Τ. Ε. Ι. ΗΠΕΙΡΟΥ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ΕΠΙΤΡΑΠΕΖΙΟΥ ΠΑΙΧΝΙΔΙΟΥ ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΤΟ ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΤΗΣ JAVA

ΡΟΥΣΣΟΥ ΜΑΡΙΑ

ΕΠΙΒΛΕΠΩΝ
ΛΙΑΡΟΚΑΠΗΣ ΔΗΜΗΤΡΙΟΣ
ΚΑΘΗΓΗΤΗΣ ΕΦΑΡΜΟΓΩΝ

ΑΡΤΑ 2015

ΠΝΕΥΜΑΤΙΚΑ ΔΙΚΑΙΩΜΑΤΑ

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα της πτυχιακής μου εργασίας με θέμα: **Δημιουργία ενός επιτραπέζιου παιχνιδιού χρησιμοποιώντας το γραφικό περιβάλλον της JAVA**, καθηγητή του Τ.Ε.Ι. Ηπείρου : **Λιαροκάπη Δημήτριο**, για την αμέριστη βοήθεια, τις ουσιώδεις συμβουλές, και τη συνεχή καθοδήγησή του καθ' όλη τη διάρκεια της πτυχιακής μου εργασίας.

Θα ήθελα να ευχαριστήσω ακόμα, όλους του καθηγητές του τμήματος Μηχανικών Πληροφορικής του Τ.Ε.Ι. Ηπείρου και του τμήματος Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών του Τ.Ε.Ι. Ιονίων Νήσων για τις πολύτιμες γνώσεις που μου προσέφεραν όλα αυτά τα χρόνια.

Τέλος, θέλω να εκφράσω ένα τεράστιο ευχαριστώ στην οικογένεια μου που με στήριξε ηθικά και οικονομικά όλα αυτά τα χρόνια φροντίζοντας για την καλύτερη δυνατή μόρφωση μου.

Ρούσσου Μαρία
Ιούνιος 2015

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία εμβαθύνω στην χρήση του γραφικού περιβάλλοντος που παρέχεται από την γλώσσα προγραμματισμού Java, και υλοποιώ ένα παιχνίδι όπως το τάβλι.

Ο στόχος της τρέχουσας πτυχιακής εργασίας είναι η αναπαράσταση των αντικειμένων και κινήσεων του παιχνιδιού και ο τελικός έλεγχος ορθότητάς τους.

Η υλοποίηση του παιχνιδιού διήρκησε 9 μήνες και ο σκοπός του είναι να ψυχαγωγήσει τους χρήστες.

Για την ανάπτυξη της εφαρμογής ακολουθήθηκαν οι παρακάτω βασικές φάσεις:

Αρχικά δημιουργήθηκε το διάγραμμα του παιχνιδιού το οποίο αποτελείται από 12 στήλες πάνω (σε σχέση με την πλατφόρμα) και 12 στήλες κάτω.

Έπειτα προστέθηκαν οι βασικοί κανόνες του παιχνιδιού και ο κώδικας αρχικοποίησης που καλείται με το πάτημα ενός κουμπιού «Νέο Παιχνίδι».

Στη συνέχεια προστέθηκε κουμπί «Ζάρια», το οποίο καλεί τον κώδικα για την δημιουργία 2 τυχαίων αριθμών από το 1 έως το 6 για την υπόδειξη της επόμενης κίνησης.

Το κουμπί «Παραίτηση» προστέθηκε για την διακοπή ενός παιχνιδιού και την έναρξη καινούριου προσθέτοντας μια νίκη στον αντίπαλο παίκτη.

Τέλος υλοποιήθηκε ο κώδικας υποστήριξης του μαζέματος των πουλιών.

ABSTRACT

At the current thesis I use the graphical interface provided by Java language, and I implement a game like backgammon.

The aim of this thesis is the representation of objects and movements of backgammon and the final check of their accuracy.

The programming of this game lasted 9 months and its objective is to entertain users.

The following main stages were followed for the development of the application:

At first I created the platform which consists of 12 columns above and 12 columns below.

Then I set up the basic rules of the game and the code which initializes the checkers on the platform and starts a new game when a button “New Game” is pressed.

Afterwards I created the button “Dice”, which calls the code for the creation of 2 random numbers between 1 and 6 to indicate the next move.

The button “Resignation” was added to stop a game and create a new one, adding 1 win (1 point) to the opposing player.

Finally, the code for the bear off was created.

ΠΕΡΙΕΧΟΜΕΝΑ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

FIBS	First Internet Backgammon Server
JFC	Java Foundation Classes
GUI	Graphical User Interface
AWT	Abstract Windowing Toolkit
WORA	Write Once Run Anywhere
JDK	Java Development Kit
HTML	HyperText Markup Language
MVC	Model View Controller
IDE	Integrated Development Environment

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1ο: ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ – ΣΥΣΤΑΤΙΚΑ – ΚΑΝΟΝΕΣ

1.1 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

1.1.1 ΤΟ ΤΑΒΛΙ ΣΤΗΝ ΑΡΧΑΙΟΤΗΤΑ [1]

Το τάβλι είναι ίσως το αρχαιότερο παιχνίδι. Πρώτη φορά εμφανίζεται στη Μεσοποταμία (2900 με 1800 πχ) σύμφωνα με την ανακάλυψη ενός Άγγλου αρχαιολόγου.

Επίσης εντοπίζεται και στην Αρχαία Αίγυπτο με το όνομα Senet ή «Παιχνίδι των τριάντα τετραγώνων». Αντί για ζάρια, χρησιμοποιούσαν τέσσερις βέργες με δύο πλευρές και προχωρούσαν ένα πούλι ανάλογα με το συνδυασμό των πλευρών των βεργών.

Στην Περσία περίπου το 1600 πχ, το παιχνίδι Τακτέ Νάρντ (Μάχη σε ξύλο), έμοιαζε πολύ περισσότερο με το τάβλι που παίζουμε σήμερα και μεταφέρθηκε στην Ευρώπη από τους Άραβες (Εικόνα 1).

Πολύ αργότερα εμφανίστηκε και στην Ελλάδα όπου και το 50 μΧ απλοποιήθηκε.

Εικόνα 1 – Το Τάβλι στην αρχαιότητα

[1] Ιστορία του Ταβλιού, el.wikipedia.org/wiki

1.1.2 ΤΟ ΤΑΒΛΙ ΣΤΗ ΣΥΓΧΡΟΝΗ ΕΠΟΧΗ [2]

Το τάβλι αρχίζει να γίνεται γνωστό στην Ευρώπη στην εποχή των μεσαιωνικών χρόνων και αποκτά διαφορετικά ονόματα ανάλογα με την περιοχή. Επειδή όμως ήταν τυχερό παιχνίδι συχνά κατακρίθηκε και απαγορεύθηκε από την Εκκλησία.

Από το 18ο αιώνα και μετά η Εκκλησία είχε άρει κάθε απαγόρευση για το τάβλι το οποίο ήταν πλέον πλήρως αποδεκτό. Μάλιστα το 1743 εκδόθηκε από τον Edmund Hoyle, ο οποίος ήταν ειδικός στα επιτραπέζια παιχνίδια, η «Πραγματεία πάνω στο παιχνίδι του ταβλιού». Μέσω του Hoyle τυποποιήθηκαν οι κανόνες του παιχνιδιού, οι οποίοι ισχύουν μέχρι και σήμερα.

Την ίδια περίοδο απέκτησε το διεθνές του όνομα backgammon.

[2] Ιστορία του Ταβλιού, livetavli.gr/history.htm

1.2 ΤΟ ΨΗΦΙΑΚΟ ΠΑΙΧΝΙΔΙ [3]

Παρά το γεγονός ότι η τύχη είναι ένας από τους καθοριστικούς παράγοντες για το αποτέλεσμα, η στρατηγική παίζει πιο σημαντικό ρόλο σε μακροπρόθεσμη βάση. Με κάθε ζαριά, οι παίκτες πρέπει να επιλέξουν ανάμεσα στις πολλές επιλογές για την μετακίνηση των πουλιών τους και να αποφύγουν τυχόν κινήσεις του αντιπάλου που θα τους φέρουν σε δύσκολη θέση.

Όπως το σκάκι, το τάβλι έχει μελετηθεί με μεγάλο ενδιαφέρον από επιστήμονες της πληροφορικής. Λόγω αυτής της έρευνας, το λογισμικό για το τάβλι έχει αναπτυχθεί για να καταστεί εύκολο το παιχνίδι μεταξύ ανθρώπων μέσω Internet, όπως και για να νικήσει παίκτες παγκόσμιας κλάσης.

Οι ζαριές παρέχονται από τυχαίες ή ψευδό-τυχαίες (pseudorandom) γεννήτριες αριθμών.

Το OnLine παιχνίδι σε πραγματικό χρόνο ξεκίνησε με τον First Internet Backgammon Server τον Ιούλιο του 1992 από τον Andreas Schneider.
Ο First Internet Backgammon Server (FIBS) είναι ο μακροβιότερος server για τάβλι στο internet.

[3] Backgammon, wikipedia.org/wiki/Backgammon

1.3 ΣΥΣΤΑΤΙΚΑ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ [4]

Το παιχνίδι λέγεται Πόρτες γιατί ο στόχος είναι να δημιουργήσουμε «πόρτες» ώστε να μεταφέρουμε τα πούλια μας με ασφάλεια στην περιοχή μαζέματος.

«Πόρτα» ονομάζεται η στήλη που καταλαμβάνεται από 2 ή περισσότερα πούλια του ίδιου χρώματος. Αν ένα πούλι μείνει μόνο του σε μια στήλη, τότε είναι ελεύθερο να χτυπηθεί από τον αντίπαλο.

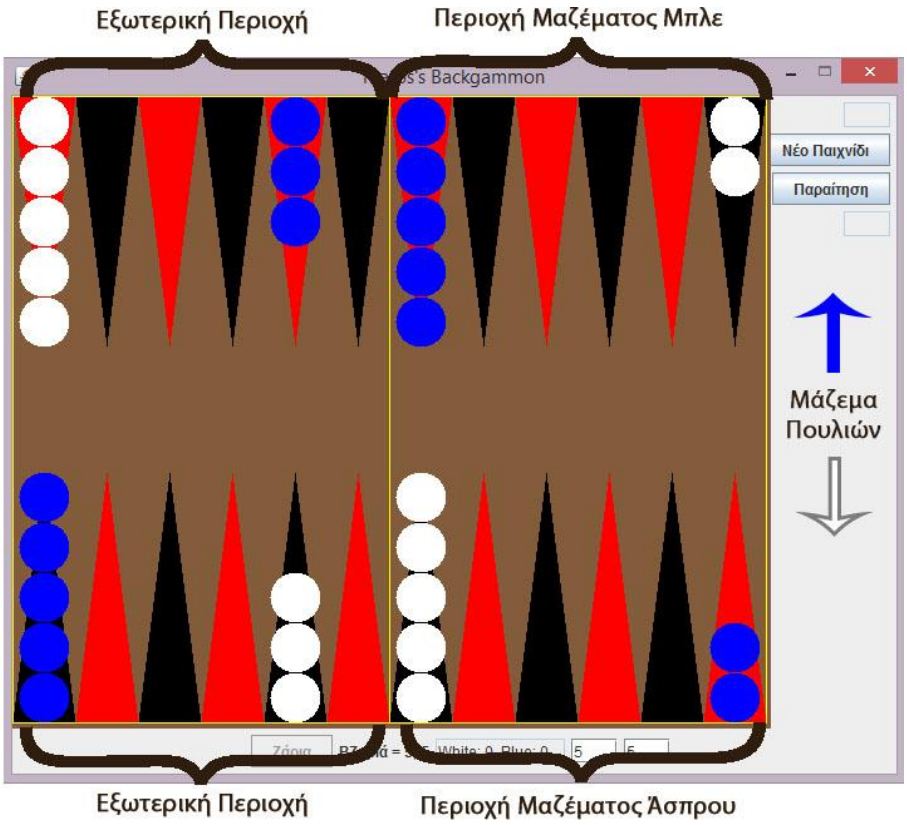
Οι Πόρτες παίζονται σε πίνακα (ταμπλό). Υπάρχουν 30 πούλια εκ των οποίων τα 15 είναι του πρώτου παίκτη και τα υπόλοιπα 15 του δεύτερου. Ο πίνακας χωρίζεται σε τέσσερις περιοχές με κάθε περιοχή να περιλαμβάνει έξι θέσεις, συνολικά δηλαδή 24 θέσεων, που έχουν κατά παράδοση τριγωνικό σχήμα. Το παιχνίδι παίζεται με δύο ζάρια.

[4] Πώς παίζονται οι Πόρτες στο Τάβλι (backgammon), foxcasino.gr/portes-tavli-backgammon/

1.4 ΣΤΗΣΙΜΟ

Το παιχνίδι «πόρτες» στήνεται όπως στην Εικόνα 2. Ο άσπρος παίκτης μαζεύει στην «Περιοχή Μαζέματος Άσπρου» και ο μπλε στην «Περιοχή Μαζέματος Μπλε». Κάθε παίκτης έχει απέναντί του την περιοχή εκκίνησης και στην μεριά του την περιοχή μαζέματος, με την περιοχή μαζέματος του ενός παίκτη να είναι η περιοχή εκκίνησης τού άλλου.

Ο ένας παίκτης κινείται δεξιόστροφα στη φορά των δεικτών του ρολογιού (στην εικόνα ο μπλε) ενώ ο άλλος (άσπρος) αριστερόστροφα, δηλαδή αντίθετα από την κίνηση των ωρολογιακών δεικτών.
Τα πιο μακρινά πούλια του άσπρου είναι τα δύο άσπρα πάνω δεξιά, ενώ 5 πούλια είναι ήδη στην «Περιοχή Μαζέματος Άσπρου».



Εικόνα 2 – Στήσιμο

1.5 ΚΑΝΟΝΕΣ

Το τάβλι είναι ένα από τα παλαιότερα επιτραπέζια παιχνίδια για δύο παίκτες. Τα πούλια κινούνται ανάλογα με τις ζαριές και ένας παίκτης κερδίζει όταν τα αφαιρέσει όλα από τον πίνακα πριν από τον αντίπαλό του.

Πιο συγκεκριμένα, το παιχνίδι «Πόρτες» αποτελείται από τους παρακάτω κανόνες: [5]

1. Στην αρχή του παιχνιδιού, δημιουργούνται δύο τυχαίοι αριθμοί από το 1 μέχρι το 6 (ένας για τον άσπρο και ένας για τον μπλε) και ο παίκτης που θα φέρει τον μεγαλύτερο παίζει πρώτος.
2. Δε μπορεί να γίνει κίνηση χωρίς να ριχτεί το ζάρι.
3. Τα πούλια προχωράνε μόνο μπροστά.
4. Το πούλι μπορεί να μετακινηθεί σε μια ελεύθερη στήλη.
5. Το πούλι μπορεί να μετακινηθεί σε στήλη που καταλαμβάνεται από πούλια του ίδιου χρώματος.
6. Το πούλι μπορεί να μετακινηθεί σε στήλη που καταλαμβάνεται από ένα μόνο πούλι του αντίπαλου χρώματος.
7. Αν σε μια θέση βρίσκεται μόνο ένα πούλι του παίκτη, ο παίκτης αυτός δεν κατέχει τη θέση και μπορεί να χτυπηθεί αν ένα πούλι του αντίπαλου έρθει στην ίδια θέση.

[5] Rules of Backgammon, bkgm.com/rules.html

8. Το χτυπημένο πούλι μετακινείται αυτόματα στο κέντρο.

    α. Αν ο παίκτης έχει χτυπημένο πούλι, δε μπορεί να κάνει καμία κίνηση , εάν το πούλι δεν μετακινηθεί ξανά στο ταμπλό.

    β. Αν ο παίκτης έχει χτυπημένο πούλι και φέρει ζαριά (μονή ή διπλή) η οποία δε μπορεί να παιχτεί, χάνει τη σειρά του.

9. Αν έρθουν διπλές, πρέπει να γίνουν τέσσερις κινήσεις.

10. Από τη στιγμή που ο παίκτης έχει μετακινήσει και τα 15 πούλια στην περιοχή του, ξεκινάει το μάζεμα.

    α. Όταν ξεκινήσει το μάζεμα, ο παίκτης αφαιρεί πούλια ανάλογα με τον αριθμό του ζαριού.

    β. Εάν το ζάρι φέρει μεγαλύτερο αριθμό από τις θέσεις στις οποίες υπάρχουν πούλια, θα πρέπει να αφαιρεθούν τα πούλια από την πιο μακρινή θέση εκείνη τη στιγμή.

11. Εάν χτυπηθεί ένα πούλι ενώ έχει ξεκινήσει το μάζεμα, πρέπει πρώτα το πούλι να γυρίσει πίσω στην περιοχή μαζέματος ώστε να ξαναξεκινήσει το μάζεμα.

12. Όταν ο παίκτης έχει μαζέψει και τα 15 πούλια, τότε κερδίζει (1 πόντος).

    α. Εάν ο παίκτης κερδίσει και ο αντίπαλος δεν έχει βάλει κανένα πούλι στην περιοχή μαζέματος, ο παίκτης παίρνει το παιχνίδι «διπλό» (2 πόντοι).

1.6 ΤΡΟΠΟΣ ΠΑΙΞΙΜΑΤΟΣ ΤΗΣ ΖΑΡΙΑΣ

Πριν κινήσει τα πούλια ένας παίκτης πρέπει να ρίξει δυο ζάρια. Στο τάβλι υπάρχουν δυο ειδών ζαριάς. Οι απλές (ή μονές) και οι διπλές.

Η ζαριά είναι απλή όταν οι ενδείξεις των δυο ζαριών είναι διαφορετικές (όπως 2-3 ή 4-5 ή 1-6).

Η ζαριά είναι διπλή όταν οι ενδείξεις των δύο ζαριών είναι ίδιες (όπως 1-1 ή 4-4 ή 6-6).

1.6.1 ΟΙ ΜΟΝΕΣ ΖΑΡΙΕΣ [6]

Ο παίκτης κινεί ένα πούλι (της επιλογής του) κατά την φορά της κίνησής του και κατά αριθμό θέσεων όσο η ένδειξη του ενός ζαριού. Δεν μπορεί να κινηθεί ένα πούλι πάνω σε πόρτα του αντιπάλου. Εν συνεχεία ο παίκτης παίζει με τον ίδιο τρόπο την ένδειξη του άλλου ζαριού (με το ίδιο ή διαφορετικό πούλι) οπότε και τελειώνει η σειρά του.

- Ο παίκτης έχει το δικαίωμα να κάνει τις δυο κινήσεις με όποια σειρά επιθυμεί.
- Ο παίκτης είναι υποχρεωμένος να παίξει και τις δύο κινήσεις.
    - Δεν επιτρέπεται να παίξει το ένα ζάρι έτσι ώστε να μην υπάρχει τρόπο να παίξει και το δεύτερο και εφόσον υπάρχει άλλος τρόπος να παίζει και τα δύο.
- Αν δεν υπάρχει νόμιμος τρόπος να παίξει οποιαδήποτε κίνηση, τελειώνει η σειρά του (δεν κινεί).
- Αν υπάρχει τρόπος να παίξει μόνο την μία κίνηση τότε παίζει μόνο αυτήν.

Υπάρχει περίπτωση να φέρει μια ζαριά έστω Χ-Υ (πχ 5-4) και να μην υπάρχει κανένας τρόπος να παίξει και τα δύο αλλά να υπάρχει τρόπος να παίξει είτε το ένα είτε το άλλο.

- Ο παίκτης είναι υποχρεωμένος να παίξει το μεγαλύτερο ζάρι.

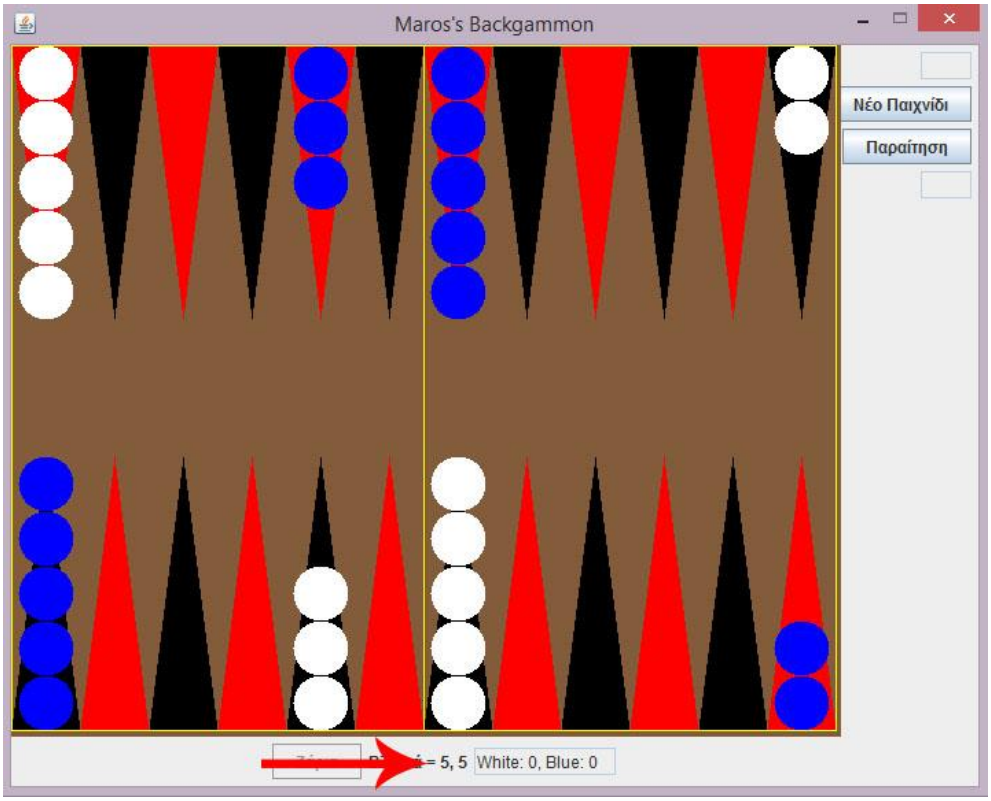
[6] Κανόνες του Ταβλιού, el.wikipedia.org/wiki/

1.6.2 ΟΙ ΔΙΠΛΕΣ ΖΑΡΙΕΣ [7]

Οι διπλές ζαριές παίζονται με τον ίδιο τρόπο με τις απλές ζαριές αλλά ο παίκτης παίζει τέσσερις φορές την (διπλή) ένδειξη του ζαριού. Για παράδειγμα αν έχει φέρει «πεντάρες» (5-5) παίζει τέσσερις φορές το πέντε.

Με την ολοκλήρωση των κινήσεών του τελειώνει και η σειρά του.

Αν ο παίκτης δεν έχει τρόπο να παίξει και τις τέσσερις κινήσεις τότε παίζει όσες έχει.



Εικόνα 3 – Οι διπλές ζαριές

[7] Κανόνες του Ταβλιού, el.wikipedia.org/wiki/

1.7 ΧΤΥΠΗΜΑ [8]

Όταν ένα πούλι βρίσκεται μόνο του σε μια θέση (δεν αποτελεί πόρτα) ο αντίπαλος έχει το δικαίωμα να κινήσει κάποιο δικό του πούλι σε αυτήν την θέση (σύμφωνα πάντα με τους κανόνες κίνησης).

Το πούλι απομακρύνεται από τον πίνακα και δίνεται "στο χέρι" του αντιπάλου (στην περίπτωση του πίνακά μας στο κέντρο).

Όταν ένας παίκτης έχει ένα ή περισσότερα πούλια στο χέρι δεν επιτρέπεται να κάνει άλλες κινήσεις πριν ξαναβάλει στον πίνακα όσα πούλια έχει στο χέρι.

Τα πούλια εισάγονται στην περιοχή εκκίνησης ανάλογα με το ζάρι. Ο παίκτης έχει το δικαίωμα να χρησιμοποιήσει όποια από τις δυο ενδείξεις των ζαριών που έφερε θέλει αλλά είναι υποχρεωμένος να το τοποθετήσει.

Στην περίπτωση που δεν μπορεί να το τοποθετήσει (γιατί υπάρχουν πόρτες του αντιπάλου στις θέσεις των ζαριών που έφερε) τελειώνει η σειρά του χωρίς να κινήσει (χάνει την σειρά του).

Στην περίπτωση που ο αντίπαλος έχει πόρτες και στις έξι θέσεις της περιοχής εκκίνησης, καμία ζαριά δεν μπορεί να επιτρέψει κίνηση και έτσι ο παίκτης χάνει τη σειρά του.

Ένας παίκτης μπορεί να ξεκινήσει το μάζεμα κανονικά όταν όλα τα πούλια του βρίσκονται στην περιοχή μαζέματος. Αν όμως στην διάρκεια του μαζέματος ο αντίπαλος χτυπήσει κάποιο πούλι του δεν επιτρέπεται να συνεχίσει το μάζεμα πριν το φέρει ξανά στην περιοχή μαζέματος.

[8] Πόρτες – Χτύπημα, el.wikipedia.org/wiki

1.8 ΒΑΘΜΟΛΟΓΙΑ [9]

Ο παίκτης που θα μαζέψει πρώτος όλα του τα πούλια νικά το παιχνίδι το οποίο και σταματά αυτομάτως.

Ο παίκτης που νικά το παιχνίδι κερδίζει ένα ή δύο βαθμούς (στην πρώτη περίπτωση λέμε το πήρε μονό ενώ στην δεύτερη διπλό).

Μονό είναι ένα παιχνίδι που ο χαμένος έχει προλάβει να κόψει (να μαζέψει έστω και ένα πούλι από τον πίνακα), ενώ διπλό είναι ένα παιχνίδι που ο χαμένος δεν έχει προλάβει να κόψει.

Εάν κάποιος παίκτης παραιτηθεί, ο αντίπαλός του κερδίζει το παιχνίδι διπλό.

[9] Κανόνες του Ταβλιού, el.wikipedia.org/wiki/

ΚΕΦΑΛΑΙΟ 2ο: ΣΥΣΤΑΤΙΚΑ AWT - SWING

2.1 AWT [10]

Το AWT είναι μέρος των JFC (Java Foundation Classes) από την Sun Microsystems, την εταιρία που δημιούργησε την Java και έχει αναπτυχθεί για τον προγραμματισμό GUI εφαρμογών.

Τα χαρακτηριστικά του περιλαμβάνουν:

- Δημιουργία συστατικών GUI όπως κουμπιά, ετικέτες, πλαίσια ελέγχου, μπάρες κύλισης, μενού κτλ.
- Χειρισμό συμβάντων που διαχειρίζεται τα γεγονότα που τροφοδοτούνται από ορισμένα συστατικά του GUI,
- Εργαλεία γραφικών και απεικόνισης,
- Χειριστές Layout για τον χειρισμό της διάταξης των στοιχείων στα παράθυρα, ανεξάρτητα από το μέγεθος παραθύρου και την ανάλυση οθόνης.

Το AWT όμως είναι platform-dependent (εξαρτημένο της πλατφόρμας), που σημαίνει ότι ένα πρόγραμμα γραμμένο σε AWT συμπεριφέρεται διαφορετικά σε διαφορετικές πλατφόρμες. Αυτό αντιτίθεται με το WORA (Write Once, Run Anywhere) που είναι η βασική φιλοσοφία της Java. [11]

Γι' αυτό το λόγο δημιουργήθηκε το Swing.

[10] AWT, docs.oracle.com

[11] Java Swings/AWT, en.wikibooks.org

2.2 SWING

Το Swing είναι μέρος των JFC και περιλαμβάνει διάφορα πακέτα για την ανάπτυξη πλούσιων desktop εφαρμογών σε Java. Τα στοιχεία του είναι γραμμένα εξ ολοκλήρου σε Java οπότε και είναι platform-independent (ανεξάρτητο της πλατφόρμας) που σημαίνει ότι συμπεριφέρεται το ίδιο σε διαφορετικές πλατφόρμες.

Το Swing περιλαμβάνει ενσωματωμένους ελέγχους όπως δέντρα, sliders, γραμμές εργαλείων, πίνακες κ.α. [12]

Επιπλέον, τα συστατικά του Swing προσφέρουν τα εξής πλεονεκτήματα σε σχέση με τα συστατικά του AWT:

- Η συμπεριφορά και εμφάνιση των συστατικών του Swing είναι ίδια σε όλες τις πλατφόρμες, ενώ του AWT διαφέρουν από πλατφόρμα σε πλατφόρμα,

- Στα συστατικά του Swing μπορεί να δοθεί το δικό τους "look and feel"[13]

- Τα συστατικά του Swing μπορούν να εκτελεστούν γρηγορότερα απ' ότι τα αντίστοιχα του AWT.

Συνοπτικά, το Swing είναι η πιο πρόσφατη εργαλειοθήκη GUI και προσφέρει πλουσιότερο σετ συστατικών διεπαφής σε σχέση με το AWT.

Από την άλλη πλευρά, τα συστατικά του Swing μπορεί να κάνουν περισσότερη ώρα να φορτωθούν σε σχέση με τα συστατικά του AWT.

---

[12] Java Swing, techopedia.com

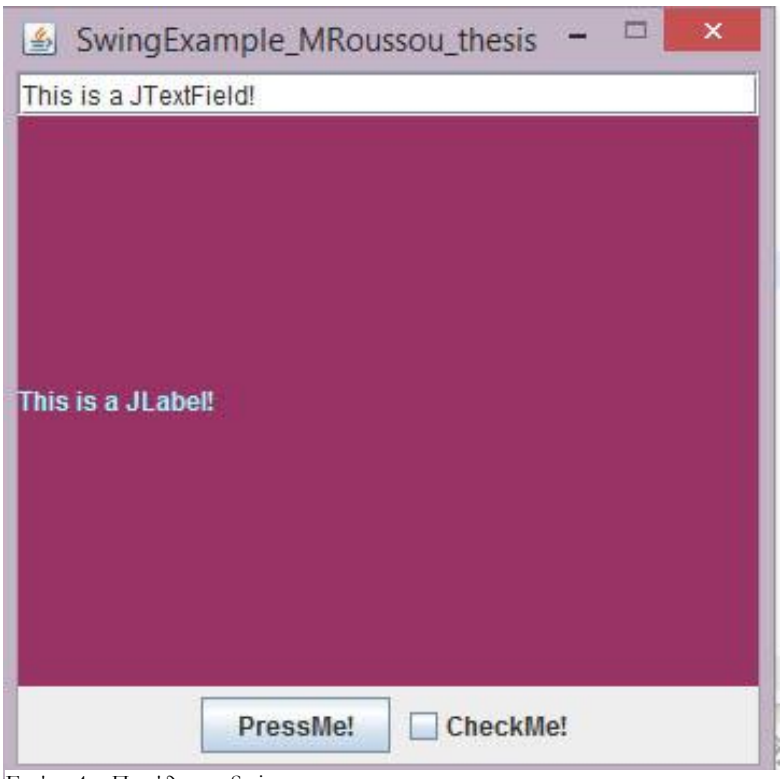
[13] How to Set the Look and Feel, "Look" refers to the appearance of GUI widgets (more formally, JComponents) and "feel" refers to the way the widgets behave"., docs.oracle.com

2.3 ΠΑΡΑΔΕΙΓΜΑ SWING

Παρακάτω υπάρχει μια λίστα με μερικά από τα συστατικά της βιβλιοθήκης Swing: [14]

- Παράθυρα (JFrame, JPanel)
- Ετικέτες (JLabel) και Κουμπιά (JButton)
- Λίστες επιλογών (JList)
- Μπάρες προόδου (JProgressBar)
- Πλαίσια Ελέγχου (JCheckBox)
- Ιεραρχικές δομές δένδρων (JTree)
- Πεδία κωδικών (JPasswordField)
- Αόρατα συστατικά (διαχειριστές διάταξης) κ.α.

Σε μια απλή εφαρμογή υλοποιώ ένα JFrame, το οποίο περιέχει ένα JPanel, ένα JTextField, ένα JLabel, ένα JButton και ένα JCheckBox όπως βλέπουμε στην Εικόνα 4.



Εικόνα 4 – Παράδειγμα Swing

[14] SWING – Controls, tutorialspoint.com/swing/swing_controls.htm

ΚΕΦΑΛΑΙΟ 3ο: ΚΛΑΣΕΙΣ, ΜΕΘΟΔΟΙ ΚΑΙ ΣΥΣΤΑΤΙΚΑ

3.1 ΟΙ ΚΛΑΣΕΙΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Η εφαρμογή περιέχει τρείς κλάσεις οι οποίες εξηγούνται περιληπτικά παρακάτω:

Η κλάση Tavli_MRoussou_thesis δημιουργεί το JFrame και τα JPanel που χρειάζονται για την εμφάνιση της εφαρμογής.

Συγκεκριμένα, καθορίζει το μέγεθος και τη θέση του παραθύρου όπως και άλλα συστατικά του και επίσης εμφανίζει δύο JPanel τα οποία χρησιμεύουν στην εμφάνιση των κουμπιών και διάφορων πληροφοριών και συμβουλών.

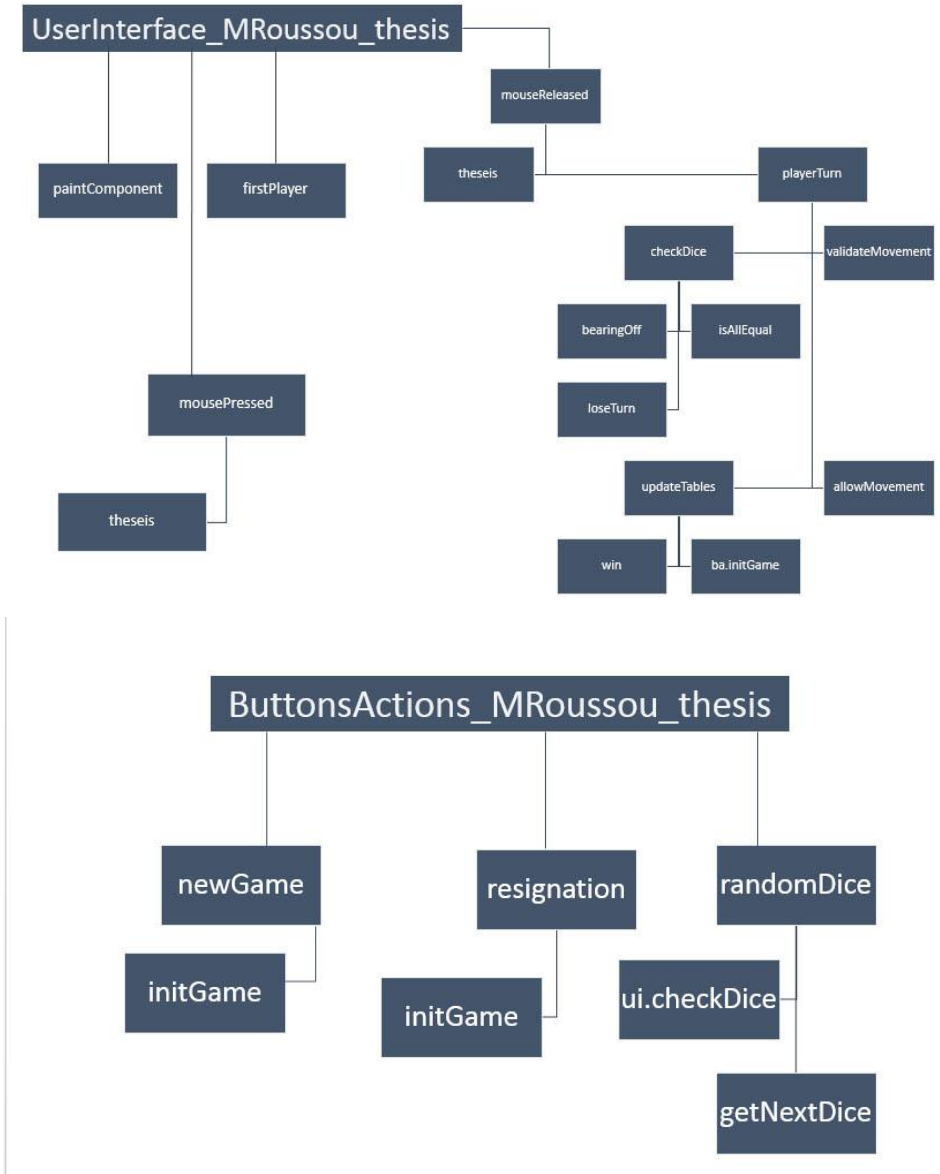
Στην εφαρμογή, υπάρχει ένα JPanel κάτω και ένα JPanel δεξιά (σχετικά με το JFrame), τα οποία περιλαμβάνουν κουμπιά για διάφορες ενέργειες όπως την έναρξη νέου παιχνιδιού και την παραίτηση κάποιου παίκτη. Επίσης υπάρχουν μερικές ετικέτες που χρησιμεύουν στην ενημέρωση του παίκτη σχετικά με το παιχνίδι - για παράδειγμα σε περιπτώσεις λάθος κινήσεων. Τέλος υπάρχουν πεδία κειμένου τα οποία εμφανίζουν στους παίκτες πληροφορίες σχετικά με το σκορ κ.α.

Η κλάση UserInterface_MRoussou_thesis σχεδιάζει όλον τον πίνακα του παιχνιδιού, τις στήλες όπως επίσης και τα πούλια. Η δεύτερη λειτουργία της είναι η υλοποίηση των κανόνων του παιχνιδιού.

Ενδεικτικά, ορίζει ποιος παίκτης θα ξεκινήσει πρώτος, πότε τελειώνει η σειρά κάποιου παίκτη, πότε κάποια κίνηση είναι λάθος, ζωγραφίζει τα πούλια όταν μετακινούνται στον πίνακα, εμφανίζει μηνύματα διαφόρων συμβουλών στις ετικέτες, ορίζει πότε κάποιος παίκτης έχει κερδίσει κ.α.

Η κλάση ButtonsActions_MRoussou_thesis δημιουργεί όλα τα κουμπιά που χρειάζονται για το παιχνίδι και τις ενέργειες που γίνονται όταν κάποιο κουμπί έχει πατηθεί. Επίσης διαθέτει μια μέθοδο η οποία όταν κληθεί από κάπου αλλού, αρχικοποιεί τον πίνακα και τα πούλια.

Στην Εικόνα 5 φαίνονται τα δέντρα για τις κλάσεις UserInterface_MRoussou_thesis και ButtonsActions_MRoussou_thesis.

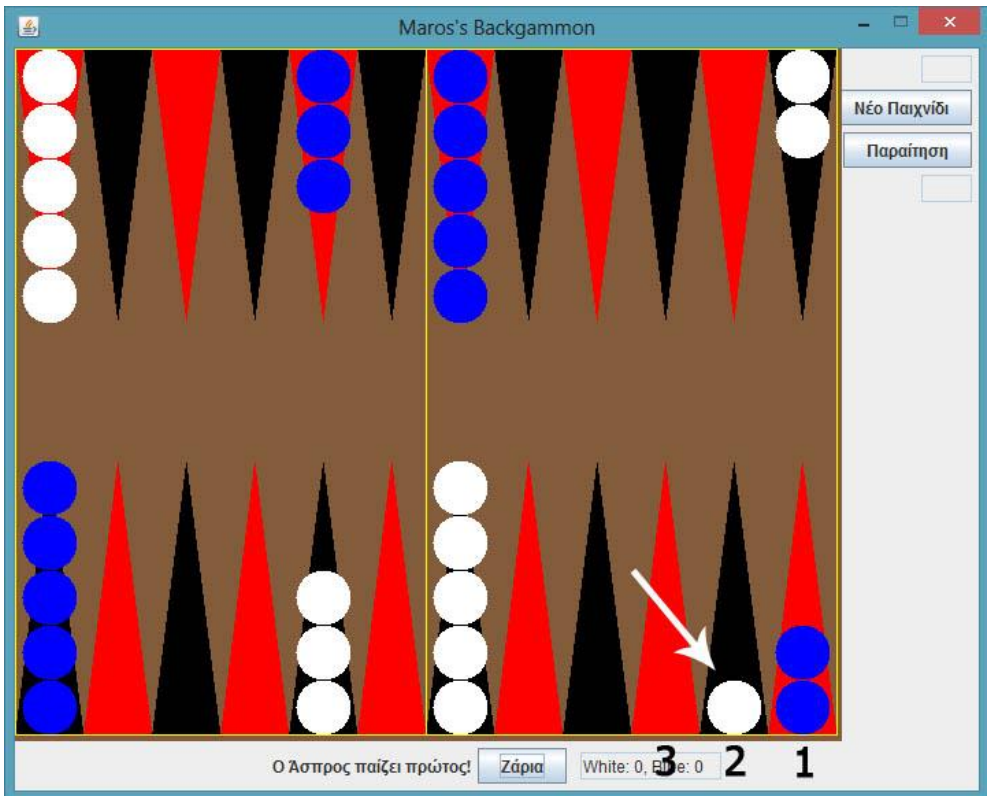


Εικόνα 5 – Δέντρα κλάσεων

Στην εφαρμογή υπάρχει η χρήση του «**Μοντέλου**» από το μοντέλο αρχιτεκτονικής λογισμικού **MVC (Model View Controller)**.

Συγκεκριμένα συναντάται στους πίνακες της εφαρμογής οι οποίοι όταν αλλάζουν, ζωγραφίζονται κάποια στοιχεία στον πίνακα.

Πιο αναλυτικά, όταν για παράδειγμα έχουμε τον πίνακα white[] = {0, **1**, 0, 0, 0, 5, ……} και στην $2^{η}$ θέση προσθέσουμε ένα πούλι, τότε στη στήλη 2 του πίνακα θα ζωγραφισθεί ένα πούλι όπως βλέπουμε στην Εικόνα 6.



Εικόνα 6 – Χρήση του «Model» από το μοντέλο «MVC»

3.2 Η ΚΛΑΣΗ Tavli_MRoussou_thesis

3.2.1 Η ΜΕΘΟΔΟΣ main (String[] args)

Στην main() μέθοδο της εφαρμογής δημιουργούνται το frame και τα panel που χρειάζονται για το γραφικό κομμάτι του κώδικα.

Η main() χρησιμοποιεί συστατικά των βιβλιοθηκών AWT (όπως το setLayout()), και Swing (όπως το JFrame και JPanel).

Συγκεκριμένα θέτω το background του frame σε καφέ χρώμα,, φτιάχνω τα όρια του frame και δηλώνω ότι το frame θέλω να κάνει "EXIT_ON_CLOSE", ώστε όταν κάποιος κλείνει το παράθυρο, να μην τρέχει η εφαρμογή στο background.

Έπειτα φτιάχνω αντικείμενο «ui» το οποίο χρειάζεται για την κλήση συστατικών από την κλάση UserInterface_MRoussou_thesis και το προσθέτω στο frame. Δηλώνω επίσης ότι θέλω το frame να μην είναι resizable – δηλαδή να μην υπάρχει η δυνατότητα αλλαγής μεγέθους του παραθύρου και επίσης να εμφανίζεται στο κέντρο της οθόνης.

Έπειτα δημιουργούνται τα δύο panel:

- Το πρώτο για την κάτω (σχετικά με την πλατφόρμα) περιοχή, στο οποίο θα εμφανίζεται το label που εμφανίζει μηνύματα λάθους και συμβουλών, το κουμπί «Ζάρια», το textField που εμφανίζει το σκορ των δύο παικτών και το label που εμφανίζει τον αριθμό των ζαριών.

- Το δεύτερο για την δεξιά περιοχή, στο οποίο εμφανίζονται τα κουμπιά «Νέο Παιχνίδι» και «Παραίτηση» όπως και τα textField για την εμφάνιση του αριθμού των πουλιών που έχουν μαζευτεί από τον κάθε παίκτη.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 7)

```
public class Tavli_MRoussou_thesis extends JFrame{
    public static void main (String[] args)
    {
        JFrame f = new JFrame("Maros's Backgammon");
        f.getContentPane().setBackground(Color.decode("#835C3B"));
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        UserInterface_MRoussou_thesis ui = new UserInterface_MRoussou_thesis();
        f.add(ui);
        ui.setOpaque(false);
        f.setSize(710, 570);
        f.setResizable(false);
        f.setLocationRelativeTo(null);
        f.setVisible(true);

        JPanel buttonPane = new JPanel();
        ButtonsActions_MRoussou_thesis ba = new ButtonsActions_MRoussou_thesis();
        buttonPane.add(ui.generalLab);
        buttonPane.add(ba.randomDice());
        buttonPane.add(ba.diceLab);
        buttonPane.add(ui.scoreTfield);
        if (ba.RANDOM_DICE==false){
            buttonPane.add(ba.customDie);
            buttonPane.add(ba.customDie2);
        }
        f.add(buttonPane, BorderLayout.SOUTH);

        JPanel buttonPaneRight = new JPanel();
        buttonPaneRight.add(ui.bearTfield2);
        buttonPaneRight.setPreferredSize(new Dimension(100, 100));
        buttonPaneRight.setMaximumSize(new Dimension(100, 100));
        buttonPaneRight.add(ba.newGame());
        buttonPaneRight.add(ba.resignation());
        buttonPaneRight.add(ui.bearTfield);
        buttonPaneRight.setLayout(new FlowLayout(FlowLayout.RIGHT));
        f.add(buttonPaneRight, BorderLayout.EAST);

        ba.ui = ui;
        ui.ba = ba;
    }
}
```

Εικόνα 7 – Ο κώδικας της main της Tavli_MRoussou_thesis

3.3 ΟΙ ΜΕΘΟΔΟΙ ΤΗΣ ΚΛΑΣΗΣ UserInterface_MRoussou_thesis

3.3.1 ΟΙ ΜΕΤΑΒΛΗΤΕΣ

Για την εφαρμογή έχουν δηλωθεί διάφορες μεταβλητές, πίνακες, labels και textFields τα οποία χρειάζονται για τον χειρισμό των κανόνων, την απεικόνιση συμβουλών στην πλατφόρμα και τον σχεδιασμό του πίνακα και των πουλιών.

Συγκεκριμένα το label “generalLab”, χρησιμοποιείται για την εμφάνιση γενικών συμβουλών και επιδείξεων στην πλατφόρμα.

Τα textFields:

- scoreTfield: Εμφανίζει στην πλατφόρμα το σκορ των δύο παικτών,
- bearTfield: Εμφανίζει τον αριθμό των μαζεμένων πουλιών του άσπρου παίκτη,
- bearTfield2: Εμφανίζει τον αριθμό των μαζεμένων πουλιών του μπλε παίκτη,

Οι μεταβλητές:

- counter: Χρησιμοποιείται για την μέτρηση κινήσεων σε περίπτωση διπλής ζαριάς,
- player: Χρησιμοποιείται για τον καθορισμό του τρέχων παίκτη,
- bearingOff: Μετράει τα πούλια στην περιοχή μαζέματος,
- move: Μετράει την απόσταση της τελικής από την αρχική στήλη,
- f1,f2: Χρησιμοποιούνται για την ενεργοποίηση και απενεργοποίηση των ζαριών,

- scoreW, scoreB: Αποθήκευση σκορ των παικτών,
- position: Χρησιμοποιείται για την αποθήκευση της θέσης ενός πουλιού,
- startingMove, endingMove: Αποθήκευση αρχικής και τελικής κίνησης,
- mostLeft: Αποθήκευση της πιο μακρινής στήλης που περιέχει πούλια (μόνο στην περιοχή μαζέματος),
- countFullColumns: Αποθήκευση συνόλου στηλών που περιέχουν πούλια,
- x, y, yk: Χρησιμοποιούνται για αποθήκευση συντεταγμένων,
- z1,z2: Αποθήκευση ζαριών,
- d1, d2: Χρησιμοποιούνται για την αποφυγή διπλότυπων τιμών στο ρίξιμο ζαριών κατά την εκκίνηση,

Οι πίνακες:

- columns[], columnsBackwards[]: Περιέχουν τις συντεταγμένες x των στηλών με την σωστή και την ανάποδη σειρά αντίστοιχα,
- white[],blue[]: Περιέχουν 26 θέσεις μία για κάθε στήλη (η 25 είναι το χτύπημα και η 26 το μάζεμα) και σε κάθε θέση αποθηκεύεται το πλήθος των πουλιών που περιέχεται στην αντίστοιχη στήλη,
- av2am[], a2m[]: Χρησιμοποιούνται για την αντιστοίχιση θέσεων από άσπρο σε μπλε και από μπλε σε άσπρο,
- canMoveZ1[], canMoveZ2[]: Αποθηκεύουν πληροφορίες οι οποίες χρησιμοποιούνται για να καθοριστεί εάν κάποιος παίκτης μπορεί να παίξει.

Τέλος ορίζεται το αντικείμενο “ba” το οποίο χρησιμοποιείται για την κλήση δεδομένων και μεθόδων από την κλάση ButtonsActions_MRoussou_thesis.

Ο κώδικας είναι ο παρακάτω: (Εικόνα 8)

```
JLabel generalLab;
JTextField scoreTfield, bearTfield, bearTfield2;

private int counter=0, player, bearingOff, move;
private boolean f1=false, f2=false;
int scoreW=0, scoreB=0, position, startingMove, endingMove, mostLeft, countFullColumns=0;
int x, y, yk, z1, z2, d1=0, d2=0;
int columns[] = {5, 55, 105, 155, 205, 255, 305, 355, 405, 455, 505, 555,605};
int columnsBackwards[] = {555, 505, 455, 405, 355, 305, 255, 205, 155, 105, 55, 5,605};

int white[] = {0, 0, 0, 0, 0, 5, 0, 3, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,0,0};
int blue[] = {0, 0, 0, 0, 0, 5, 0, 3, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,0,0};
int av2am [] = {0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25};
int a2m [] = {23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,24,25};

int canMoveZ1[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int canMoveZ2[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

ButtonsActions_MRoussou_thesis ba;
```

Εικόνα 8 – Ο κώδικας των μεταβλητών της UserInterface_MRoussou_thesis

3.3.2 Ο CONSTRUCTOR UserInterface_MRoussou_thesis()

Ο constructor της κλάσης UserInterface_MRoussou_thesis θέτει τα label και τα textFields που χρειάζονται για την απεικόνιση διάφορων συμβουλών στην οθόνη του χρήστη, και καλεί την μέθοδο firstPlayer() η οποία ορίζει ποιος παίκτης θα ξεκινήσει πρώτος.

Ο κώδικας του constructor είναι ο παρακάτω: (Εικόνα 9)

```
public UserInterface_MRoussou_thesis(){
    ba = new ButtonsActions_MRoussou_thesis();
    generalLab = new JLabel();
    scoreTfield = new JTextField(9);
    bearTfield = new JTextField(3);
    bearTfield2 = new JTextField(3);
    bearTfield.setEditable(false);
    bearTfield2.setEditable(false);
    scoreTfield.setEditable(false);
    firstPlayer();
}
```

Εικόνα 9 – Ο κώδικας του Constructor της UserInterface_MRoussou_thesis

3.3.3 ΜΕΘΟΔΟΙ SET-GET

Για την κλάση UserInterface_MRoussou_thesis δημιουργήθηκαν οι μέθοδοι setF1(boolean f1), setF2(boolean f2), setPlayer(int player), setCounter(int counter), setBearingOff(int bearingOff), setMovement(int move) οι οποίες όταν καλούνται θέτουν μια τιμή στις μεταβλητές f1, f2, player, counter, bearingOff και move αντίστοιχα.

Οι μέθοδοι getF1(), getF2(), getPlayer(), getCounter(), getBearingOff() και getMovement(), επιστρέφουν την τιμή των μεταβλητών f1, f2, player, counter, bearingOff και move αντίστοιχα.

Ο κώδικας των μεθόδων είναι ο παρακάτω: (Εικόνα 10)

```
public void setF1(boolean f1){
    this.f1=f1;
}
public void setF2(boolean f2){
    this.f2=f2;
}
public void setPlayer(int player){
    this.player=player;
}
public void setCounter(int counter){
    this.counter=counter;
}
public void setBearingOff(int bearingOff){
    this.bearingOff=bearingOff;
}
public void setMovement(int move){
    this.move=move;
}
public boolean getF1(){
    return f1;
}
public boolean getF2(){
    return f2;
}
public int getPlayer(){
    return player;
}
public int getCounter(){
    return counter;
}
public int getBearingOff(){
    return bearingOff;
}
public int getMovement(){
    return move;
}
```

Εικόνα 10 – Ο κώδικας των SET-GET της UserInterface_MRoussou_thesis

3.3.4 Η ΜΕΘΟΔΟΣ paintComponent(Graphics g)

Η μέθοδος paintComponent(Graphics g) κάνει όλες τις ενέργειες για να «ζωγραφιστούν» στην πλατφόρμα όλα τα γραφικά στοιχεία με τη χρήση της βιβλιοθήκης AWT.

Συγκεκριμένα:

- Δημιουργούνται οι πάνω και οι κάτω (σχετικά με την πλατφόρμα) στήλες με την χρήση του fillPolygon(arg1, arg2, arg3) το οποίο παίρνει τις συντεταγμένες x και y ως arg1 και arg2 αντίστοιχα και τον αριθμό των σημείων στο arg3, και «γεμίζει» ένα πολύγωνο με τα στοιχεία που του έχουμε δώσει.

- Στη συνέχεια δημιουργούνται τα πάνω και τα κάτω άσπρα πούλια και έπειτα τα πάνω και τα κάτω μπλε πούλια με την χρήση της fillOval(arg1, arg2, arg3, arg4), όπου arg1 και arg2 είναι οι x και y συντεταγμένες αντίστοιχα και arg3 και arg4 είναι το πλάτος και το ύψος του oval που δημιουργείται. Η fillOval() παίρνει τα στοιχεία που της έχουμε δώσει και «γεμίζει» ένα οβάλ.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνες 11 - 14)

Στην Εικόνα 10 βλέπουμε τη δημιουργία των πάνω στηλών χρησιμοποιώντας το fillPolygon().

```
public void paintComponent(Graphics g){
    super.paintComponent(g);
    this.addMouseListener(this);
    this.addMouseMotionListener(this);
    scoreTfield.setText("White: "+scoreW+", Blue: "+scoreB);

    //CREATION OF COLUMNS.
        //Upper Columns
        int yxpoints[] = {0, 0, 200};
        int x1points[] = {0, 50, 25};
        int npoints = 3;

        g.setColor(Color.RED);
        g.fillPolygon(x1points, yxpoints, npoints);

        for (int i=0; i<11; i++){
            for (int j=0; j<3; j++)
                x1points[j]=x1points[j]+50;
        if (i%2==0)
            g.setColor(Color.BLACK);
        else
            g.setColor(Color.RED);
        g.fillPolygon(x1points, yxpoints, npoints);
        }
```

Εικόνα 11 – Ο κώδικας της paintComponent(Graphics g)

Στην Εικόνα 12 βλέπουμε την δημιουργία των κάτω στηλών χρησιμοποιώντας το fillPolygon(), όπως και την δημιουργία του πλαισίου του πίνακα και της γραμμής στο κέντρο με την χρήση των drawRect() και drawLine() αντίστοιχα.

```
//Lower Columns
int ykpoints[] = {500,300,500};
int k1points[] = {0, 25, 50};

g.setColor(Color.BLACK);
g.fillPolygon(k1points, ykpoints, npoints);

for (int d=0; d<11; d++){
    for (int k=0; k<3; k++)
        k1points[k]=k1points[k]+50;
    if (d%2==0)
        g.setColor(Color.RED);
    else
        g.setColor(Color.BLACK);
    g.fillPolygon(k1points, ykpoints, npoints);
}

g.setColor(Color.YELLOW);
g.drawLine(300, 0, 300, 500);
g.drawRect (0, 0, 600, 500);
g.setColor(Color.BLUE);
```

Εικόνα 12 – Ο κώδικας της paintComponent(Graphics g)

Στην Εικόνα 13 βλέπουμε την δημιουργία των άσπρων πουλιών για τις επάνω και κάτω στήλες, χρησιμοποιώντας το fillOval()

```
//Upper White
g.setColor(Color.WHITE);
for (int a=0; a<14; a++){
    y=0;
    yk=200;
    for (int b=0; b<white[a+12]; b++){
        if (a==12){
            g.fillOval(293, yk, 15, 15);
            position=25;
            yk=yk+17;
        }
        else if (a==13)
            bearTfield.setText(" "+white[25]);
        else{
        g.fillOval(columns[a], y, 40, 40);
        y = y+40;
        }}}

//Lower White
for (int j=0; j<12; j++){
    y=460;
    for (int i=0; i<white[j]; i++){
        g.fillOval(columnsBackwards[j], y, 40, 40);
        y = y-40;
    }
}
```

Εικόνα 13 – Ο κώδικας της paintComponent(Graphics g)

Στην Εικόνα 14 βλέπουμε την δημιουργία των μπλε πουλιών για τις επάνω και κάτω στήλες με τη χρήση του fillOval().

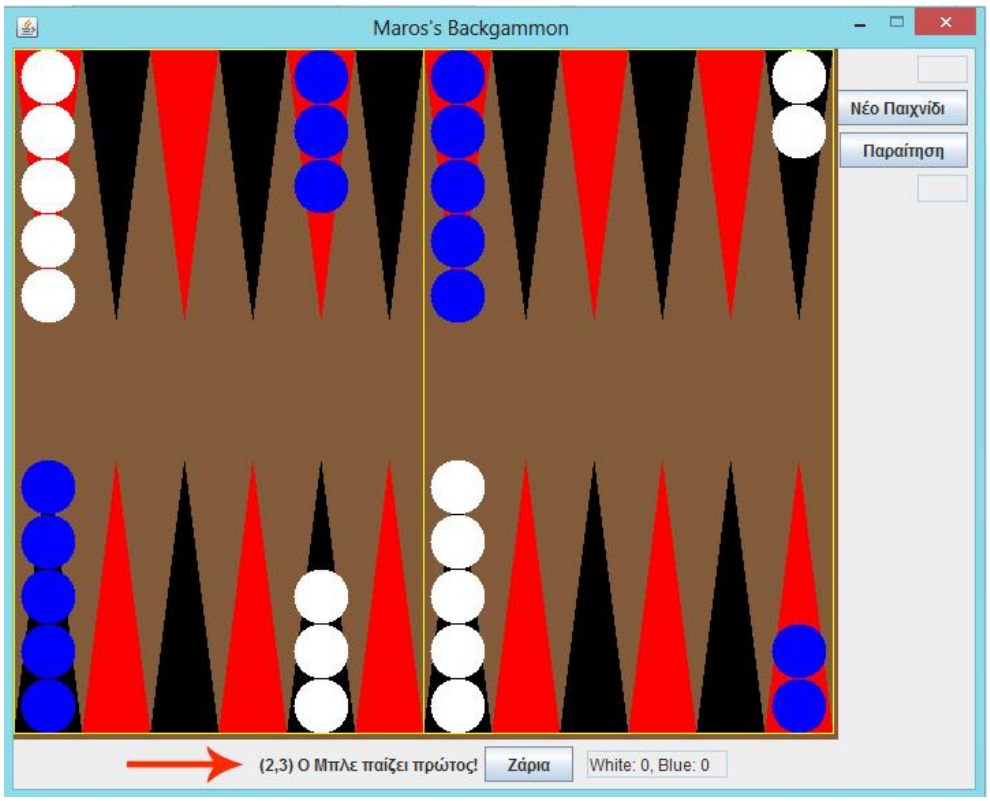
```
        //Upper Blue
        g.setColor(Color.BLUE);
        for (int a=0; a<14; a++){
            y=460;
            yk=285;
            for (int b=0; b<blue[a+12]; b++){
                if (a==12){
                    g.fillOval(293, yk, 15, 15);
                    position=25;
                    yk=yk-17;
                }
                else if (a==13){
                        bearTfield2.setText(" "+blue[25]);
                }
                else{
                g.fillOval(columns[a], y, 40, 40);
                y = y-40;
                }}}

        //Lower Blue
        for (int j=0; j<12; j++){
            y=0;
            for (int i=0; i<blue[j]; i++){
                g.fillOval(columnsBackwards[j], y, 40, 40);
                y = y+40;
            }
        }
}
```

Εικόνα 14 – Ο κώδικας της paintComponent(Graphics g)

3.3.5 Η ΜΕΘΟΔΟΣ firstPlayer()

Αρχικά δημιουργούνται δύο τυχαίοι αριθμοί (που όμως ποτέ δεν είναι ίσοι) στην κλάση firstPlayer() για να καθοριστεί ποιος παίκτης θα ξεκινήσει πρώτος.

Οι αριθμοί που προκύπτουν εμφανίζονται σε ένα label στον πίνακα όπως φαίνεται στην Εικόνα 15. Στο παράδειγμά μας έχουν δημιουργηθεί οι αριθμοί (2,3) όπου ο πρώτος (2) αντιστοιχεί στον άσπρο και ο δεύτερος (3) αντιστοιχεί στον μπλε. Γι' αυτό τον λόγο εμφανίζεται σε ένα label το μήνυμα: «Ο Μπλε παίζει πρώτος».



Εικόνα 15 – Καθορισμός πρώτου παίκτη

Συγκεκριμένα στον κώδικα δημιουργούνται δύο τυχαίοι αριθμοί από το 1 έως το 6.

Εάν ο πρώτος αριθμός είναι μεγαλύτερος από τον δεύτερο, παίζει ο Άσπρος παίκτης, εάν είναι μικρότερος, παίζει ο μπλε.

Η εντολή ελέγχου ροής while, ελέγχει εάν οι δύο αριθμοί που δημιουργήθηκαν είναι ίσοι, εάν αυτό ισχύει, τότε δημιουργούνται καινούριοι ώστε να αποφευχθεί η περίπτωση δύο ίδιων αριθμών.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 16).

```
public void firstPlayer(){
    while (d1==d2){ //To avoid same number
        d1=ba.rand.nextInt(6)+1;
        d2=ba.rand.nextInt(6)+1;
    }

    if (d1>d2){ //White player goes first
        generalLab.setText("("+d1+","+d2+")"+" Ο Άσπρος παίζει πρώτος!");
        setPlayer(0);
    }
    else if (d2>d1){ //Blue player goes first
        generalLab.setText("("+d1+","+d2+")"+" Ο Μπλε παίζει πρώτος!");
        setPlayer(1);
    }

    d1=0;
    d2=0;
}
```

Εικόνα 16 – Ο κώδικας της firstPlayer()

3.3.6 Η ΜΕΘΟΔΟΣ mousePressed(mouseEvent e)

Αφού το κουμπί «Ζάρια» έχει πατηθεί και εφόσον μπορούν να παιχτούν και τα δύο ζάρια, ο παίκτης πρέπει να μετακινήσει τα πούλια του τόσες θέσεις όσες υποδηλώνουν οι αριθμοί.

Όταν ο παίκτης πάει να «πιάσει» κάποιο πούλι, καλείται η μέθοδος mousePressed() η οποία παίρνει τις συντεταγμένες του «κλικ» πάνω στον πίνακα και τους αντιστοιχεί μια θέση (position) η οποία ορίζεται από την μέθοδο theseis(). Αυτή η θέση ορίζεται ως startingMove (αρχική θέση).

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 17)

```
public void mousePressed(MouseEvent e){
    x=e.getX();
    y=e.getY();
    if (!e.isConsumed()){
        System.out.println(""+x+","+y);
        theseis(x, y);

    startingMove=position;
    }
    e.consume();
}
```

Εικόνα 17 – Ο κώδικας της mousePressed()

3.3.7 Η ΜΕΘΟΔΟΣ mouseReleased(mouseEvent e)

Όταν ο παίκτης «αφήσει» το πούλι που «έπιασε» κάπου, καλείται η μέθοδος mouseReleased() η οποία παίρνει τις συντεταγμένες της θέσης που άφησε το πούλι πάνω στον πίνακα και τους αντιστοιχεί μια θέση (position) η οποία ορίζεται και αυτή από την μέθοδο theseis(). Αυτή η θέση ορίζεται ως endingMove (τελική θέση).

Έπειτα από την ανάθεση της αρχικής και τελικής θέσης, καλείται η μέθοδος playerTurn().

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 18)

```
public void mouseReleased(MouseEvent e){
    int dx = e.getX() - x;
      int dy = e.getY() - y;
    if (!e.isConsumed()){
        theseis(x+dx, y+dy);
        endingMove=position;
        e.consume();
        playerTurn();
    }
    e.consume();
}
```

Εικόνα 18 – Ο κώδικας της mouseReleased()

3.3.8 Η ΜΕΘΟΔΟΣ playerTurn()

Η μέθοδος playerTurn() αρχικά καλεί την μέθοδο checkDice() για να ελέγξει εάν κάποιο από τα δύο ή και τα δύο ζάρια δεν μπορούν να παιχτούν. Εάν αυτό ισχύει, ο παίκτης χάνει τη σειρά του.

Εάν τα ζάρια μπορούν να παιχτούν καλείται η μέθοδος validateMovement() η οποία ελέγχει εάν η κίνηση είναι έγκυρη λαμβάνοντας υπόψιν τους κανόνες του Ταβλιού, στην οποία έχουν οριστεί.

Εάν κάποιος κανόνας έχει παραβιαστεί, η μέθοδος validateMovement() επιστρέφει false, η κίνηση δεν γίνεται και ο παίκτης καλείται να κάνει πάλι κίνηση.

Εάν δεν έχει παραβιαστεί κανένας κανόνας, η ροή συνεχίζεται κανονικά.

Στη συνέχεια καλείται η μέθοδος updateTables() η οποία αφαιρεί ένα πούλι από την αρχική θέση και προσθέτει ένα πούλι στην τελική θέση.

Έπειτα ελέγχεται εάν η πρώτη ζαριά είναι ίση με τη δεύτερη. Εάν αυτό ισχύει η ζαριά θεωρείται διπλή και ο παίκτης πρέπει να κάνει τέσσερις κινήσεις. Σε αυτή την περίπτωση καλείται η μέθοδος allowMovement() η οποία εκτελεί κάποιους κανόνες σχετικά με την κίνηση και όταν έχουν γίνει και οι τέσσερις κινήσεις επιστρέφει στην μέθοδο playerTurn() και αλλάζει ο παίκτης.

Εάν η ζαριά ήταν μονή, τότε καλείται η μέθοδος allowMovement(), εκτελεί κάποιους κανόνες σχετικά με την κίνηση και όταν έχουν γίνει και οι δύο κινήσεις επιστρέφει στην μέθοδο playerTurn() και αλλάζει ο παίκτης.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 19)

```
public void playerTurn(){
    //Check if dice can be played
    checkDice();
    //Check if player follows the rules
    if (!validateMovement())
        return;
    repaint();
    //Add and remove pieces
    updateTables();
    //If dice are "double"
    if (ba.getDie1()==ba.getDie2()){
        setCounter(getCounter()+1);
        while(getCounter()!=4){
            if (!allowMovement())
                return;
        }
        setCounter(0);
    }
    //Check if player finished his movement
    else if (!allowMovement()){
        checkDice();
        setCounter(getCounter()+1);;
        return;
    }
    setCounter(0);
       if (player==0){
           setPlayer(1);
           JOptionPane.showMessageDialog(null, "Σειρά του Μπλε");
       }
       else{
           setPlayer(0);
           JOptionPane.showMessageDialog(null, "Σειρά του Άσπρου");
       }
   //Initialize components
   ba.zaria.setEnabled(true);
   generalLab.setText("");
   ba.diceLab.setText("");
   }
}
```

Εικόνα 19 – Ο κώδικας της playerTurn()

3.3.9 Η ΜΕΘΟΔΟΣ validateMovement()

Η μέθοδος validateMovement() καλείται μέσα στην playerTurn(), ορίζει τους κανόνες του Ταβλιού και επιστρέφει false όταν κάποιος τους έχει παραβεί.

Αρχικά αναθέτει στην μεταβλητή movement τα βήματα που έγιναν τα οποία μετριούνται αφαιρώντας την τελική από την αρχική.

Παρακάτω εξηγώ τους κανόνες που ορίζονται στην μέθοδο validateMovement():

- Όταν το κουμπί «Ζάρια» είναι ενεργό (που σημαίνει ότι δεν έχει πατηθεί) και ο παίκτης προσπαθήσει να κάνει κάποια κίνηση, η μέθοδος επιστρέφει false.

- Όταν ο τρέχων παίκτης έχει πούλι στο κέντρο (έχει χτυπηθεί) και η κίνησή του δεν ξεκινάει από το κέντρο, η μέθοδος επιστρέφει false.

- Εάν ο τρέχων παίκτης έχει πούλι στο κέντρο και η κίνησή του τελειώνει σε στήλη που ο αντίπαλος παίκτης έχει παραπάνω από ένα πούλια, η μέθοδος επιστρέφει false.

- Όταν ο τρέχων παίκτης μπορεί να μαζέψει πούλια, και η τελική θέση είναι η θέση μαζέματος και επιπλέον και τα δύο ζάρια είναι μεγαλύτερα από το mostLeft (το πιο μακρινό πούλι), τότε εάν η αρχική κίνηση δεν είναι ίση με το πιο μακρινό πούλι, η μέθοδος επιστρέφει false.

- Όταν ο τρέχων παίκτης μπορεί να μαζέψει πούλια, και η τελική θέση είναι η θέση μαζέματος και επιπλέον η αρχική θέση είναι ίση με το πρώτο ή το δεύτερο ζάρι (ή και με τα δύο), τότε εάν η αρχική είναι ίση με το πρώτο ζάρι, θέτουμε την κίνηση ίση με το πρώτο ζάρι. Αντίστοιχα εάν η αρχική είναι ίση με το δεύτερο ζάρι, θέτουμε την κίνηση ίση με το δεύτερο.

- Όταν ο τρέχων παίκτης δεν μπορεί να μαζέψει πούλια και εάν τα βήματα που έγιναν (movement) δεν είναι ίσα ούτε με το πρώτο ούτε με το δεύτερο ζάρι (αυτό σημαίνει ότι ο παίκτης έκανε περισσότερα ή λιγότερα βήματα από όσα έπρεπε), η μέθοδος επιστρέφει false.

- Όταν ο τρέχων παίκτης αφήσει κάποιο πούλι σε θέση στην οποία ο αντίπαλος παίκτης έχει ένα μόνο πούλι (χτύπημα), η μέθοδος επιστρέφει true.

- Όταν ο τρέχων παίκτης προσπαθήσει να μετακινήσει κάποιο πούλι από θέση στην οποία δεν υπάρχει πούλι, θεωρείται κενό ξεκίνημα και η μέθοδος επιστρέφει false.

- Όταν ο τρέχων παίκτης αφήσει κάποιο πούλι σε θέση στην οποία ο αντίπαλος παίκτης έχει παραπάνω από ένα πούλια, θεωρείται δεσμευμένη θέση και η μέθοδος επιστρέφει false.

- Όταν ο τρέχων παίκτης προσπαθήσει να μαζέψει κάποιο πούλι και τα πούλια στην περιοχή μαζέματος είναι λιγότερα από 15, η μέθοδος επιστρέφει false.

- Όταν ο τρέχων παίκτης έχει παίξει ήδη μια φορά κάποιο ζάρι και προσπαθήσει να παίξει το ίδιο ζάρι και δεύτερη φορά, τότε η μέθοδος επιστρέφει false.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνες 20-22)

Οι πίνακες current[] και other[] δημιουργούνται για να μπορούμε να ελέγξουμε ή και να αλλάξουμε κάποιες πληροφορίες για τον τρέχων ή τον αντίπαλο παίκτη, χωρίς όμως να χρειάζεται να ξέρουμε ποιος είναι συγκεκριμένα ο κάθε παίκτης. Αυτό βοηθάει για να γίνει ο κώδικας πιο γενικός άρα και πιο ευέλικτος.

```
private boolean validateMovement(){
    int current[];
    int other[];
    int moveStarts, moveEnds;

    if (player == 0){
        current = white;
        other = blue;
        moveStarts = av2am[startingMove];
        moveEnds = av2am[endingMove];
    }
    else{
        current = blue;
        other = white;
        moveStarts = av2am[startingMove];
        moveEnds = av2am[endingMove];
    }

    setMovement(startingMove-endingMove);
```

Εικόνα 20 – Ο κώδικας της validateMovement()

```
    //Check if dice are rolled
    if (ba.zaria.isEnabled()){
        generalLab.setText("Πρέπει να ρίξεις το ζάρι");
        return false;
    }


    //Return piece from center
    if (current[24] > 0){
        if (moveStarts!=24){
            generalLab.setText("Πρέπει να μετακινήσεις το χτυπημένο πούλι");
            return false;
        }
        else if(other[a2m[moveEnds]]>1 ){
            generalLab.setText("Λάθος τελείωμα");
            return false;
        }
    }

    //Check if movement is acceptable
    if(moveEnds==25 && (z1>mostLeft && z2>mostLeft) && getBearingOff() == 15){
            if (startingMove!=mostLeft){
                generalLab.setText("Πρέπει να παίξεις το πιο μακρινό πούλι");
                return false;
            }
    }
else if (moveEnds==25 && (startingMove==z1 || startingMove==z2) && getBearingOff() == 15){
    if (startingMove==z1)
        setMovement(z1);
    else if (startingMove==z2)
        setMovement(z2);
}
else if (moveEnds==25 && startingMove==mostLeft && getBearingOff() == 15){
    if (z1>mostLeft && z2<=mostLeft && (getF1() ^ getF2())){
        if (getF1())
            setMovement(z2);
        else if (getF2())
            setMovement(z1);
    }
    else if (z2>mostLeft && z1<=mostLeft && (getF1() ^ getF2())){
        if (getF1())
            setMovement(z2);
        else if (getF2())
            setMovement(z1);
    }
}
else if (getMovement()!=z1 && getMovement()!=z2){
    generalLab.setText("Τα βήματα πρέπει να είναι ίδια με το ζάρι");
    return false;
}
```

Εικόνα 21 – Ο κώδικας της validateMovement()

```
//Check if ending position is acceptable
if (other[a2m[moveEnds]]==1 && (moveEnds!=25 && other[25]==1)){
    System.out.println("Χτύπημα!");
    return true;
}

//Check if starting move is acceptable
        if (current[moveStarts] <= 0){
                generalLab.setText("Λάθος: Κενό Ξεκίνημα");
                return false;
        }

if (other[a2m[moveEnds]] > 1 && moveStarts!=24 && moveEnds!=25){
    generalLab.setText("Λάθος: Δεσμευμένο Τέλος");
    return false;
}

//Check if player can bear off
if (moveEnds==25){
if (getBearingOff() < 15){
    generalLab.setText("Δεν μπορείς να το κάνεις αυτό");
    setBearingOff(0);
    return false;
}}
```

Εικόνα 22 – Ο κώδικας της validateMovement()

3.3.10 Η ΜΕΘΟΔΟΣ allowMovement()

Η μέθοδος allowMovement() καλείται μέσα στην μέθοδο playerTurn() και ελέγχει εάν παίχτηκαν τα ζάρια και τα απενεργοποιεί (όταν κάποιο ζάρι απενεργοποιείται ο παίκτης δεν μπορεί να μετακινήσει κάποιο πούλι στον αριθμό του ζαριού αυτού).

Πιο συγκεκριμένα στη μέθοδο allowMovement() γίνονται οι εξής ενέργειες:

- Όταν ο παίκτης φέρει διπλή ζαριά, ελέγχεται ένας μετρητής. Μόλις ο μετρητής είναι 4 επιστρέφεται true και συνεχίζει η ροή (τελειώνει τη σειρά του ο παίκτης και συνεχίζει ο επόμενος). Εάν ο μετρητής δεν έχει γίνει 4, επιστρέφει false και ο παίκτης πρέπει να συνεχίσει να παίζει.

- Εάν η αρχική θέση δεν είναι και η πιο μακρινή, ή εάν η αρχική θέση είναι και η πιο μακρινή αλλά δεν έχει παιχτεί ένα από τα δύο ζάρια (ή κανένα από τα δύο) τότε:
    - Εάν η κίνηση (αρχική μείον τελική θέση) είναι ίση με το πρώτο ζάρι, τότε αυτό απενεργοποιείται
    - Αντίστοιχα εάν η κίνηση είναι ίση με το δεύτερο ζάρι, τότε αυτό απενεργοποιείται.

- Εάν η αρχική κίνηση είναι και η πιο μακρινή θέση και η τελική είναι η θέση μαζέματος τότε:

    - Εάν και τα δύο ζάρια είναι μεγαλύτερα από την πιο μακρινή στήλη που έχει πούλια, όταν ο παίκτης μετακινήσει το πούλι, τότε ελέγχεται εάν έχει παιχτεί το πρώτο και εάν αυτό ισχύει, απενεργοποιείται το δεύτερο, αλλιώς απενεργοποιείται μόνο το πρώτο ζάρι (δεν μπορεί να ξαναπαιχτεί κίνηση ίση με το πρώτο ζάρι). Αυτό συμβαίνει γιατί εφόσον και τα δύο ζάρια είναι μεγαλύτερα από την πιο μακρινή στήλη που περιέχει πούλια (άρα και αναγκαστικά η τελική θέση θα είναι και η θέση μαζέματος), δεν μας ενδιαφέρει τι ζαριές έχει φέρει ο παίκτης, απλά απενεργοποιούμε πρώτα την πρώτη και μετά την δεύτερη.

- Εάν δεν έχει παιχτεί κανένα από τα δύο ζάρια, ή δεν έχει παιχτεί μόνο ένα από τα δύο, τότε:

    - Εάν είτε το πρώτο είτε το δεύτερο ζάρι είναι μεγαλύτερο από την πιο μακρινή θέση τότε απενεργοποιείται είτε το πρώτο ζάρι είτε το δεύτερο, σύμφωνα με τα παραπάνω.

    - Εάν το πρώτο ζάρι είναι μεγαλύτερο από την πιο μακρινή θέση και το δεύτερο ζάρι είναι μικρότερο ή ίσο της πιο μακρινής θέσης, τότε απενεργοποιείται το πρώτο ζάρι

    - Αντίστοιχα, εάν το δεύτερο ζάρι είναι μεγαλύτερο από την πιο μακρινή θέση και το πρώτο ζάρι είναι μικρότερο ή ίσο της πιο μακρινής θέσης, τότε απενεργοποιείται το δεύτερο ζάρι.

- Τελικά εάν έχουν απενεργοποιηθεί και οι δύο ζαριές, τελειώνει η σειρά του τρέχων παίκτη, και ξαναενεργοποιούνται τα ζάρια ώστε να μπορεί να παίξει ο επόμενος.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνες 23 -24)

```
private boolean allowMovement(){
    //IF DOUBLE ROLL
    if (ba.getDie1()==ba.getDie2()){
        if (getCounter()==4){
            return true;
        }
        else
            return false;
    }
    //IF NOT DOUBLE ROLL
    else if (startingMove!=mostLeft
            || (startingMove==mostLeft && (!getF1() || !getF2()))){
        if (getMovement()==z1){
            setF1(true);
        }
        if(getMovement()==z2){
            setF2(true);
        }
    }
 if(startingMove==mostLeft && endingMove==26){
     if (z1>mostLeft && z2>mostLeft){
         if (getF1())
             setF2(true);
         setF1(true);
     }
     if ((!getF1() && !getF2()) || (!getF1() ^ !getF2())){
         if (z1>mostLeft ^ z2>mostLeft){
             if (z1>mostLeft){
                 setF1(true);
                 return false;
             }
             else if (z2>mostLeft){
                 setF2(true);
                 return false;
             }
         }
```

Εικόνα 23 – Ο κώδικας της allowMovement()

Στην περίπτωση που έχουν παιχτεί και τα δύο ζάρια όπως φαίνεται στην Εικόνα 24, αρχικοποιούνται πάλι έτσι ώστε να μπορέσει να κάνει κίνηση ο επόμενος παίκτης.

```
    if (z1>mostLeft && z2<=mostLeft){
        setF1(true);
        System.out.println("z1>mostLeft && z2<=mostLeft");
    }
    else if(z2>mostLeft && z1<=mostLeft){
        setF2(true);
        System.out.println("z2>mostLeft && z1<=mostLeft");
    }
}}
```

```
    //IF BOTH DICE ARE PLAYED
    if (getF1()==true && getF2()==true){
        setF1(false);
        setF2(false);
        return true;
    }
return false;
}
```

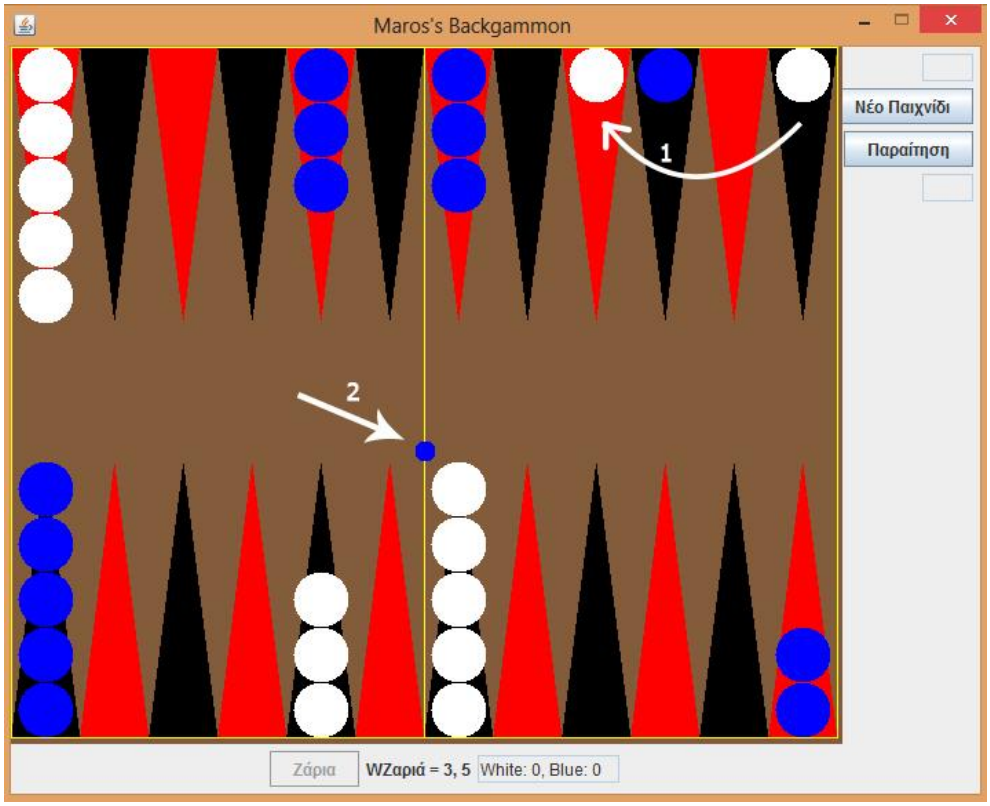
Εικόνα 24 – Ο κώδικας της allowMovement()

3.3.11 Η ΜΕΘΟΔΟΣ updateTables()

Η μέθοδος updateTables() όταν κληθεί μέσα στην playerTurn(), αφαιρεί ένα πούλι από την αρχική θέση και προσθέτει ένα πούλι στην τελική θέση.

Επίσης ελέγχει εάν ο αντίπαλος παίκτης έχει πούλι στην τελική θέση του τρέχων παίκτη.

Εάν αυτό ισχύει, τότε αφαιρείται ένα πούλι του αντιπάλου από την τελική θέση στην οποία ο τρέχων παίκτης άφησε κάποιο δικό του πούλι, και προστίθεται ένα πούλι του αντιπάλου στο κέντρο (χτύπημα) όπως φαίνεται στην Εικόνα 25.



Εικόνα 25 – Χτύπημα πουλιού

Η updateTables() επίσης καλεί την μέθοδο win() και εμφανίζει σε ένα label το σκορ των δύο παικτών.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 26)

```
public void updateTables(){
    int current[] = null;
    int other[] = null;

    if (player==0){
        current = white;
        other = blue;
    }
    else if (player==1){
        current = blue;
        other = white;
    }

    if (other[a2m[endingMove-1]] ==1 && endingMove!=26){
        --other[a2m[endingMove-1]];
        ++other[24];
    }
    //Add a piece where player dropped his piece
    ++current[endingMove-1];
    //Remove a piece from where player picked his piece
    --current[startingMove-1];

    win();
    scoreTfield.setText("Άσπρος: "+scoreW+", Μπλε: "+scoreB);

}
```

Εικόνα 26 – Ο κώδικας της updateTables()

3.3.12 Η ΜΕΘΟΔΟΣ bearingOff()

Η μέθοδος bearingOff() καλείται μέσα στην μέθοδο checkDice() και λειτουργία της είναι να μετράει πόσα πούλια βρίσκονται στην περιοχή μαζέματος.

Εάν βρίσκονται λιγότερα από 15 πούλια στην περιοχή μαζέματος του παίκτη τότε δεν κάνει τίποτα άλλο.

Στην περίπτωση που βρίσκονται και τα 15 πούλια μέσα στην περιοχή μαζέματος, η bearingOff() βρίσκει την πιο μακρινή στήλη που περιέχει πούλια και την αποθηκεύει στην μεταβλητή mostLeft. Η mostLeft μπορεί να χρησιμοποιηθεί αργότερα όταν ο παίκτης ξεκινήσει να μαζεύει.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 27)

```
public void bearingOff(){
    int current[];
    if (player==0)
        current=white;
    else
        current=blue;

        setBearingOff(0);
        for (int i=0; i<6; i++)
            setBearingOff(getBearingOff()+current[i]);
        //If player has 15 pieces on his area,
        //save the mostLeft non-empty column
        if (getBearingOff() == 15){
            for (int i=0; i<6; i++){
                if (current[i]!=0){
                    mostLeft=i;
                    mostLeft++;
                }
            }
        }
    }
```

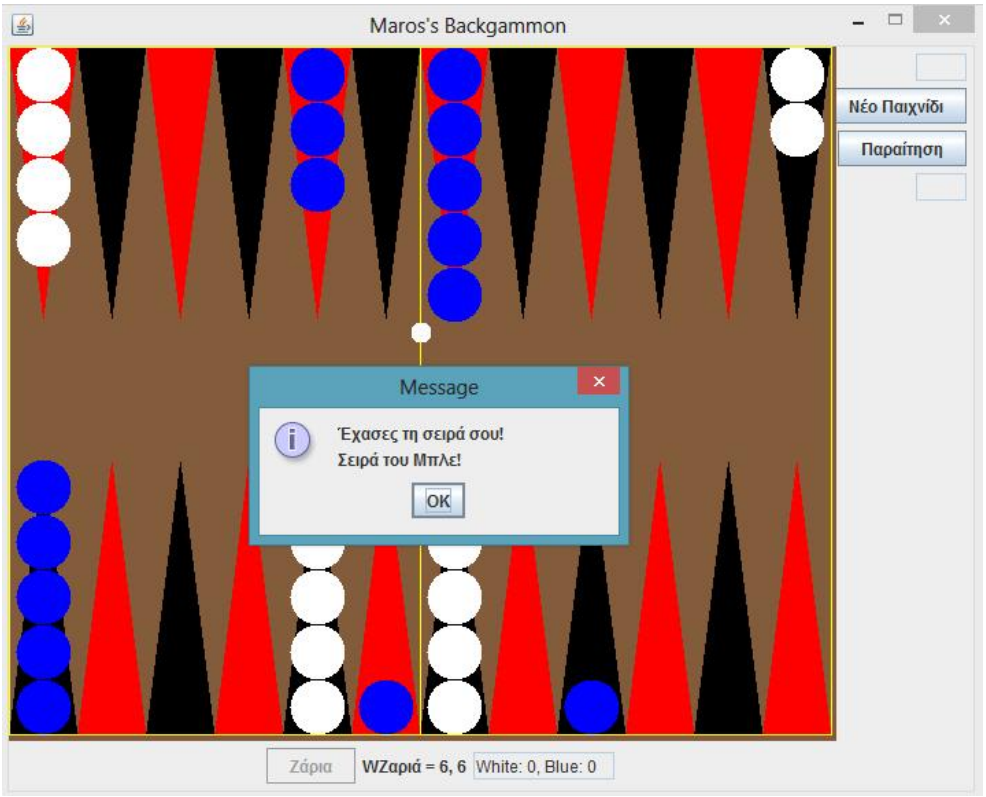
Εικόνα 27 – Ο κώδικας της bearingOff()

3.3.13 Η ΜΕΘΟΔΟΣ loseTurn()

Η συγκεκριμένη μέθοδος καλείται μέσα στην checkDice(), όταν οι κανόνες του Ταβλιού ορίζουν ότι ο παίκτης πρέπει να χάσει τη σειρά του.

Εάν η μέθοδος loseTurn() κληθεί, η σειρά δίνεται στον αντίπαλο παίκτη από αυτόν που παίζει την συγκεκριμένη στιγμή (στο παράδειγμά μας, ο άσπρος χάνει τη σειρά του και παίζει ο μπλε) και εμφανίζεται ένα παράθυρο μηνύματος που ενημερώνει τον τρέχων παίκτη ότι έχασε τη σειρά του. (Εικόνα 28)

Τέλος, αρχικοποιούνται τα ζάρια και το label, και ξαναενεργοποιείται το κουμπί «Ζάρια» ώστε να παίξει ο επόμενος παίκτης.



Εικόνα 28 – Τερματισμός σειράς

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 29)

```
public void loseTurn(){
    //If loseTurn() is called, current player loses his turn
    if (player==0){
        JOptionPane.showMessageDialog(null, "Έχασες τη σειρά σου! \nΣειρά του Μπλε!");
        setPlayer(1);
    }
    else if (player==1){
        JOptionPane.showMessageDialog(null, "Έχασες τη σειρά σου! \nΣειρά του Άσπρου!");
        setPlayer(0);
    }
    //Initialize components
    ba.diceLab.setText("");
    ba.zaria.setEnabled(true);
    setF1(false);
    setF2(false);
}
```

Εικόνα 29 – Ο κώδικας της loseTurn()

3.3.14 Η ΜΕΘΟΔΟΣ win()

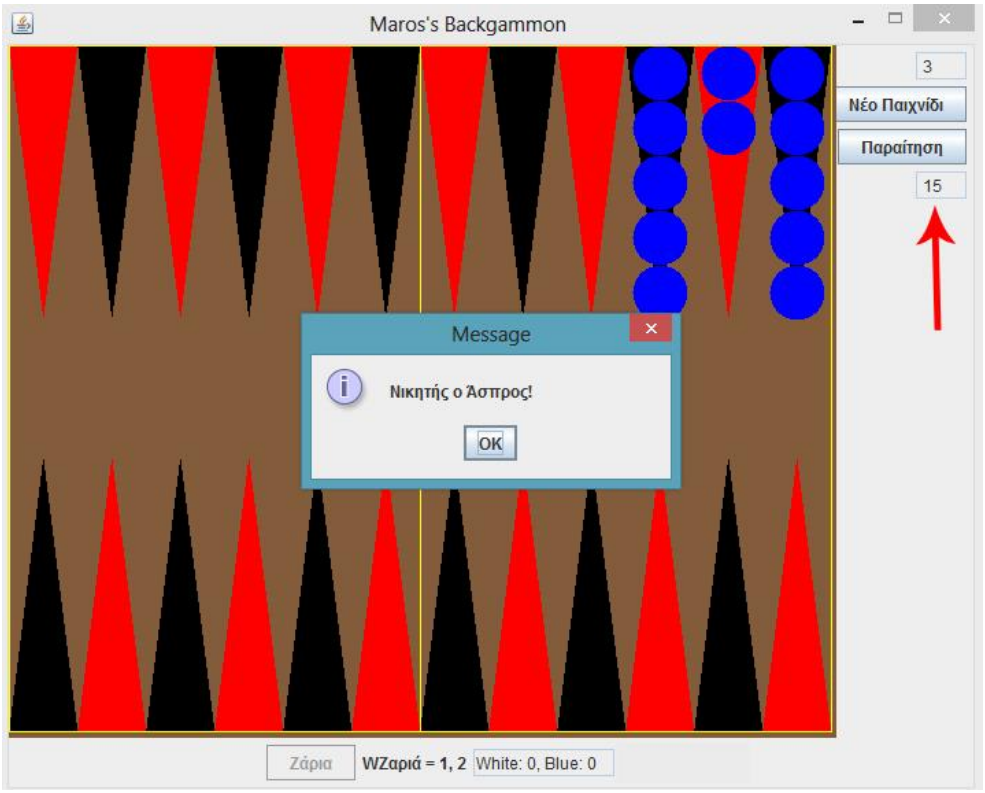
Η μέθοδος win() καλείται μέσα στην updateTables(). Όταν ο παίκτης έχει μαζέψει και τα 15 πούλια του, εμφανίζει ένα παράθυρο μηνύματος που λέει ότι κέρδισε ο τρέχων παίκτης, όπως φαίνεται στην Εικόνα 30.

Εάν ο αντίπαλος παίκτης δεν έχει μαζέψει κανένα πούλι, τότε ο τρέχων παίκτης κερδίζει το παιχνίδι διπλό (παίρνει 2 πόντους).

Εάν ο αντίπαλος παίκτης έχει μαζέψει έστω και ένα πούλι, τότε ο τρέχων παίκτης κερδίζει το παιχνίδι μονό (παίρνει 1 πόντο).

Οι πόντοι εμφανίζονται σε ένα label στο κάτω μέρος της πλατφόρμας.

Στο τέλος αρχικοποιείται ο πίνακας έχοντας κληθεί η μέθοδος initGame() της κλάσης buttonsActions.



Εικόνα 30 – Νίκη παίκτη

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 31)

```
public void win(){
    //If current==white && bear off is complete
    if (player==0 && white[25]==15){
            if (blue[25]==0)
                ++scoreW;
            JOptionPane.showMessageDialog(null, "Νικητής ο Άσπρος!");
            ++scoreW;
            ba.initGame();
    }
    //If current==blue && bear off is complete
    else if (player==1 && blue[25]==15){
            if (white[25]==0)
                ++scoreB;
            JOptionPane.showMessageDialog(null, "Νικητής ο Μπλε!");
            ++scoreB;
            ba.initGame();
    }
}
```

Εικόνα 31 – Ο κώδικας της win()

3.3.15 Η ΜΕΘΟΔΟΣ isAllEqual(int[] a, int plithos)

Η μέθοδος isAllEqual(int[] a, int plithos) καλείται μέσα στην checkDice() και χρησιμοποιείται για να ελέγξει εάν οι εγγραφές κάποιου πίνακα (a) έχουν τιμή διάφορη της προηγούμενης εγγραφής ή αν έχουν την τιμή 0. Τότε επιστρέφεται false και συνεχίζει ο παίκτης να παίζει.

Συγκεκριμένα, δεν μας ενδιαφέρει εάν ο πίνακας έχει έστω και μία τιμή μηδέν, γιατί αυτό σημαίνει ότι μπορεί να παιχτεί έστω και ένα ζάρι, άρα ο παίκτης συνεχίζει να παίζει.

Αυτό γίνεται γιατί μέσα στην μέθοδο checkDice() εάν κάποιο πούλι δεν μπορεί να μετακινηθεί ένας πίνακας ενημερώνεται με την τιμή 1.

Εάν όλες οι εγγραφές έχουν την τιμή 1 (δηλαδή εάν κανένα πούλι δεν μπορεί να παιχτεί, και η isAllEqual(int[] a, int plithos) επιστρέψει true), τότε ο παίκτης χάνει τη σειρά του.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 32)

```
public boolean isAllEqual(int[] a, int plithos){
    //If contents are equal or if any of
    //them==0, player can continue playing
    for (int i=1; i<plithos; i++){
        if (a[0]!=a[i] || a[i]==0)
            return false;
    }
        return true;
}
```

Εικόνα 32 – Ο κώδικας της isAllEqual (int[] a, int plithos)

3.3.16 Η ΜΕΘΟΔΟΣ checkDice()

Η μέθοδος checkDice() καλείται μέσα στην playerTurn() και ελέγχει εάν η ζαριά που έχει έρθει μπορεί να παιχτεί. Συγκεκριμένα, ελέγχει όλες τις στήλες που ο τρέχων παίκτης έχει έστω και ένα πούλι και υπολογίζει εάν από την κάθε στήλη μπορεί να παιχτεί το κάθε ζάρι.

Ένας πίνακας ενημερώνεται με την τιμή 1 για κάθε φορά που ένα πούλι δεν μπορεί να μετακινηθεί σε κάποια στήλη.

Η μέθοδος isAllEqual() καλείται για να βρει εάν όλες οι εγγραφές του πίνακα είναι ίδιες (εάν έχουν την τιμή 1).

Εάν όλες οι εγγραφές του πίνακα έχουν την τιμή 1, τότε καλείται η μέθοδος loseTurn() η οποία τερματίζει τη σειρά του τρέχων παίκτη.

Εάν ο τρέχων παίκτης έχει κάποιο πούλι στο κέντρο (χτυπημένο), η μέθοδος checkDice() επίσης ελέγχει για τον εάν είναι δυνατόν – σύμφωνα με τα ζάρια, να μετακινηθεί σε κάποια στήλη.

Στην περίπτωση που είναι αδύνατον να μετακινηθεί το πούλι σύμφωνα με το ένα από τα δύο ζάρια, τότε ο παίκτης είναι υποχρεωμένος να παίξει το άλλο, και μόλις τελειώσει την κίνηση χάνει τη σειρά του.

Όταν όμως ούτε το ένα, ούτε το άλλο ζάρι μπορεί να παιχτεί, τότε καλείται η μέθοδος loseTurn() και ο παίκτης χάνει τη σειρά του.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνες 33 - 35)

Στην μέθοδο checkDice() χρησιμοποιείται πάλι η λογική των πινάκων current[] και other[] για την ελαστικότητα του κώδικα.

```
public void checkDice(){
    bearingOff();
    z1=ba.getDie1();
    z2=ba.getDie2();
    int position1, position2, a=0;
    boolean W11=false, W12=false;

    int current[], other[];
    if (player==0){
        current=white;
        other=blue;
    }
    else{
        current=blue;
        other=white;
    }
```

Εικόνα 33 – Ο κώδικας της checkDice()

```
//If a column is not empty
for (int i=0; i<25; i++){
    if (current[i]>0){
        position1=i-z1;
        position2=i-z2;
        countFullColumns++;

        //If an endingMove is less than 0 (die1)
        if (position1<0){
            canMoveZ1[a]=1;
        }
        //If at an endingMove opponent has
        //more than one pieces (die1)
        else if(other[a2m[position1]]>1){
            canMoveZ1[a]=1;
    }
       //If an endingMove is less than 0 (die2)
       if (position2<0){
           canMoveZ2[a]=1;
       }
       //If at an endingMove opponent has
       //more than one pieces (die2)
       else if (other[a2m[position2]]>1){
           canMoveZ2[a]=1;
    }
    //If a die is already played
    if (getF1())
        canMoveZ1[a]=1;
    if (getF2())
        canMoveZ2[a]=1;
    a++;
    }
}
```

Εικόνα 34 – Ο κώδικας της checkDice()

Στην περίπτωση που ένα ή κανένα από τα δύο ζάρια δεν μπορεί να παιχτεί, τερματίζεται η σειρά του παίκτη.

```
    W11 = isAllEqual(canMoveZ1, countFullColumns);
    System.out.println (""+W11);
    W12 = isAllEqual(canMoveZ2, countFullColumns);
    System.out.println (""+W12);

//If isAllEqual==true, player loses his turn
if (getBearingOff()!=15){
    if (W11 && W12){
        loseTurn();
    }
}
//Initialize components
for(int i=0; i<25; i++){
    canMoveZ1[i]=0;
    canMoveZ2[i]=0;
    countFullColumns=0;
}
    //One of the dice cannot be played
    if (player==0 && white[24]>=1){
        if ((blue[z1-1]>1 && blue[z2-1]<=1 && getF2()==true)
                || (blue[z2-1]>1 && blue[z1-1]<=1 && getF1()==true)){
            loseTurn();
        }
        //Neither dice can be played
        else if(blue[z1-1]>1 && blue[z2-1]>1){
            loseTurn();
        }
    }

    //One of the dice cannot be played
    else if (player==1 && blue[24]>=1){
        if ((white[z1-1]>1 && white[z2-1]<=1 && getF2()==true)
                || (white[z2-1]>1 && white[z1-1]<=1 && getF1()==true)){
            loseTurn();
        }
        //Neither dice can be played
        else if(white[z1-1]>1 && white[z2-1]>1){
            loseTurn();
        }
    }
}
```

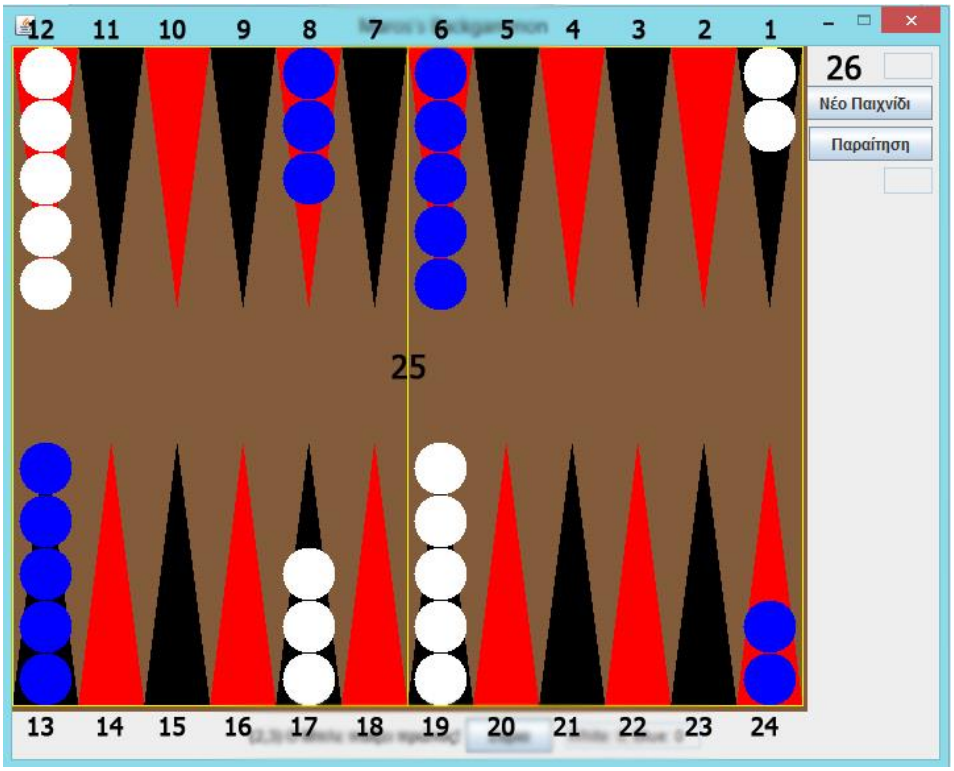
Εικόνα 35 – Ο κώδικας της checkDice()

3.3.17 Η ΜΕΘΟΔΟΣ theseis(int x, int y)

Η μέθοδος theseis() καλείται μέσα σε δύο άλλες μεθόδους, την mousePressed(mouseEvent e) για να αναγνωρίσει την θέση στην οποία «έπιασε» ο παίκτης κάποιο πούλι και η οποία ορίζεται ως αρχική και στην mouseReleased(mouseEvent e), για να αναγνωριστεί η θέση στην οποία ο παίκτης «άφησε» κάποιο πούλι και η οποία ορίζεται ως τελική. Οι θέσεις μετριούνται από το 1 ως το 24.

Στην εφαρμογή έχω προσθέσει δύο ακόμα θέσεις, την 25 και την 26, όπου η 25 είναι το κέντρο, εκεί δηλαδή που μεταφέρεται το χτυπημένο πούλι και η 26 είναι το μάζεμα, εκεί δηλαδή που μεταφέρει ο παίκτης τα πούλια του όταν τα έχει μαζέψει όλα πλέον στην περιοχή μαζέματος.

Στην Εικόνα 36, δείχνουμε την αρίθμηση για τον μπλε παίκτη. Η αρίθμηση για τον άσπρο παίκτη ξεκινάει με τον αριθμό 1 να βρίσκεται στην θέση που βρίσκεται το 24 για τον μπλε.



Εικόνα 36 – Οι θέσεις του Ταβλιού

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνες 37 - 40)

Παρακάτω φαίνεται η αρίθμηση των πουλιών για τον άσπρο παίκτη, και στις επόμενες σελίδες φαίνεται η αρίθμηση των πουλιών για τον μπλε παίκτη.

```
public void theseis(int x, int y){
    if (player==0){
    if(x>=0 && x<=50){
        if (y>=0 && y<=200)
            position=13;
        else if(y>=300 && y<=500)
            position=12;
     }
     else if (x>=50 && x<=100){
     if (y>=0 && y<=200)
            position=14;
        else if(y>=300 && y<=500)
            position=11;
     }
    else if (x>=100 && x<=150){
        if (y>=0 && y<=200)
            position=15;
        else if(y>=300 && y<=500)
            position=10;
    }
    else if (x>=150 && x<=200){
        if (y>=0 && y<=200)
            position=16;
        else if(y>=300 && y<=500)
            position=9;
    }
```

Εικόνα 37 – Ο κώδικας της theseis(int x, int y)

```
else if (x>=200 && x<=250){
    if (y>=0 && y<=200)
        position=17;
    else if(y>=300 && y<=500)
        position=8;
}
else if (x>=250 && x<=300 && y>=0 && y<=200)
    position=18;
else if(x>=250 && x<=300 && y>=300 && y<=500)
        position=7;
else if (x>=300 && x<=350 && y>=0 && y<=200)
        position=19;
else if(x>=300 && x<=350 && y>=300 && y<=500)
    position=6;
else if (x>=350 && x<=400 && y>=0 && y<=200)
        position=20;
else if(x>=350 && x<=400 && y>=300 && y<=500)
    position=5;
else if (x>=400 && x<=450 && y>=0 && y<=200)
        position=21;
else if (x>=400 && x<=450 && y>=300 && y<=500)
    position=4;
else if (x>=450 && x<=500 && y>=0 && y<=200)
        position=22;
else if(x>=450 && x<=500 && y>=300 && y<=500)
    position=3;
else if (x>=500 && x<=550 && y>=0 && y<=200)
    position=23;
```

Εικόνα 38 – Ο κώδικας της theseis(int x, int y)

```
 else if (x>=50 && x<=100){
 if (y>=0 && y<=200)
        position=11;
    else if(y>=300 && y<=500)
        position=14;
 }
else if (x>=100 && x<=150){
    if (y>=0 && y<=200)
        position=10;
    else if(y>=300 && y<=500)
        position=15;
}
else if (x>=150 && x<=200){
    if (y>=0 && y<=200)
        position=9;
    else if(y>=300 && y<=500)
        position=16;
}
else if (x>=200 && x<=250){
    if (y>=0 && y<=200)
        position=8;
    else if(y>=300 && y<=500)
        position=17;
}
else if (x>=250 && x<=300 && y>=0 && y<=200)
        position=7;
else if (x>=250 && x<=300 && y>=300 && y<=500)
    position=18;

else if(x>=500 && x<=550 && y>=300 && y<=500)
    position=2;
else if (x>=550 && x<=600){
    if (y>=0 && y<=200)
        position=24;
    else if(y>=300 && y<=500)
        position=1;
}
else if(x>=270 && x<=330 && y>=201 && y<=249)
        position=25;
else if(x>=600 && x<=650 && y>=300 && y<=500)
        position =26;
}


//BLUE PLAYER

if (player==1){
if(x>=0 && x<=50){
    if (y>=0 && y<=200)
        position=12;
    else if(y>=300 && y<=500)
        position=13;
 }
```

Εικόνα 39 – Ο κώδικας της theseis(int x, int y)

```
else if (x>=300 && x<=350 && y>=0 && y<=200)
        position=6;
else if(x>=300 && x<=350 && y>=300 && y<=500)
    position=19;
else if(x>=350 && x<=400 && y>=0 && y<=200)
        position=5;
else if (x>=350 && x<=400 && y>=300 && y<=500)
        position=20;
else if (x>=400 && x<=450 && y>=0 && y<=200)
        position=4;
else if (x>=400 && x<=450 && y>=300 && y<=500)
    position=21;
else if (x>=450 && x<=500 && y>=0 && y<=200)
        position=3;
else if(x>=450 && x<=500 && y>=300 && y<=500)
    position=22;
else if (x>=500 && x<=550){
    if (y>=0 && y<=200)
        position=2;
    else if(y>=300 && y<=500)
        position=23;
}
else if (x>=550 && x<=600){
    if (y>=0 && y<=200)
        position=1;
    else if(y>=300 && y<=500)
        position=24;
}
    else if (x>=550 && x<=600){
        if (y>=0 && y<=200)
            position=1;
        else if(y>=300 && y<=500)
            position=24;
    }
    else if(x>=270 && x<=330 && y>=250 && y<=299)
            position=25;
    else if(x>=600 && x<=650 && y>=0 && y<=200)
            position =26;
    }

    if (y>=200 && y<=300){
        if ((x>=0 && x<=269) || (x>=331 && x<=600))
        generalLab.setText("Λάθος ξεκίνημα");
    }
}
```

Εικόνα 40 – Ο κώδικας της theseis(int x, int y)

3.4 ΟΙ ΜΕΘΟΔΟΙ ΤΗΣ ΚΛΑΣΗΣ ButtonsActions_MRoussou_thesis

3.4.1 ΟΙ ΜΕΤΑΒΛΗΤΕΣ

Στην κλάση ButtonsActions_MRoussou_thesis έχουν δηλωθεί διάφορες μεταβλητές και συστατικά Swing όπως JTextField, JButtons και JLabel.

Συγκεκριμένα τα JButton:

- newGame: Όταν πατηθεί από τον χρήστη, γίνονται ενέργειες για να ξεκινήσει καινούριο παιχνίδι,
- zaria: Όταν πατηθεί από τον χρήστη, δημιουργούνται δύο τυχαίοι αριθμοί που χρησιμεύουν ως τα ζάρια του παιχνιδιού,
- resign: Όταν πατηθεί από τον χρήστη, ο τρέχων παίκτης παραιτείται και γίνονται ενέργειες για να ξεκινήσει καινούριο παιχνίδι και να κερδίσει έναν πόντο ο αντίπαλος παίκτης,

Το JLabel “diceLab” χρησιμοποιείται για να εμφανίσει στην οθόνη του χρήστη τις ζαριές που έχουν έρθει στο πάτημα του κουμπιού «zaria».

Τα JTextField «customDie» και «customDie2» μπορούν να χρησιμοποιηθούν όταν ο προγραμματιστής θέλει να κάνει κάποιες δοκιμές στο παιχνίδι – με αυτά μπορεί να δηλώσει χειροκίνητα τις ζαριές

Οι μεταβλητές:

- die1, die2: Αποθηκεύουν τις τιμές των ζαριών,
- RANDOM_DICE: Χρησιμοποιείται για τον καθορισμό της τυχαίας ή προγραμματισμένης ζαριάς – προγραμματισμένη είναι όταν την καθορίζει ο προγραμματιστής

- CUSTOMISED_START: Καθορισμός πίνακα ως «κλασσικού» (σε σχέση με το παιχνίδι «Πόρτες») ή ως προσαρμοσμένου στην περίπτωση που έχει οριστεί από τον προγραμματιστή.

Τέλος, θέτω το αντικείμενο «UI», το οποίο χρησιμοποιείται για την κλήση μεταβλητών και μεθόδων από την κλάση UserInterface_MRoussou_thesis.

Ο κώδικας είναι ο παρακάτω: (Εικόνα 41)

```
JButton newGame, zaria, resign;
JLabel diceLab;
JTextField customDie, customDie2;
Random rand;

private int die1=0, die2=0;
public UserInterface_MRoussou_thesis ui;

public final boolean RANDOM_DICE=true;
private final boolean CUSTOMISED_START=false;
```

Εικόνα 41 – Ο κώδικας των μεταβλητών της ButtonsActions_MRoussou_thesis

3.4.2 Ο CONSTRUCTOR ButtonsActions_MRoussou_thesis()

Ο constructor της κλάσης ButtonsActions_MRoussou_thesis το μόνο που κάνει είναι να αρχικοποιεί τα δύο textFields στα οποία μπορεί ο προγραμματιστής να δώσει δικές του τιμές στα ζάρια (αν είναι false η μεταβλητή RANDOM_DICE) ώστε να δοκιμάσει κάποιες περιπτώσεις.

Ο κώδικας του constructor είναι ο παρακάτω: (Εικόνα 42)

```
public ButtonsActions_MRoussou_thesis(){
    newGame = new JButton();
    zaria = new JButton("Ζάρια");
    diceLab = new JLabel();
    customDie = new JTextField(3);
    customDie2 = new JTextField(3);
    resign = new JButton("Παραίτηση");
    rand = new Random();
}
```

Εικόνα 42 – Ο κώδικας του Constructor της ButtonsActions_MRoussou_thesis

3.4.3 ΜΕΘΟΔΟΙ SET-GET

Παρακάτω βλέπουμε τις μεθόδους setDie1 (int die1) και setDie2(int die2) οι οποίες όταν καλούνται θέτουν μια τιμή στις μεταβλητές die1 και die2 αντίστοιχα, όπως επίσης και τις μεθόδους getDie1() και getDie2()επιστρέφουν τις μεταβλητές die1 και die2 αντίστοιχα.

Ο κώδικας είναι ο παρακάτω: (Εικόνα 43)

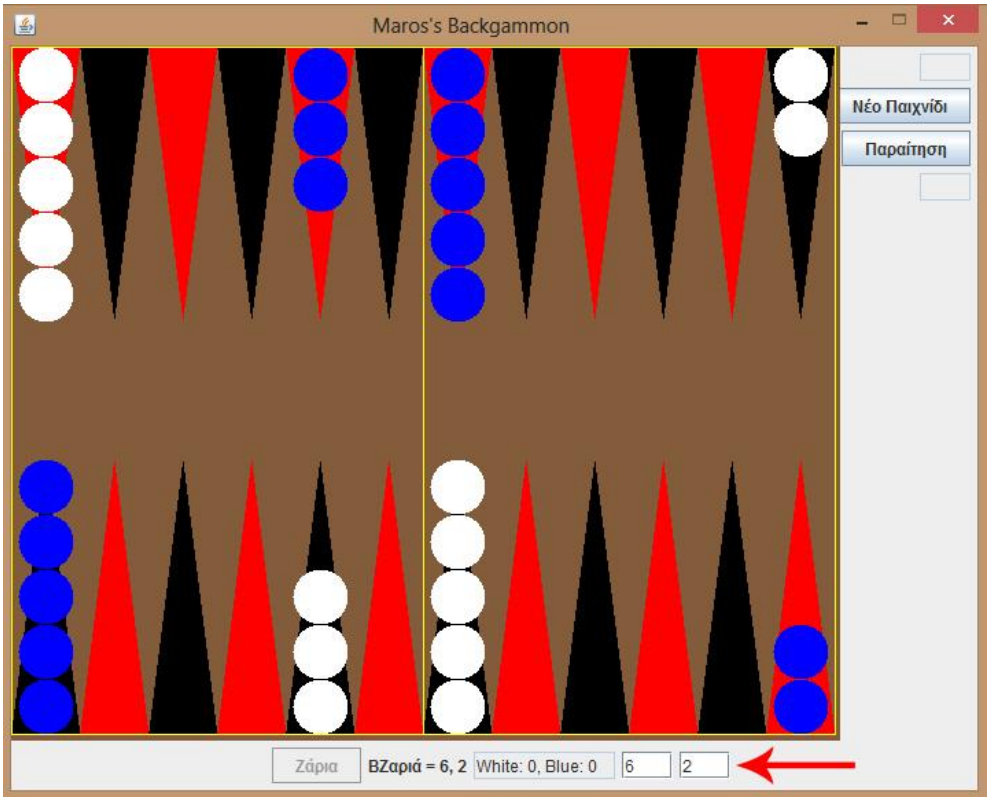
```
public void setDie1(int die1){
    this.die1 = die1;
}
public void setDie2(int die2){
    this.die2 = die2;
}

public int getDie1(){
    return die1;
}
public int getDie2(){
    return die2;
}
```

Εικόνα 43 – Ο κώδικας των SET-GET της ButtonsActions_MRoussou_thesis()

3.4.4 ΜΕΘΟΔΟΣ getNextDice(int n)

Η μέθοδος getNextDice(int n) δημιουργήθηκε για την ευκολία του προγραμματιστή στην υλοποίηση διαφόρων περιπτώσεων.

Εάν ο προγραμματιστής έχει θέσει false την μεταβλητή RANDOM_DICE τότε μπορεί και να επιλέξει εκείνος τι ζαριά θέλει να παίξει, γράφοντας την επιλογή του στα δύο textField που εμφανίζονται στο Panel, όπως φαίνεται στην Εικόνα 44.



Εικόνα 44 – Επιλογή ζαριάς από τον προγραμματιστή

Εάν ο προγραμματιστής έχει θέσει true την μεταβλητή RANDOM_DICE τότε δημιουργούνται δύο τυχαίοι αριθμοί.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 45)

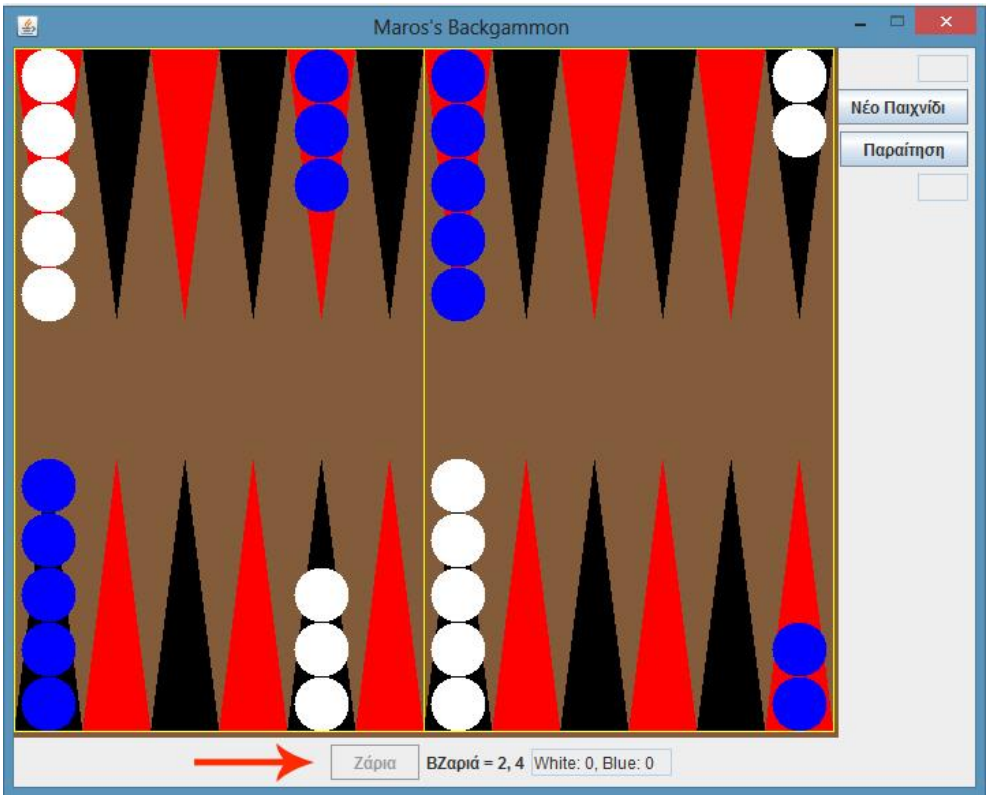
Η μεταβλητή n χρησιμεύει για να αναγνωρισθεί εάν πληκτρολογήθηκε αριθμός για το πρώτο ή για το δεύτερο ζάρι. Εάν το n είναι 0, τότε πληκτρολογήθηκε αριθμός για το πρώτο ζάρι, ενώ αν δεν είναι 0, για το δεύτερο.

```
private int getNextDice(int n){
    if (RANDOM_DICE==true){
        return rand.nextInt(6)+1;
    }
    else{
        if (n==0)
            return Integer.parseInt(customDie.getText());
        else
            return Integer.parseInt(customDie2.getText());
    }
```

Εικόνα 45 – Ο κώδικας της getNextDice(int n)

3.4.5 Η ΜΕΘΟΔΟΣ randomDice()

Ο τρέχων παίκτης θα πρέπει να πατήσει το κουμπί «Ζάρια» και η actionPerformed του, θέτει στις μεταβλητές die1 και die2 τις τιμές που εισάχθηκαν από την μέθοδο getNextDice() (2 τυχαίους ή 2 custom αριθμούς). Τότε εμφανίζονται σε ένα label οι τιμές των ζαριών όπως φαίνεται στην Εικόνα 46.



Εικόνα 46 – Απενεργοποίηση κουμπιού «Ζάρια»

Όταν το κουμπί «Ζάρια» πατιέται, η μέθοδος randomDice() το απενεργοποιεί έτσι ώστε να αποτραπεί ο παίκτης από να το ξαναπατήσει.

Επίσης μέσα από την μέθοδο randomDice() καλείται η μέθοδος checkDice() της κλάσης UserInterface_MRoussou_thesis, η οποία ελέγχει εάν κάποιο από τα ζάρια ή και τα δύο δεν μπορούν να παιχτούν. Εάν συμβαίνει αυτό, καλείται η μέθοδος loseTurn(), η οποία τερματίζει την σειρά του παίκτη.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 47)

Για τον καθορισμό των ζαριών χρησιμοποιείται η μέθοδος getNextDice(n), η οποία αναλόγως εάν η μεταβλητή RANDOM_DICE είναι true ή false, ορίζει εάν οι ζαριές θα είναι τυχαίες ή εάν θα τις καθορίσει ο προγραμματιστής.

```
public JButton randomDice(){
    zaria.addActionListener(new ActionListener() {

        public void actionPerformed(ActionEvent e){
            setDie1(getNextDice(0));
            setDie2(getNextDice(1));
            ui.generalLab.setText("");
            if (ui.getPlayer() == 0)
                diceLab.setText("WΖαριά = "+ getDie1() +", "+getDie2());
            else if (ui.getPlayer() == 1)
                diceLab.setText("BΖαριά = "+ getDie1() +", "+getDie2());
            zaria.setEnabled(false);
            ui.checkDice();
        }
    });
return zaria;
}
```

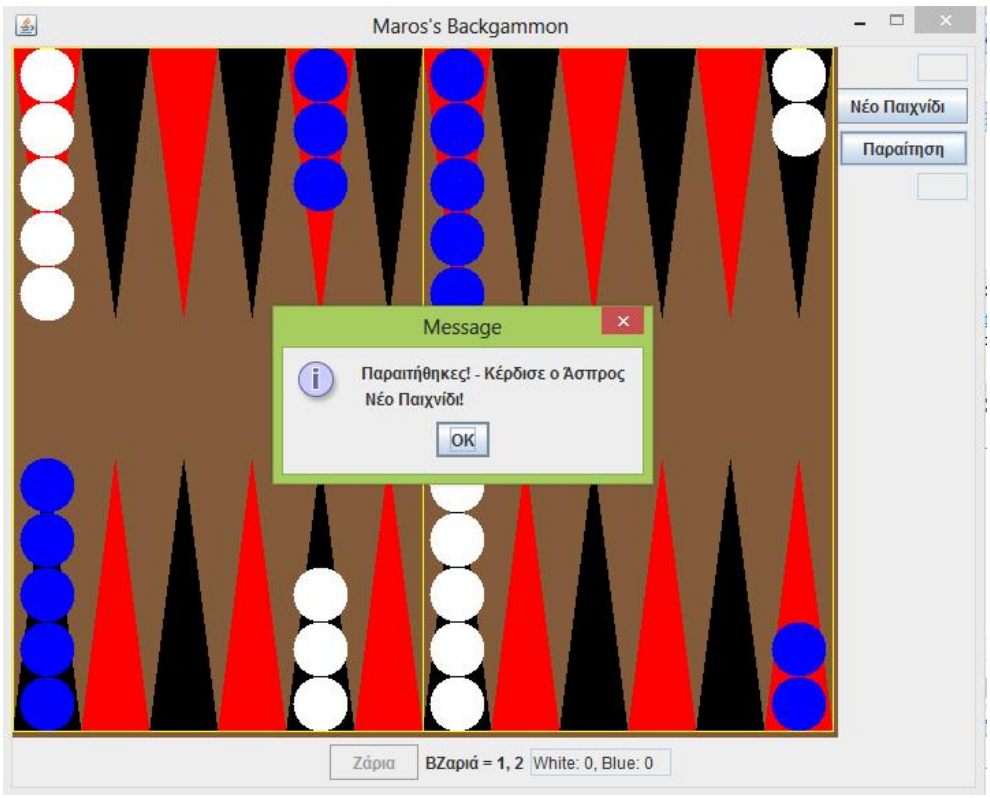
Εικόνα 47 – Ο κώδικας της randomDice()

3.4.6 Η ΜΕΘΟΔΟΣ resignation()

Όταν ο παίκτης πατήσει το κουμπί «Παραίτηση», καλείται η μέθοδος resignation() της κλάσης ButtonsActions_MRoussou_thesis.

Η μέθοδος ελέγχει ποιος είναι ο παίκτης και προσθέτει έναν βαθμό (ή δύο βαθμούς αναλόγως εάν είναι μονή ή διπλή η νίκη) στον αντίπαλό του.

Επίσης εμφανίζει ένα μήνυμα για να τον ενημερώσει ότι παραιτήθηκε και κέρδισε ο αντίπαλος όπως φαίνεται στην Εικόνα 48.



Εικόνα 48 – Παραίτηση παίκτη

Τέλος αρχικοποιείται ο πίνακας μέσω της μεθόδου initGame().

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 49)

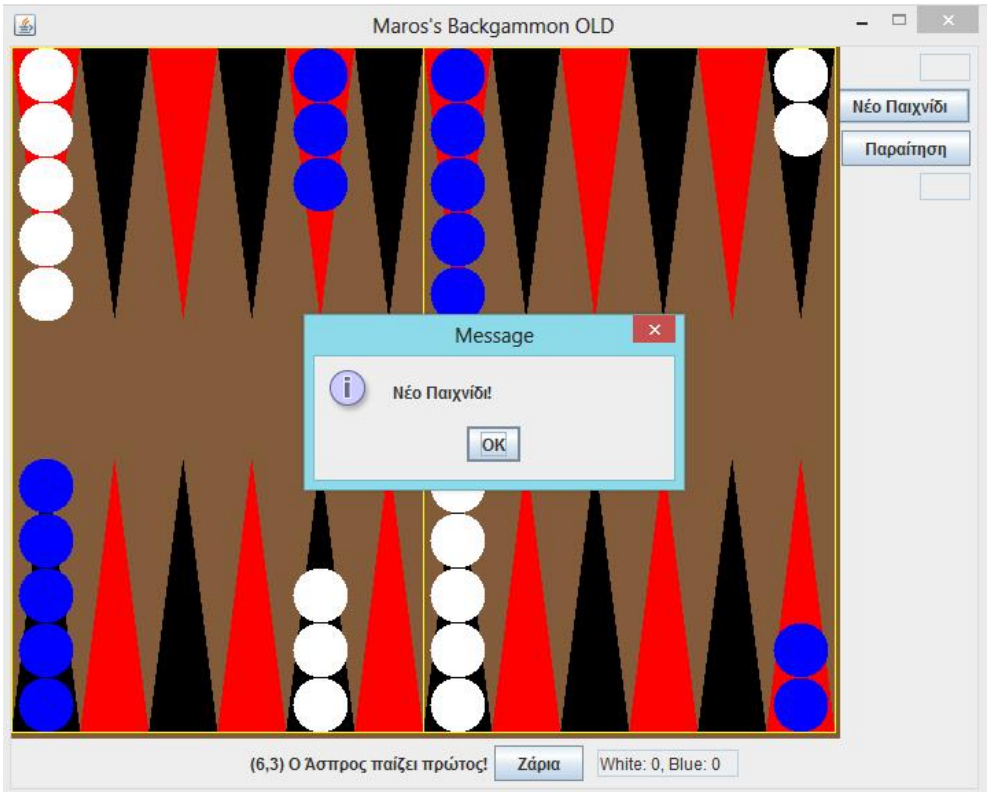
```
    public JButton resignation(){
        resign.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e){
            if (ui.getPlayer()==0){
                ui.scoreB++;
                if (ui.white[25]==0)
                    ui.scoreB++;
                JOptionPane.showMessageDialog(null, "Παραιτήθηκες! - Κέρδισε ο Μπλε \n Νέο Παιχνίδι!")
            }
            else{
                ui.scoreW++;
                if (ui.blue[25]==0)
                    ui.scoreW++;
                JOptionPane.showMessageDialog(null, "Παραιτήθηκες! - Κέρδισε ο Άσπρος \n Νέο Παιχνίδι!
            }
            initGame();
        }
    });
return resign;
}}
```

Εικόνα 49 – Ο κώδικας της resignation()

3.4.7 Η ΜΕΘΟΔΟΣ newGame()

Όταν ο παίκτης πατήσει το κουμπί «Νέο Παιχνίδι», καλείται η μέθοδος newGame() της κλάσης ButtonsActions_MRoussou_thesis.

Η μέθοδος εμφανίζει ένα παράθυρο μηνύματος που ενημερώνει τους παίκτες ότι ξεκινάει καινούριο παιχνίδι όπως φαίνεται στην Εικόνα 50 και επιπλέον καλεί την μέθοδο initGame() η οποία αρχικοποιεί τον πίνακα.



Εικόνα 50 – Καινούριο Παιχνίδι

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 51).

```
public JButton newGame(){
    newGame.setText("Νέο Παιχνίδι");
      newGame.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e){
                initGame();
                JOptionPane.showMessageDialog(null, "Νέο Παιχνίδι!");
            }
        });
    return newGame;
}
```

Εικόνα 51 – Ο κώδικας της newGame()

3.4.8 Η ΜΕΘΟΔΟΣ initGame()

Η μέθοδος initGame() καλείται στις περιπτώσεις που ξεκινάει καινούριο παιχνίδι.

Αρχικά τοποθετεί τα πούλια στον πίνακα σύμφωνα με το παιχνίδι «Πόρτες» και έπειτα ενεργοποιεί το κουμπί «Ζάρια».

Επίσης αρχικοποιεί όλα τα label και τα textField.

Εάν η μεταβλητή CUSTOMISED_START είναι ενεργοποιημένη από τον προγραμματιστή, δημιουργούνται κάποιες «ειδικές», προσαρμοσμένες περιπτώσεις για τον έλεγχο του προγράμματος. Στον κώδικα βλέπουμε δύο περιπτώσεις για δοκιμή.

Ο κώδικας της μεθόδου είναι ο παρακάτω: (Εικόνα 52). Εάν η μεταβλητή CUSTOMISED_START είναι true, ο πίνακας θα αρχικοποιηθεί με τους πίνακες οι οποίοι έχουν προσαρμοστεί παρακάτω.

```
public void initGame(){
    zaria.setEnabled(true);

    for (int i=0; i<26; i++){ //INIT PIECES
        if (i==5 || i==12){
            ui.white[i] = 5;
            ui.blue[i] = 5;
        }
        else if (i==7){
            ui.white[i] = 3;
            ui.blue[i] = 3;
        }
        else if (i==23){
            ui.white[i] = 2;
            ui.blue[i] = 2;
        }
        else{
            ui.white[i] = 0;
            ui.blue[i] = 0;
        }}

if (CUSTOMISED_START){ //{0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25
    //ui.white = new int[] {3, 2, 5, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,0,0};
    //ui.blue = new int[] {0, 2, 0, 2, 1, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0,0,0};
    ui.white = new int[] {5, 5, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,0,0};
    ui.blue = new int[] {5, 5, 4, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,0,0};
}

diceLab.setText("");
ui.repaint();
ui.firstPlayer();
ui.setCounter(0);
ui.bearTfield.setText("");
ui.bearTfield2.setText("");
}
```

Εικόνα 52 – Ο κώδικας της initGame()

ΚΕΦΑΛΑΙΟ 4ο: ΠΑΡΑΔΕΙΓΜΑΤΑ ΠΕΡΙΠΤΩΣΕΩΝ

4.1 ΧΤΥΠΗΜΑ ΕΝ ΩΡΑ ΜΑΖΕΜΑΤΟΣ

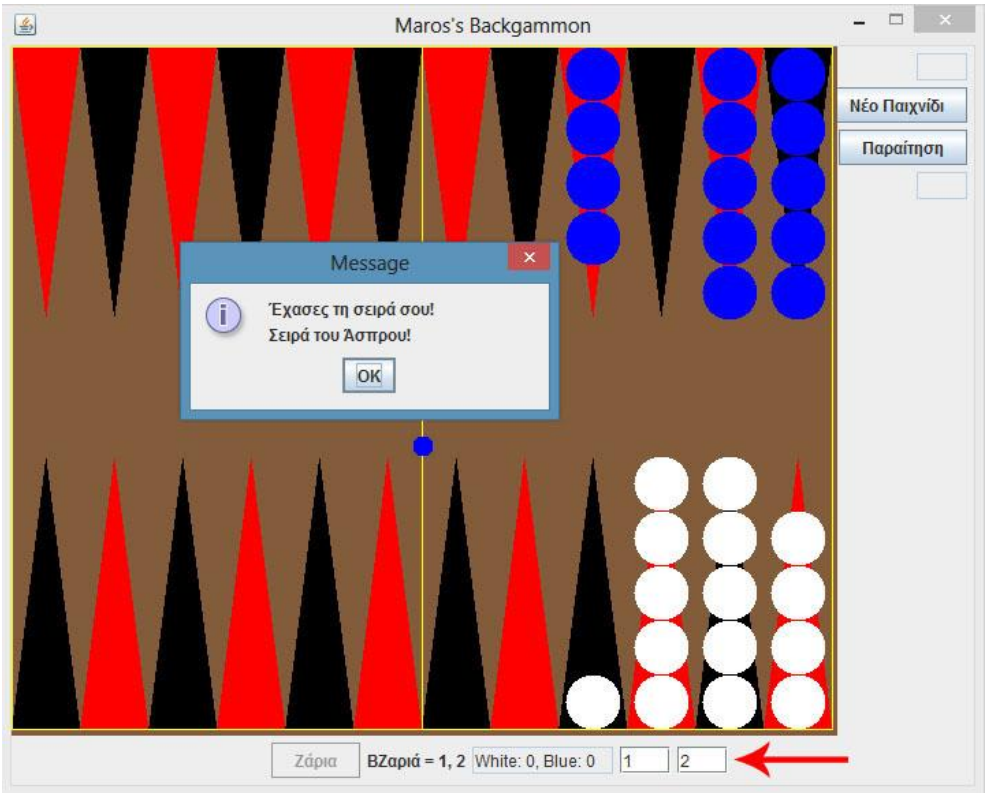
Παρακάτω βλέπουμε μια προσαρμοσμένη περίπτωση στην οποία οι παίκτες έχουν ξεκινήσει το μάζεμα και ο μπλε παίκτης έχει χτυπηθεί από τον άσπρο.

Σύμφωνα με τους δύο παρακάτω κανόνες του Ταβλιού: *«Εάν χτυπηθεί ένα πούλι ενώ έχει ξεκινήσει το μάζεμα, πρέπει πρώτα το πούλι να γυρίσει πίσω στην περιοχή μαζέματος ώστε να ξαναξεκινήσει το μάζεμα.»* και «Αν ο παίκτης έχει χτυπημένο πούλι και φέρει ζαριά (μονή ή διπλή) η οποία δε μπορεί να παιχτεί, χάνει τη σειρά του.» [15]

Έτσι στο παράδειγμά μας, εφόσον ο μπλε παίκτης μαζεύει αλλά είναι και «χτυπημένος», πρέπει να επιστρέψει πρώτα την περιοχή μαζέματός του και έπειτα να συνεχίσει να μαζεύει. Στο ρίξιμο των ζαριών έφερε 1 και 2 και στις αντίστοιχες στήλες (1 και 2) του άσπρου, υπάρχουν πάνω από ένα πούλια. Αυτό σημαίνει ότι ο παίκτης δεν μπορεί να μετακινήσει το χτυπημένο πούλι και χάνει τη σειρά του.

Συγκεκριμένα στον κώδικα καλείται η μέθοδος checkDice() η οποία μόλις βλέπει ότι η κίνηση δεν μπορεί να γίνει, καλεί την μέθοδο loseTurn() η οποία ακυρώνει την σειρά του τρέχων παίκτη.

[15] Rules of Backgammon, bkgm.com/rules.html

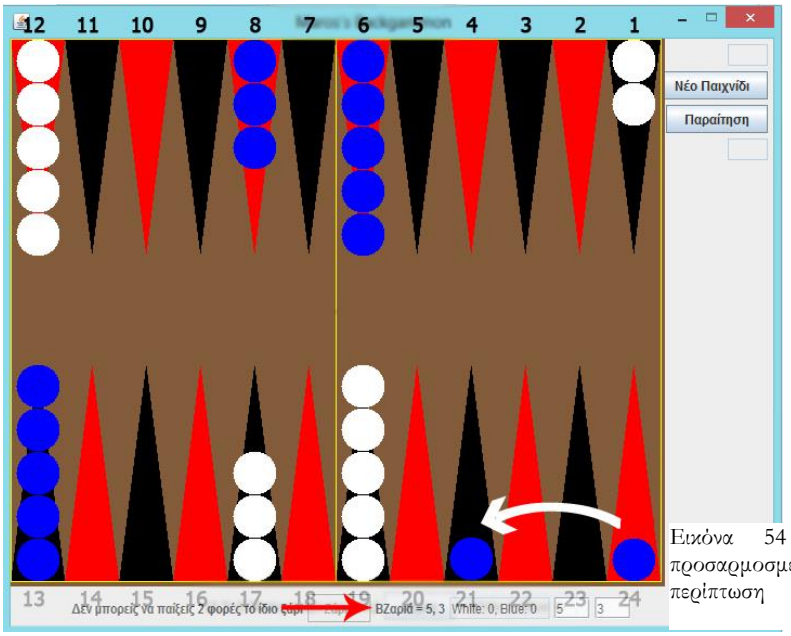


Εικόνα 53 – 1η προσαρμοσμένη περίπτωση

4.2 ΑΠΟΠΕΙΡΑ ΚΙΝΗΣΗΣ ΙΔΙΟΥ ΖΑΡΙΟΥ 2 ΦΟΡΕΣ

Στην παρακάτω προσαρμοσμένη περίπτωση ο μπλε παίκτης έχει φέρει τη ζαριά 5,3 και κινεί ένα πούλι από την στήλη 24 στην θέση 21 μετακινώντας 3 θέσεις το πούλι του. Έπειτα επιχειρεί να μετακινήσει ακόμα ένα πούλι από την στήλη 24 στην θέση 21, δηλαδή προσπαθεί να παίξει δεύτερη φορά το ίδιο ζάρι. Αυτό δεν γίνεται και εμφανίζεται σε label το μήνυμα «Δεν μπορείς να παίξεις 2 φορές το ίδιο ζάρι». Αυτή η περίπτωση ισχύει μόνο για τις μονές ζαριές.

Συγκεκριμένα στον κώδικα καλείται η μέθοδος validateMovement() η οποία ελέγχει εάν οι παίκτες ακολουθούν τους κανόνες του παιχνιδιού «Πόρτες». Σε μια περίπτωση όπως στο παράδειγμά μας, ελέγχεται αν πάει κάποια κίνηση να γίνει δεύτερη φορά εξετάζοντας τα ζάρια. Εάν κάποιο ζάρι έχει απενεργοποιηθεί – δηλαδή έχει παιχτεί κίνηση ίδια με τον αριθμό του, τότε δεν μπορεί να ξαναπαιχτεί κίνηση ίδια με αυτό τον αριθμό. Εάν αποπειραθεί κίνηση, τότε η validateMovement() εμφανίζει μήνυμα σε label «Δεν μπορείς να παίξεις 2 φορές το ίδιο ζάρι» και ο παίκτης αναγκάζεται να κάνει διαφορετική κίνηση.

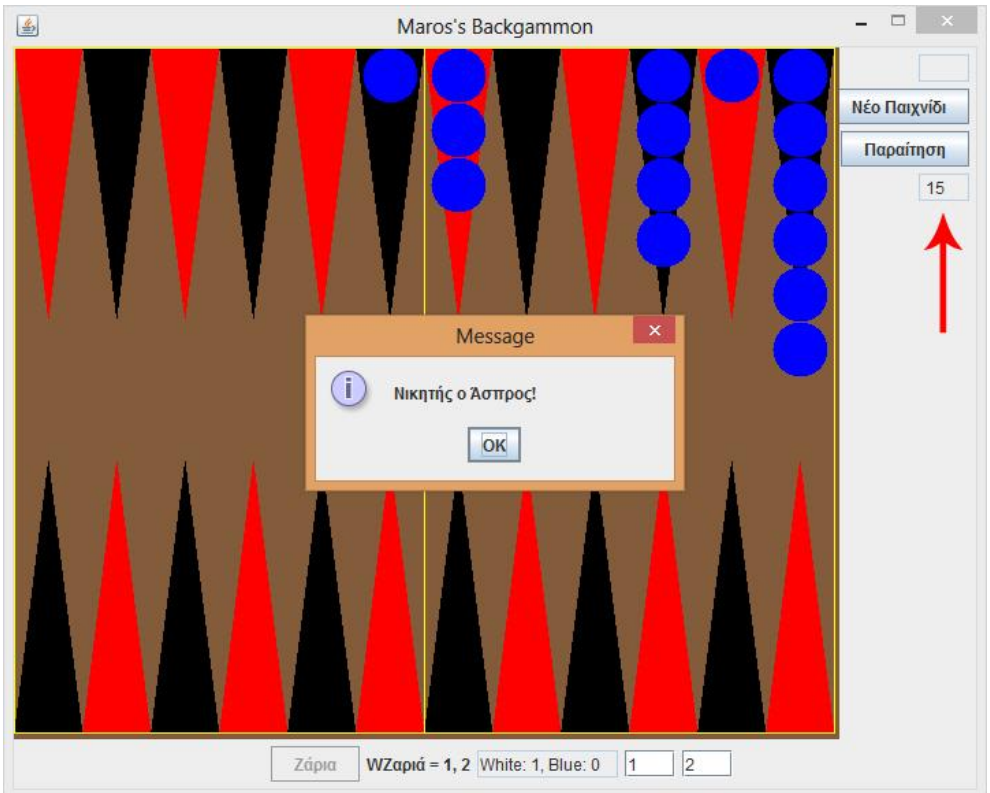


Εικόνα 54 – 2η προσαρμοσμένη περίπτωση

4.3 ΝΙΚΗ ΕΝΟΣ ΠΑΙΚΤΗ

Σύμφωνα με τον κανόνα του Ταβλιού *«Όταν ο παίκτης έχει μαζέψει και τα 15 πούλια, τότε κερδίζει (1 πόντος)»* [16], παρακάτω, βλέπουμε στην Εικόνα 55 την περίπτωση νίκης του άσπρου παίκτη, έχοντας μαζέψει 15 πούλια.

Συγκεκριμένα, στον κώδικα, μέσα στην μέθοδο updateTables() καλείται η win() η οποία ελέγχει εάν ο τρέχων παίκτης έχει 15 πούλια στην περιοχή μαζέματος. Εάν αυτό ισχύει τότε κερδίζει 1 πόντο (στην περίπτωση μονού παιχνιδιού), εμφανίζεται παράθυρο μηνύματος που ενημερώνει τους παίκτες για την νίκη του άσπρου (στο παράδειγμά μας) και αρχικοποιείται ο πίνακας σύμφωνα με το παιχνίδι «Πόρτες».



Εικόνα 55 – 3η προσαρμοσμένη περίπτωση

[16] Rules of Backgammon, bkgm.com/rules.html

ΚΕΦΑΛΑΙΟ 5ο: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

5.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

Η ανάπτυξη μιας εφαρμογής σε Java είναι μια αρκετά δύσκολη διαδικασία για την οποία απαιτούνται πολύ καλές γνώσεις της γλώσσας, συγκέντρωση και πολλές ιδέες.

Οι μηδαμινές γνώσεις μου στην Java με δυσκόλεψαν αρκετά, και χρειάστηκε να κάνω online μαθήματα και να προσπαθήσω να μάθω από την αρχή την γλώσσα ώστε να καταφέρω να ολοκληρώσω την εργασία.

Ολοκληρώνοντας την παρούσα πτυχιακή έχω πάρει αρκετές γνώσεις και καινούριους τρόπους σκέψης που με βοήθησαν στην πιο ολοκληρωμένη κατανόηση της γλώσσας Java και του αντικειμενοστραφή προγραμματισμού γενικότερα.

5.2 ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

Μια καλή εξέλιξη της εφαρμογής θα ήταν η βελτιστοποίησή της αρχικά ως προς το γραφικό περιβάλλον και έπειτα ως προς τη λειτουργία.

Συγκεκριμένα θα ήθελα να προσθέσω τον κανόνα τον οποίο δεν κατάφερα να υλοποιήσω και είναι ο εξής: *«Αν μόνο ένα ζάρι μπορεί να παιχτεί, παίζεται αυτό με το μεγαλύτερο αριθμό»*.[17]

Επιπρόσθετα θα ήθελα να υλοποιήσω κουμπί «Αναίρεση» το οποίο στο πάτημα θα ακυρώνει την τελευταία κίνηση του τρέχων παίκτη.

[17] Κανόνες για Τάβλι: gr.backgammoninfo.net/tavli

Επίσης θα μπορούσαν να γίνουν κάποιες τροποποιήσεις ώστε να είναι πιο ευανάγνωστος ο κώδικας, όπως η εφαρμογή του μοντέλου αρχιτεκτονικής MVC (Model View Controller) και η αντικατάσταση διπλότυπων σημείων του κώδικα με μεθόδους οι οποίες θα μπορούν να χρησιμοποιούνται περισσότερες από μια φορές.

Τέλος, η εφαρμογή μελλοντικά θα μπορούσε να εξελιχθεί σε online παιχνίδι σε πραγματικό χρόνο, έτσι ώστε να είναι δυνατό το παιχνίδι μεταξύ παικτών σε ξεχωριστό υπολογιστή και σε διαφορετική γεωγραφική περιοχή.

Μια καλή εξέλιξη του online παιχνιδιού θα ήταν η προσθήκη χρονομέτρου το οποίο θα μετράει κάποιον χρόνο ως το τέλος της κίνησης κάποιου παίκτη (πχ 90 δευτερόλεπτα). Εάν κάποιος παίκτης δεν έχει κάνει κάποια κίνηση μέσα στον προκαθορισμένο χρόνο θα χάνει τη σειρά του και θα συνεχίζει ο αντίπαλος παίκτης. Στην περίπτωση που για τρείς συνεχόμενες φορές ο παίκτης δεν έχει κάνει κάποια κίνηση, το παιχνίδι θα τερματίζεται.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Τάβλι, KCharitakis, τελευταία επεξεργασία 9/3/15, διαθέσιμο: el.wikipedia.org/wiki/Τάβλι
2. Ιστορία του Ταβλιού, διαθέσιμο: livetavli.gr/history.htm
3. Backgammon, τελευταία επεξεργασία 16/5/15, διαθέσιμο: wikipedia.org/wiki/Backgammon
4. Πώς παίζονται οι Πόρτες στο Τάβλι (backgammon), 22/8/14, διαθέσιμο: foxcasino.gr/portes-tavli-backgammon/
5. Rules of Backgammon, διαθέσιμο: bkgm.com/rules.html
6. Κανόνες του ταβλιού, C messier, τελευταία επεξεργασία 4/5/14, διαθέσιμο: el.wikipedia.opg/wiki/Κανόνες_του_ταβλιού
7. Πόρτες, διαθέσιμο, KCharitakis, τελευταία επεξεργασία 28/04/15: el.wikipedia.org/wiki/Πόρτες
8. Κανόνες για Τάβλι, διαθέσιμο: gr.backgammoninfo.net/tavli
9. AWT, Oracle and/or its affiliates, 2002, διαθέσιμο: docs.oracle.com/javase/1.5.0/docs/guide/awt/
10. Java Swings/AWT, Matiia, τελευταία επεξεργασία 21/4/15, διαθέσιμο: en.wikibooks.org/wiki/Java_Swings/AWT
11. Java Swing, Cory Janssen, διαθέσιμο: techopedia.com/definition/26102/java-swing
12. How to Set the Look and Feel, διαθέσιμο: docs.oracle.com/javase/tutorial/uiswing/lookandfeel
13. SWING – Controls, διαθέσιμο: tutorialspoint.com/swing/swing_controls
14. AWT vs. Swing, Josh Fletcher, διαθέσιμο: edn.embarcadero.com/article/26970

15. AWT vs. Swing, IDL Online Help, 16/6/05, διαθέσιμο: northstar-www.dartmouth.edu/doc/idl/html_6.2/AWT_vs._Swing.html
16. Objects, LearnJavaOnline.org, διαθέσιμο: learnjavaonline.org/en/Objects
17. Programming a Java Chess Engine, Jonathan Warkentin, 24/7/13 - 27/11/13, διαθέσιμο: youtube.com/user/jonathanwarkentin
18. Τι είναι το Eclipse, Ευθυμιάδης Αλέξανδρος, διαθέσιμο: openlab.teipir.gr/2012/04/install-eclipse/

ΠΑΡΑΡΤΗΜΑ: ΤΟ ΠΕΡΙΒΑΛΛΟΝ ΑΝΑΠΤΥΞΗΣ ECLIPSE

1. ΤΙ ΕΙΝΑΙ ΤΟ ECLIPSE [18]

Το Eclipse είναι ένα από τα δημοφιλέστερα IDEs (Integrated Development Environment ή στα ελληνικά: Ολοκληρωμένο Περιβάλλον Ανάπτυξης) που χρησιμοποιείται από χιλιάδες προγραμματιστές παγκοσμίως για τη συγγραφή και εκτέλεση κώδικα.

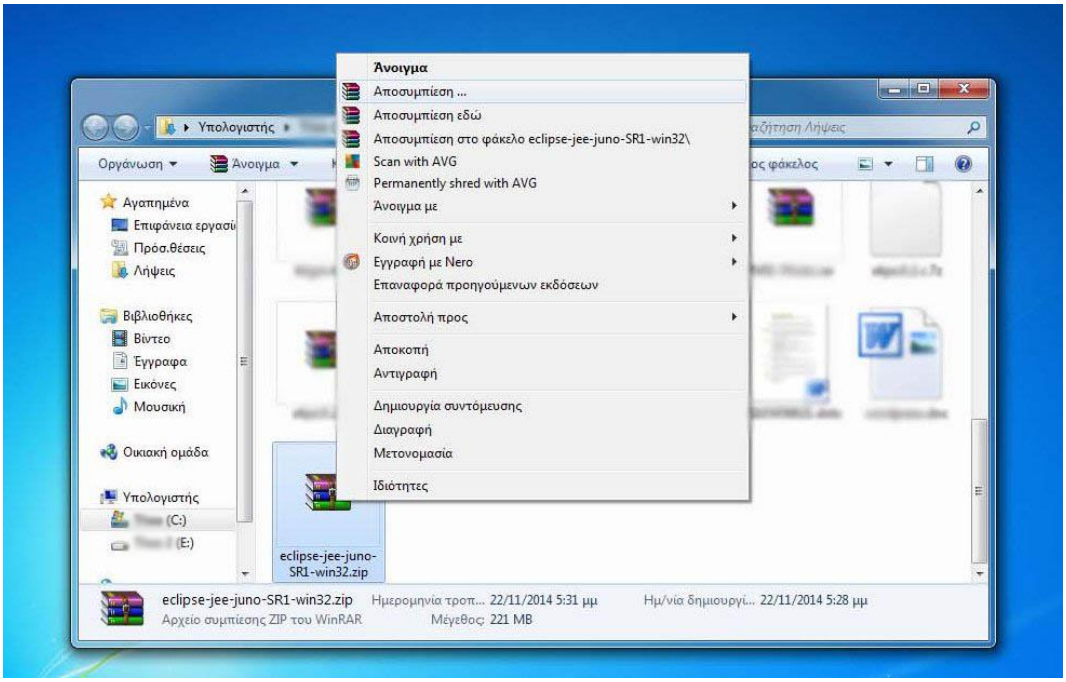
Η επιτυχία του οφείλεται στο λιτό του περιβάλλον το οποίο είναι φιλικό ακόμα και στον αρχάριο προγραμματιστή, επίσης είναι σχεδιασμένο να λειτουργεί σε πολλά λειτουργικά συστήματα (Linux, Mac, Windows), αλλά κυρίως επειδή υποστηρίζει πολλές γλώσσες προγραμματισμού, όπως Java, C, C++,Perl, PHP, Python δίνοντας έτσι στο προγραμματιστή το ίδιο περιβάλλον εργασίας για πολλές διαφορετικές γλώσσες.

Εικόνα 56 – Eclipse

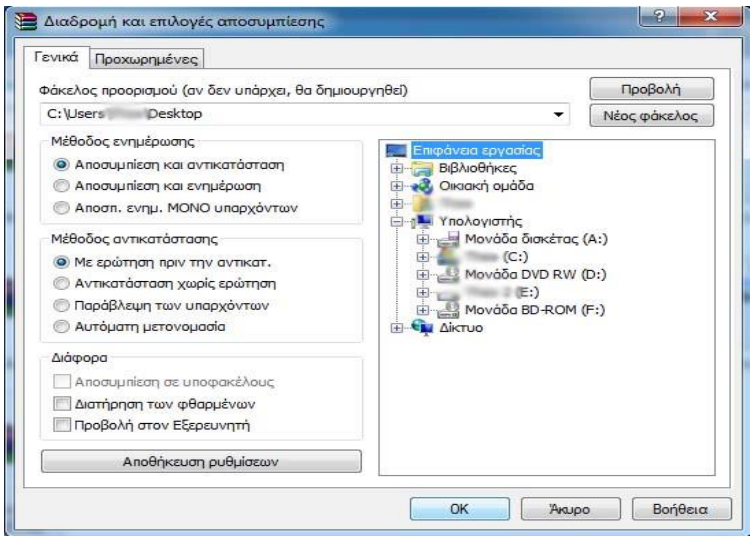
[18] Τι είναι το Eclipse; - Ευθυμιάδης Αλέξανδρος, openlab.teipir.gr/2012/04/install-eclipse/

2. ΕΓΚΑΤΑΣΤΑΣΗ ΣΕ WINDOWS 7

Για την εγκατάσταση του Eclipse σε Windows 7, θα πρέπει να κατεβάσουμε το αρχείο από την ιστοσελίδα του Eclipse: eclipse.org/downloads. Μόλις κατεβάσουμε το αρχείο, το αναζητούμε και το αποσυμπιέζουμε στην επιφάνεια εργασίας όπως φαίνεται στις Εικόνες 57 - 60.

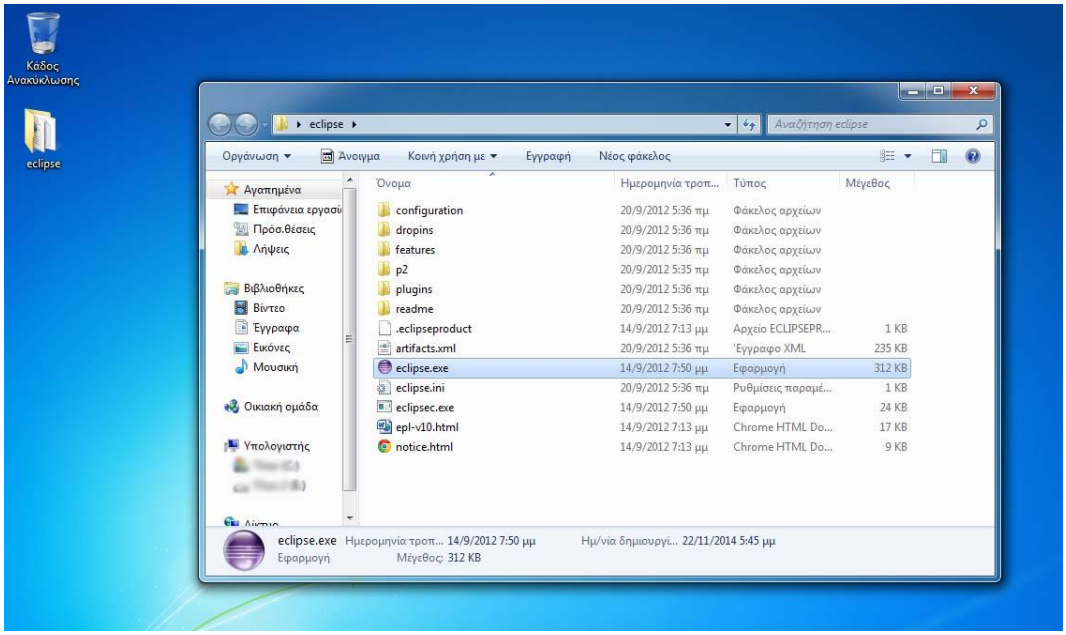


Εικόνα 57 – Εγκατάσταση σε Windows 7



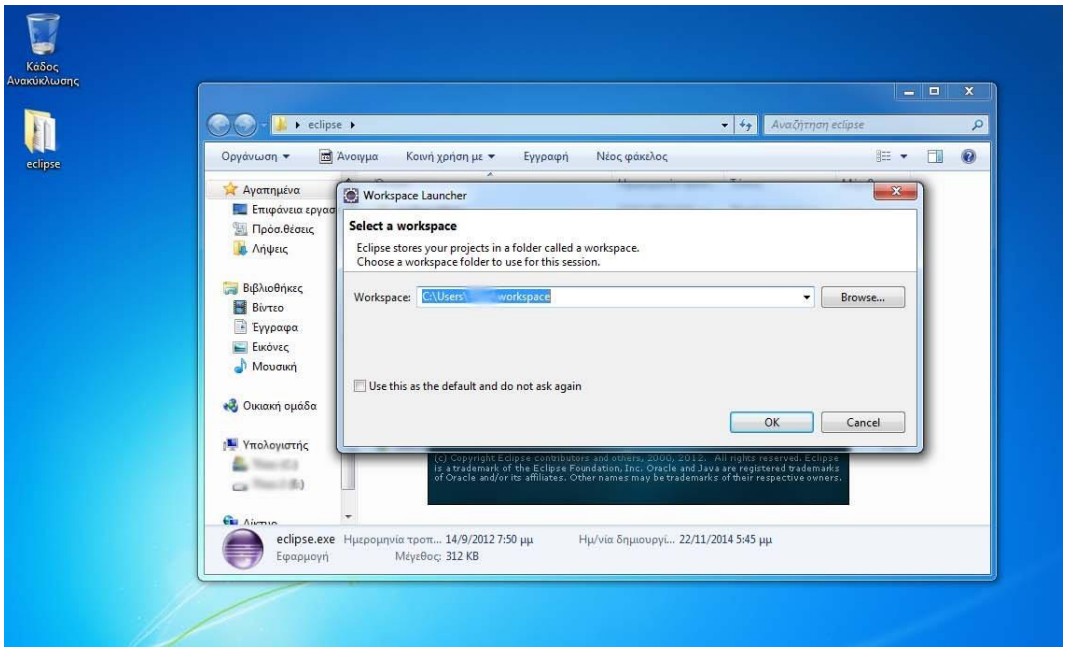
Εικόνα 58 – Εγκατάσταση σε Windows 7

Μόλις τελειώσει η αποσυμπίεση ανοίγουμε το φάκελο «eclipse» που βρίσκεται στην επιφάνεια εργασίας και εκτελούμε το αρχείο «eclipse.exe» (Εικόνα 59).



Εικόνα 59 – Εγκατάσταση σε Windows 7

Τέλος όταν ανοίγουμε για πρώτη φορά το Eclipse μας ζητάει να του δώσουμε το μονοπάτι που θα αποθηκεύονται τα προγράμματα που θα συγγράφουμε (Εικόνα 60).



Εικόνα 60 – Εγκατάσταση σε Windows 7