήμα. Το εργαλείο παρέχει μια διεπαφή, όπου ο χρήστης μπορεί 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 Ob-

server 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

Η διπλωματική εργασία περιέχει 7 κεφάλαια. 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

2.1.1 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 κώδικά 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.

2.1.2 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 οου 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.

2.1.3 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 υλοποιεί το εργαλείο. Τέλος δεν τροποποιεί τον υπάρχων κώδικά και παρά μόνο εξάγει τον σκελετό του μοτίβου.

2.1.4 Alphaworks Design Pattern Toolkit

Αυτό το εργαλείο [2] είναι αντίστοιχο του Patternbox??, με την διαφορά ότι το Alphaworks μετασχηματίζει τον κώδικα του προγραμματιστή 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.

2.1.5 Σύγκριση

Το Design Pattern Builder, το οποίο είναι το εργαλείο που αποτελεί αντικείμενο της διπλωματικής αυτής, παρουσιάζει πλεονεκτήματα έναντι των παραπάνω εργαλείων, καθώς ο προγραμματιστής μπορεί να τροποποιήσει οποιοδήποτε μοτίβο, ώστε το μοτίβο να ταιριάξει με τις ανάγκες του έργου του προγραμματιστή. Επίσης εξάγει annotations -τα οποία αποδίδουν τον ρόλο που έχει η κλάση ή 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 του μοτίβου με κάποιο στοιχείο του έργου του προγραμματιστή. Τέλος ένα ακόμα μειονέκτημα είναι η δυνατότητα εξαγωγής κώδικα και σε άλλες γλώσσες προγραμματισμού εκτός της java.

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 εμπει-

ρία και την εμπειρογνωμοσύνη των τεσσάρων συγγραφέων, οι GoF, 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 και παραδείγματα καθώς αναδεικνύουν την εφαρμογή της σχεδίασης αυτής. Για την περιγραφή των μοτίβων χρησιμοποιείτε 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 που αφορούν το εργαλείο, σε μορφή πινάκων, καθώς και οι περιπτώσεις χρήσης του.

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].

Πίνακας 3.1: Ιστορίες χρήστη

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 προσαρμοσω 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 προσαρμοσω 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 προσαρμοσω 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 προσαρμοσω 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 προσαρμοσω 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 κώδικα μου.
IX10	Προγραμματιστής	Να μπορώ να προσθέσω νέες κλάσεις.	Έτσι ώστε να προσαρμοσω το μοτίβο στον κώδικά μου.
IX11	Προγραμματιστής	Να μπορώ να καθορίσω ποίες διεπαφές θα υλοποιούν οι νέες κλάσεις.	Έτσι ώστε να προσαρμοσω τις κλάσεις αυτές στον κώδικά μου και στις ανάγκες του μοτίβου.

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, δηλαδή κάποιον που παίζει έναν ρόλο και αλληλεπιδρά με το σύστημα με τον τρόπο που ορίζει το περιεχόμενο της

περίπτωσης χρήσης.

Περίπτωση χρήσης: Επιλογή κατηγορίας μοτίβου.
Αναγνωριστικό: ΠΧ1
Προϋποθέσεις: 1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει το έργο που θα εργαστεί.
Ροή γεγονότων: 1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής επιλέξει Import pattern, κάτω από το μενού Design pattern builder στο παράθυρο New του eclipse. 2. Το σύστημα εμφανίζει έναν οδηγό. 3. Ο προγραμματιστής επιλέγει την κατηγορία μοτίβου που επιθυμεί.
Μετα-συνθήκες: Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.2: Επιλογή κατηγορίας μοτίβου.

Περίπτωση χρήσης: Επιλογή μοτίβου
Αναγνωριστικό: ΠΧ2
Προϋποθέσεις: 1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει κατηγορία μοτίβου.
Ροή γεγονότων: 1. Η περίπτωση χρήσης ξεκινάει όταν ο προγραμματιστής κάνει κλικ στην πτυσσόμενη λίστα. 2. Το σύστημα εμφανίζει τα διαθέσιμα μοτίβα. 3. Ο προγραμματιστής επιλέγει το μοτίβο που επιθυμεί.
Μετα-συνθήκες: Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.3: Επιλογή μοτίβου.

Περίπτωση χρήσης: Καθορισμός μεθόδων κλάσης
Αναγνωριστικό: ΠΧ3
Προϋποθέσεις: 1. Ο προγραμματιστής χρειάζεται να είναι στο παράθυρο επεξεργασίας της κλάσης.
Ροή γεγονότων: 1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επεξεργασίας πεδίων της κλάσης. 2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις μεθόδους της τρέχουσας κλάσης. 3. Για κάθε μέθοδο: (α) Ο προγραμματιστής επεξεργάζεται το όνομα της μεθόδου. (β) Ο προγραμματιστής επεξεργάζεται τον επιστρεφόμενο τύπο της μεθόδου. (γ) Ο προγραμματιστής επεξεργάζεται την ορατότητα της μεθόδου. 4. Ο προγραμματιστής κάνει κλικ στο κουμπί finish. 5. Το σύστημα κλείνει το παράθυρο.
Μετα-συνθήκες: 1. Το σύστημα ρυθμίζει το όνομα της κλάσης. 2. Το σύστημα ρυθμίζει τα πεδία της κλάσης. 3. Το σύστημα ρυθμίζει τις μεθόδους της κλάσης.

Πίνακας 3.4: Καθορισμός μεθόδων κλάσης.

Περίπτωση χρήσης: Ονοματοδοσία κλάσης.
Αναγνωριστικό: ΠΧ4
Προϋποθέσεις: 1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει μοτίβο.
Ροή γεγονότων: 1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επιλογής μοτίβου. 2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις κλάσεις και τις διεπαφές του μοτίβου. 3. Ο προγραμματιστής επιλέγει την κλάση που επιθυμεί. 4. Ο προγραμματιστής κάνει κλικ στο κουμπί edit class. 5. Το σύστημα εμφανίζει ένα νέο παράθυρο όπου ο χρήστης μπορεί να επεξεργαστεί το όνομα της κλάσης. 6. Ο προγραμματιστής επεξεργάζεται το όνομα της κλάσης. 7. Ο προγραμματιστής κάνει κλικ στο κουμπί Next. 8. Το σύστημα εμφανίζει ένα νέο παράθυρο.
Μετα-συνθήκες: Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.5: Ονοματοδοσία κλάσης.

Περίπτωση χρήσης: Καθορισμός πεδίων κλάσης
Αναγνωριστικό: ΠΧ5
Προϋποθέσεις: 1. Ο προγραμματιστής χρειάζεται να έχει επεξεργαστεί το όνομα της κλάσης
Ροή γεγονότων: 1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επεξεργασίας ονόματος της κλάσης. 2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τα πεδία της τρέχουσας κλάσης. 3. Για κάθε μέθοδο: (α) Ο προγραμματιστής επεξεργάζεται το όνομα του πεδίου. (β) Ο προγραμματιστής επεξεργάζεται τον τύπο του πεδίου. (γ) Ο προγραμματιστής επεξεργάζεται την ορατότητα του πεδίου. 4. Ο προγραμματιστής κάνει κλικ στο κουμπί Next. 5. Το σύστημα εμφανίζει ένα νέο παράθυρο.
Μετα-συνθήκες:

Πίνακας 3.6: Καθορισμός πεδίων κλάσης.

Περίπτωση χρήσης: Ονοματοδοσία διεπαφής
Αναγνωριστικό: ΠΧ6
Προϋποθέσεις: 1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει μοτίβο.
Ροή γεγονότων: 1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επιλογής μοτίβου. 2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις κλάσεις και τις διεπαφές του μοτίβου. 3. Ο προγραμματιστής επιλέγει την διεπαφή που επιθυμεί. 4. Ο προγραμματιστής κάνει κλικ στο κουμπί edit interface. 5. Το σύστημα εμφανίζει ένα νέο παράθυρο όπου ο χρήστης μπορεί να επεξεργαστεί το όνομα της διεπαφής. 6. Ο προγραμματιστής επεξεργάζεται το όνομα της διεπαφής. 7. Ο προγραμματιστής κάνει κλικ στο κουμπί Next. 8. Το σύστημα εμφανίζει ένα νέο παράθυρο.
Μετα-συνθήκες:

Πίνακας 3.7: Ονοματοδοσία διεπαφής.

Περίπτωση χρήσης: Καθορισμός μεθόδων διεπαφής
Αναγνωριστικό: ΠΧ7
Προϋποθέσεις: 1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επεξεργασίας ονόματος της διεπαφής. 2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις μεθόδους της τρέχουσας διεπαφής. 3. Για κάθε μέθοδο: (α) Ο προγραμματιστής επεξεργάζεται το όνομα της μεθόδου. (β) Ο προγραμματιστής επεξεργάζεται τον επιστρεφόμενο τύπο της μεθόδου. (γ) Ο προγραμματιστής επεξεργάζεται την ορατότητα της μεθόδου. 4. Ο προγραμματιστής κάνει κλικ στο κουμπί finish. 5. Το σύστημα κλείνει το παράθυρο.
Μετα-συνθήκες: 1. Το σύστημα ρυθμίζει το όνομα της διεπαφής. 2. Το σύστημα ρυθμίζει τις μεθόδους της διεπαφής.

Πίνακας 3.8: Καθορισμός μεθόδων διεπαφής.

Περίπτωση χρήσης: Εξαγωγή σκελετού επιλεγμένου μοτίβου.
Αναγνωριστικό: ΠΧ8
Προϋποθέσεις: 1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει κατηγορία μοτίβου.
Ροή γεγονότων: 1. Η Περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί finish στο παράθυρο επιλογής κλάσης ή διεπαφής. 2. Το σύστημα δημιουργεί τα απαραίτητα πηγαία αρχεία java στο επιλεγμένο πακέτο. 3. Το σύστημα τερματίζει.
Μετα-συνθήκες: Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.9: Εξαγωγή σκελετού επιλεγμένου μοτίβου.

Περίπτωση χρήσης: Προσθήκη νέας κλάσης
Αναγνωριστικό: ΠΧ10
Προϋποθέσεις: 1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει μοτίβο. 2. Το μοτίβο πρέπει να επιτρέπει την εισαγωγή καινούργιας κλάσης.
Ροή γεγονότων: 1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επιλογής μοτίβου. 2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις κλάσεις και τις διεπαφές του μοτίβου. 3. Ο προγραμματιστής Κάνει κλικ στο κουμπί Add Class. 4. Το σύστημα προσθέτει μία νέα κλάση.
Μετα-συνθήκες: Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.10: Προσθήκη νέας κλάσης.

Περίπτωση χρήσης: Καθορισμός υλοποιημένης διεπαφής
Αναγνωριστικό: ΠΧ11
Προϋποθέσεις: 1. Ο προγραμματιστής χρειάζεται να έχει επιλέξει μοτίβο.
Ροή γεγονότων: 1. Η περίπτωση χρήσης ξεκινά όταν ο προγραμματιστής κάνει κλικ στο κουμπί Next του παραθύρου επιλογής μοτίβου. 2. Το σύστημα εμφανίζει ένα νέο παράθυρο με τις κλάσεις και τις διεπαφές του μοτίβου. 3. Ο προγραμματιστής επιλέγει μία κλάση που έχει προσθέσει ο ίδιος. 4. Ο προγραμματιστής κάνει κλικ στο κουμπί edit class. 5. Το σύστημα εμφανίζει ένα νέο παράθυρο όπου ο χρήστης μπορεί να επεξεργαστεί την διεπαφή που μπορεί να υλοποιεί η κλάση αυτή. 6. Ο προγραμματιστής επεξεργάζεται την διεπαφή που μπορεί να υλοποιεί η κλάση αυτή. 7. Ο προγραμματιστής κάνει κλικ στο κουμπί Next. 8. Το σύστημα εμφανίζει ένα νέο παράθυρο.
Μετα-συνθήκες: Η κατάσταση του κώδικα που υλοποιεί ο προγραμματιστής παραμένει ως έχει.

Πίνακας 3.11: Καθορισμός υλοποιημένης διεπαφής.

ΚΕΦΑΛΑΙΟ 4

Σχεδίαση και αρχιτεκτονική λογισμικού

4.1 Πακέτα συστήματος

4.2 Κλάσεις συστήματος

4.2.1 Ανάλυση κλάσεων

4.2.2 Διάγραμμα κλάσεων

4.3 Κάρτες αρμοδιοτήτων και συνεργασιών κλάσεων

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

5.2 Έλεγχος μεθόδων

Σε αυτήν την ενότητα, θα παρουσιαστούν οι έλεγχοι για την δομή των μοτίβων τα οποία περιγράφονται σε ένα αρχείο xml. 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, 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. 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 μοτίβου, 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, 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 μοτίβου, 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.

5.3 ΄Ελεγχος μεθόδων δημιουργίας πηγαίου κώδικα java

ΚΕΦΑΛΑΙΟ 6

Οδηγός Χρήσης Design Pattern Builder

6.1 Λειτουργίες χρήστη

ΚΕΦΑΛΑΙΟ 7

Επίλογος

7.1 Μελλοντικές επεκτάσεις

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.