ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Στατιστική επεξεργασία συνθηκών τυχερών παιχνιδιών – Ανάπτυξη εφαρμογής

Του φοιτητή

Σκαρλάτου Ηλία

Αρ. Μητρώου: 07 / 3186

Επιβλέπων καθηγητής

Σφέτσος Παναγιώτης

Θεσσαλονίκη 2012

ΠΡΟΛΟΓΟΣ

Πολλές μελέτες έχουν αναπτυχθεί μέχρι στιγμής, και ακόμα περισσότερες θα ολοκληρωθούν στο μέλλον σχετικά με το πόσο τυχαία είναι τα νούμερα που κληρώνονται στα τυχερά παιγνίδια, όπως το «τζόκερ», το «λόττο» ή το «κίνο». Έχει ήδη κατατεθεί το ερώτημα σχετικά με το εάν τα νούμερα που παράγονται από κάποια γεννήτρια τυχαίων αριθμών ακολουθεί κάποια στατιστικά ανάλυση και αν τελικά είναι εφικτό μετά από αρκετή μαθηματική μελέτη να προβλεφθεί με επιτυχία κάποιος συνδυασμός που παράγεται από γεννήτρια τυχαίων αριθμών και θα αποτελεί την νικήτρια στήλη.

Η αγάπη μου για τα μαθηματικά, λόγω της ιδιότητάς μου του αποφοίτου της Σχολής Θετικών Σπουδών του Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης, του τμήματος Μαθηματικών, αλλά και η περιέργειά μου για το αν υπάρχει κάποιος τρόπος, χρησιμοποιώντας την θεωρία της στατιστικής, να προβλεφθεί με επιτυχία κάποιος συνδυασμός της νικήτριας στήλης σε κάποιο από τα τυχερά παιγνίδια του ΟΠΑΠ, και πιο συγκεκριμένα του «τζόκερ», μου έδωσαν το κίνητρο να ασχοληθώ με το συγκεκριμένο θέμα και να εκπονήσω την πτυχιακή μου εργασία μελετώντας το παραπάνω ερώτημα.

Σε αυτό το σημείο, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ.Σφέτσο, ο οποίος χωρίς δεύτερη σκέψη μου ανέθεσε με απευθείας ανάθεση το θέμα, το οποίο θα αναλυθεί παρακάτω. Χωρίς τις εύστοχες παρατηρήσεις του και τις κατευθυντήριες οδούς που μου υπέδειξε, πιστεύω δεν θα μπορούσα να έχω ολοκληρώσει με επιτυχία την πτυχιακή μου εργασία. Η βοήθεια που μου παρείχε, ήταν από τους κινητήριους μοχλούς της εργασίας.

ΠΕΡΙΛΗΨΗ

Στα κεφάλαια που ακολουθούν γίνεται αναφορά στο θεωρητικό υπόβαθρο πάνω στο οποίο στηρίχθηκε η εφαρμογή στο σύνολό της, όπως και σε κάποια κομμάτια του κώδικα της εφαρμογής έτσι, ώστε να αναλυθεί η αλγοριθμική που ακολουθήθηκε για την υλοποίηση κάποιων ιδιαίτερων και σημαντικών στοιχείων της. Ακόμα σχολιάζονται και επεξηγούνται εκτενέστερα κάποιες τεχνολογίες και βιβλιοθήκες ελεύθερου λογισμικού που χρησιμοποιήθηκαν για να προκύψει το βέλτιστο δυνατό αποτέλεσμα.

Μερικές εξ αυτών των βιβλιοθηκών τις οποίες παρουσιάζουμε στα επόμενα είναι η βιβλιοθήκη SQLite. Μια βιβλιοθήκη ελεύθερου για κάθε χρήση λογισμικού, η οποία έχει υλοποιηθεί σε γλώσσα αντικειμενοστραφούς προγραμματισμού C++ και η βασική της λειτουργία είναι η υλοποίηση μιας μηχανής βάσης δεδομένων, που είναι ανεξάρτητη από εξυπηρετητές και αναπτύσσεται απλά σε ένα αρχείο κατάληξης .dat.

Επίσης, γίνεται σχολιασμός όλων των προαπαιτούμενων για την αλληλεπίδραση με κάποια βάση δεδομένων. Εξηγείται η συνολική διαδικασία σύνδεσης μέσω ενός JDBC (Java DataBase Connectivity) driver σε κάποια βάση δεδομένων, συγκεκριμένα στην τοπική βάση που δημιουργείται κατά τη έναρξη της εφαρμογής, και η εκτέλεση βασικών ερωτημάτων, συνήθως παραμετρικών, για την εισαγωγή, διαγραφή και ενημέρωση εγγραφών στους πίνακες που έχουν δημιουργηθεί στην τοπική βάση.

Παρακάτω θα προσπαθήσουμε να σας εξηγήσουμε την σημαντικότητα χρήσης κάποιου είδους βάσης δεδομένων, και δη της SQLite database engine, για την αποθήκευση και διαχείριση των δεδομένων που χρησιμοποιούνται για την πραγματοποίηση των υπολογισμών που απαιτούνται για την στατιστική ανάλυση κάποιων στατιστικών όρων, όπως παρέχονται από την εφαρμογή.

Φυσικά, τα αποτελέσματα των στατιστικών μελετών βασίζονται κατά κύριο λόγο στην εγκυρότητα και το μέγεθος του δείγματος ενός πληθυσμού, ως προς το οποίο γίνεται η αντίστοιχη μελέτη. Για αυτό τον λόγο, δόθηκε ιδιαίτερη βαρύτητα στην πολλαπλότητα και λειτουργικότητα των επιλογών ενημέρωσης των δεδομένων που αποθηκεύονται στην τοπική βάση δεδομένων. Υπάρχει η δυνατότητα για τον χρήστη να ενημερώσει την τοπική του βάση φορτώνοντας κάποιο αρχείο κειμένου (.doc, .txt, .csv) με κάποιο συγκεκριμένο format, όπως αναφέρεται στην εφαρμογή, είτε μέσω κάποιου αρχείου excel επίσης με συγκεκριμένο format, όπως παρακινεί τον χρήστη η εφαρμογή, αλλά και με τον πιο εύχρηστο, άμεσο και ασφαλέστερο τρόπο, αρκεί να επιτρέπεται στο τερματικό η σύνδεσή του στο διαδίκτυο, που είναι η απευθείας ενημέρωση των δεδομένων από τα αρχεία excel που υπάρχουν στο επίσημο διαδικτυακό τόπο της ΟΠΑΠ Α.Ε. Οι διαδικασίες και η αλγοριθμική που χρησιμοποιούνται για την αποτελεσματική εκπόνηση όλων των παραπάνω εξηγούνται λεπτομερέστερα στα επόμενα κεφάλαια.

Βασικός σκοπός της εφαρμογής αυτής είναι να προσφέρει στον χρήστη, μέσω της στατιστικής ανάλυσης των κληρώσεων του παιγνιδιού τύχης, το «τζόκερ», τη δυνατότητα να αποκτήσει μια σφαιρική αντίληψη των αριθμών που κληρώνονται, ποιοι είναι αυτοί που έχουν την μεγαλύτερη συχνότητα εμφάνισης, καθυστέρησης, ποιοι κληρώνονται συχνότερα δύο φορές συνεχόμενα και πολλούς άλλους στατιστικούς όρους έτσι, ώστε τελικά να επιλέξει έναν συνδυασμό αριθμό, ο οποίος θα έχει μαθηματικά αποδεδειγμένα περισσότερες πιθανότητες εμφάνισης στην επόμενη κλήρωση. Σε αυτό το σημείο θα πρέπει να τονίσουμε ότι κάθε παιγνίδι τύχης δεν μπορεί να στηριχθεί μόνο σε μαθηματικά μοντέλα και υπολογισμούς, αφού ο παράγοντας τύχη είναι αυτός που παίζει τον πιο κρίσιμο ρόλο στην τελική επιτυχία ή αποτυχία. Συνεπώς, η εφαρμογή δεν εξασφαλίζει την επιτυχία στο συγκεκριμένο παιγνίδι τύχης, απλώς βοηθάει και κατευθύνει τον χρήστη στην επιλογή εκείνων των αριθμών που έχουν τις περισσότερες πιθανότητες να κληρωθούν.

Συνεπώς, βασική, αν όχι και η πιο σημαντική λειτουργία της εφαρμογής είναι ο τρόπος δημιουργίας μέσα από κάποιο υποσύνολο του συνόλου αριθμών {1, 2, 3, …, 45} κάποιων συνδυασμών που μπορεί ο χρήστης να παίξει και ίσως να κερδίσει. Η επιλογή αυτού του υποσυνόλου των αριθμών μπορεί να γίνει μέσω δύο διαφορετικών τρόπων. Αρχικά μπορούν να χρησιμοποιηθούν τα αποτελέσματα όπως προκύπτουν από την στατιστική ανάλυση των δεδομένων για κάποια συγκεκριμένη χρονική περίοδο. Ο χρήστης, αφού έχει εξετάσει τα αποτελέσματα αυτά και έχει κατασταλάξει σε συγκεκριμένους στατιστικούς όρους που πιστεύει ότι είναι σημαντικοί, μπορεί να τους ορίσει φίλτρα στην επιλογή των αριθμών του υποσυνόλου, μέσα από το οποίο θα επιλεγούν οι αριθμοί για την δημιουργία των συνδυασμών.

Ωστόσο, η εφαρμογή έχει αναπτυχθεί με τέτοιον τρόπο ώστε να επιτρέπει στον χρήστη να επιλέγει συγκεκριμένους αριθμούς, που θα ορίζουν την γεννήτρια των αριθμών των συνδυασμών και να καθορίζει τα όρια μέσα στα οποία θα πρέπει να κινηθεί το πλήθος των αριθμών ενός δεύτερου υποσυνόλου αριθμών, επίσης επιλογής του χρήστη, που συμμετέχουν στον τελικό συνδυασμό. Με αυτόν τον τρόπο ο χρήστης καθορίζει όλους εκείνους του συνδυασμούς που προκύπτουν μέσα από ένα υποσύνολο αριθμών του υπερσυνόλου {1, 2, 3, …, 45} και ικανοποιούν κάποιους περιορισμούς στο πλήθος των αριθμών αυτών στον τελικό συνδυασμό. Αυτό το υποσύνολο ονομάζεται βασικός όρος και εκτενέστερη αναφορά και επεξήγηση αυτού γίνεται στο τέλος του κειμένου, στον οδηγό της εφαρμογής που έχει συγγραφεί για να καθοδηγήσει τον χρήστη στα πρώτα του βήματα στην εφαρμογή.

ABSTRACT

The following chapters refer to the theoretical underpinning on which the application was based on the whole as well as on some parts of the application code so as to analyze the algorithmic that was followed for the implementation of some particular and major details. Furthermore we comment and explain extensively some technologies and free software libraries that were used for the best result.

Some of these libraries that we present in the following chapters are the SQLite library. A library of free all-purpose software, which is implemented in object-oriented programming language C++ and its main function is to implement a database engine that is independent of servers and is simply developed in a suffix file .dat.

Also, there is a commentary of all prerequisites for the interaction with a database. It is explained the overall process of connecting through a JDBC(Java Database Connectivity) driver in a database, namely on the locally generated at the start of the implementation and the enforcement of basic questions, usually parametric, to insert, delete and update records in tables that were created in the local base.

Bellow we will try to explain the significance of using some kind of database, particularly the SQLite database engine, for storing and managing the data that is used for the implementation of the required calculations, for the statistical analysis of some statistical terms, as they are provided by the application.

Of course, the results of statistical studies are based primarily on the validity and size of a population sample, on which is made the corresponding study. For this reason, special attention was given on the multiplicity and functionality of options that update the database that are stored in the local database. It is possible for the user to inform his local database by loading a text file(.doc, .txt, .csv)with a specific format, as is indicated in the application or via an excel file, also with a specific format, as the application prompts the user, but also in the most convenient, direct and safest way as long as it is allowed the connection of the terminal to the internet, which is the direct update of data from excel files that are available on the official website of OPAP SA. The procedures and the algorithmic that are used for the efficient development of the above are explained in detail in the following chapters.

The main purpose of this application is to provide the user through the statistical analysis of the lottery of the lucky game, the “joker”, the opportunity to gain a global understanding of the numbers that are drawn, which are those who have the greatest frequency, delay, which are those who are often drawn twice in succession and many other statistical terms so as to finally choose a combination number that will mathematically proven have more chances to the next draw. At this point we should emphasize that any lucky game cannot rely solely on

mathematical models and calculations, since luck is the factor that plays the most crucial role at the final success or failure. Hence, the application does not ensure the success in this game of luck, but just helps and guides the user in selecting those numbers that are most likely to be drawn.

Hence, basic, if not the most important function of the application is how to create through a subset of a set of numbers {1, 2, 3,…., 45} of some combinations that the user could play and maybe win. The choice of this subset of numbers can be done through two different ways. Initially, the results can be used, as they are derived from the statistical analysis of data for a specific period of time. The user, after examining these results and has …in particular statistic terms that believes they are important, he can set them filters to the choice of numbers of subset, through which the numbers will be selected for the creation of combinations.

However, the application has been developed in such a way that allows the user to choose specific numbers, which will define the generator of combination of numbers and determine the limits within the crowd of numbers of another subset of numbers, also user ´s choice should be, who participate in the final combination. In this way the user defines all combinations of those arising through a subset of the superset of numbers {1, 2, 3, …, 45} and satisfy some restrictions on the multitude of the final combination' s numbers. This subset is referred as basic term and extensive reporting and explanation of this is texting in the end of the document, at the manual of the application that has been written to guide the user through his first steps in implementation.

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο εικόνων

ΕΙΣΑΓΩΓΗ

Η εκπόνηση της εργασίας στηρίζεται σε δύο βασικούς πυλώνες, αφενός σε αυτόν της θεωρητικής ανάλυσης της στατιστικής θεωρίας, σε συνδυασμό με την θεωρητική μελέτη των αποτελεσμάτων των στατιστικών όρων. Αφετέρου στο κατασκευαστικό μέρος της εφαρμογής, το οποίο αποτελεί το εργαλείο εκείνο με το οποίο μπορεί ο χρήστης να εκμεταλλευτεί στο έπακρο όλες τις δυνατότητες μιας γλώσσας προγραμματισμού, όπως είναι η Java της εταιρίας Oracle, σε συνδυασμό με τα συμπεράσματα που μπορούν να προκύψουν από την θεωρητική μελέτη των στατιστικών δεδομένων που παράγονται από τις προηγούμενες κληρώσεις του τυχερού παιγνιδιού και να καταλήξει σε συγκεκριμένους συνδυασμούς αριθμών, αυξάνοντας έτσι τις πιθανότητές του να προβλέψει την νικήτρια στήλη του παιγνιδιού τύχης, όπως το «τζόκερ».

Ο κεντρικός άξονας της θεωρητικής ανάλυσης της στατιστικής θεωρίας στηρίζεται σε μεγάλο βαθμό σε δύο κύρια στατιστικά μεγέθη, τα οποία χρησιμοποιούνται στην στατιστική μελέτη ποσοτικών μεγεθών, όπως είναι οι συνδυασμοί αριθμών. Αυτά τα στατιστικά μεγέθη είναι η μέση τιμή και η τυπική απόκλιση, τα οποία θα αναλυθούν λεπτομερέστερα παρακάτω.

Με την βοήθεια των παραπάνω μεγεθών γίνεται μία μελέτη εκ βαθέων σχετικά με την συχνότητα εμφάνισης και καθυστέρησης των αριθμών στις κληρώσεις. Θεωρούμε ως δεδομένο ότι οι αριθμοί αυτοί ακολουθούν την κανονική κατανομή, οπότε χρησιμοποιώντας την μέση τιμή και την τυπική απόκλιση υπάρχει η δυνατότητα να προκύψουν ασφαλή συμπεράσματα, με κάποιο μικρό περιθώριο λάθους, για την συχνότητα εμφάνισης και καθυστέρησης των αριθμών.

Ωστόσο, η μελέτη δεν έχει σταματήσει μόνο σε αυτούς τους δύο στατιστικούς όρους, την συχνότητα εμφάνισης και καθυστέρησης των αριθμών που μπορούν να χρησιμοποιηθούν στη δημιουργία στήλης μιας κλήρωσης. Έχουν εμπλουτιστεί και με επιπλέον στατιστικούς όρους, οι οποίοι δίνουν μία πιο σφαιρική και ολοκληρωμένη στατιστική ανάλυση της τυχαιότητας εμφάνισης των αριθμών αυτών στις στήλες των κληρώσεων. Όλα τα παραπάνω θα αναλυθούν εκτενέστερα παρακάτω.

Ο δεύτερος ακρογωνιαίος λίθος στην υλοποίηση της πτυχιακής μου ήταν το κατασκευαστικό μέρος της εφαρμογής που αποτελεί και το βασικό μέρος της εργασίας αυτής. Η επιθυμία μου να διευρύνω τις γνώσεις μου και τις εμπειρίες μου πάνω στο τμήμα της επιστήμης των υπολογιστών, όπως είναι η ανάλυση και ανάπτυξη λογισμικού με ώθησαν στο να επιλέξω το συγκεκριμένο θέμα για να εκπονήσω την πτυχιακή μου εργασία. Το θεωρητικό υπόβαθρο που αποκόμισα από το τμήμα Πληροφορικής του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης κατά την διάρκεια των σπουδών μου, μου παρείχε όλα

εκείνα τα εργαλεία που με βοήθησαν για να ξεκινήσω και να ολοκληρώσω την εργασία αυτή.

Η αλγοριθμική αποτέλεσε έναν από τους κυριότερες άξονες στην ολοκλήρωση του κατασκευαστικού μέρους της πτυχιακής μου εργασίας. Ο συνδυασμός μαθηματικών μαζί με ανάπτυξη λογισμικού μπορεί να στεφθεί με επιτυχία μόνο εάν η αλγοριθμική που θα ακολουθηθεί στο κατασκευαστικό μέρος είναι προσεγμένη και πληροί κάποιους από τους βασικότερους κανόνες της αλγοριθμικής θεωρίας. Παρακάτω θα αναλυθούν όλες οι τεχνικές αλγοριθμικής που χρησιμοποιήθηκαν έτσι, ώστε να ολοκληρωθεί η εργασία.

Στην ανάλυση των απαιτήσεων που ορίστηκαν από τον επιβλέποντα καθηγητή περιλαμβάνεται και η διαχείριση αρχείων για την αποθήκευση των αποτελεσμάτων της στατιστικής μελέτης που διενεργείται στην εφαρμογή. Το είδος και ο τρόπος χρήσης των αρχείων αυτών από την εφαρμογή ποικίλει και θα καθοριστούν πλήρως παρακάτω.

Ένα από τα βασικότερα προβλήματα που έχρηζαν ιδιαίτερης προσοχής και ανάλυσης ήταν η διαχείριση μεγάλου όγκου δεδομένων που έχει συσσωρευτεί μετά από τόσα χρόνια λειτουργίας του συγκεκριμένου παιγνιδιού του ΟΠΑΠ, πάνω στο οποίο έχει γίνει η μελέτη και έχει υλοποιηθεί η εφαρμογή. Όλες οι στήλες των κληρώσεων από 16/11/1997 μέχρι και σήμερα μπορούν να χρησιμοποιηθούν ως δεδομένα προς στατιστική ανάλυση. Όπως γίνεται αντιληπτό, ήταν απαραίτητο να οργανωθεί έτσι το σύστημα, ώστε η διαχείριση όλων αυτών των δεδομένων να μην επιβαρύνει την άρτια λειτουργία της εφαρμογής, χωρίς καθυστερήσεις και λάθη. Για να ικανοποιηθεί η συγκεκριμένη ανάγκη αποφασίστηκε να χρησιμοποιηθεί μια βάση δεδομένων, στην οποία θα αποθηκεύονται όλα τα δεδομένα που χρησιμοποιούνται στην στατιστική ανάλυση. Ωστόσο προαπαιτούμενο στην υλοποίηση της εφαρμογής αποτελεί η ανεξαρτησία της εφαρμογής από servers βάσεων δεδομένων.

Για να υλοποιηθεί η παραπάνω απαίτηση, χρησιμοποιήθηκε ένα jar αρχείο, το οποίο μετατρέπει ένα αρχείο .db σε μια τοπική βάση. Το jar αυτό αρχείο, το οποίο ονομάζεται Sqlite είναι ανοικτού κώδικα, αποτελεί την πιο διαδεδομένη μηχανή SQL βάσης δεδομένων, που χρησιμοποιούνται σε προγράμματα κυρίως ανοικτού κώδικα. Ο μικρός χρόνος απόκρισης των διάφορων SQL commands από και προς την μηχανή αυτή, καθώς επίσης και η χρήση της χωρίς περαιτέρω ρυθμίσεις καθιστούν την χρήση της συγκεκριμένης βιβλιοθήκης ένα από τα σημαντικότερα εργαλεία διαχείρισης μεγάλου όγκου δεδομένων. Ακόμα, το μικρό μέγεθος του αρχείου που δημιουργείται για να αποτελέσει την βάση δεδομένων είναι πολύ μικρό, με αποτέλεσμα η συγκεκριμένη βιβλιοθήκη να αποτελεί αναπόσπαστο κομμάτι ανάπτυξης λογισμικού και εφαρμογών για smart phones και tablets.

Ένα από τα στοιχεία του κατασκευαστικού μέρους της πτυχιακής μου εργασίας, το οποίο είναι πολύ σημαντικό και λειτουργικό για τον χρήστη είναι η δυνατότητα ενημέρωσης των δεδομένων των στηλών των κληρώσεων με πολλούς

διαφορετικούς τρόπους. Υπάρχει η δυνατότητα για τον χρήστη να συνδεθεί, αν είναι διαθέσιμη η συγκεκριμένη λειτουργία στο τερματικό, που εκτελείται η εφαρμογή, με την επίσημη ιστοσελίδα του ΟΠΑΠ και να διαβάσει όλα τα αρχεία excel με τις στήλες των κληρώσεων από την 16/11/1997 μέχρι και σήμερα. Έτσι, πατώντας ένα κουμπί, θα μπορεί ο χρήστης να έχει ενημερωμένη την βάση δεδομένων του και να προκύπτουν πιο ασφαλή συμπεράσματα.

Ωστόσο, η δυνατότητα χρήσης αρχείων excel και text για την ενημέρωση της βάσης δεδομένων επιτρέπει στον χρήστη να ενημερώσει την τοπική του βάση με οποιεσδήποτε στήλες επιθυμεί και να προκύψει στατιστική μελέτη σύμφωνα με τα δικά του δεδομένα.

Μία επιπλέον δυνατότητα που προσφέρεται στον χρήστη είναι να μπορέσει να βρει όλες τις δυνατές στήλες που μπορούν να δημιουργηθούν με βάσει των αποτελεσμάτων της στατιστικής ανάλυσης που έχει προηγηθεί για κάποια χρονική περίοδο που έχει διαλέξει ο χρήστης. Συνεπώς, αφού έχει μελετήσει ο χρήστης τα στατιστικά δεδομένα που προσφέρονται από την εφαρμογή για κάποια συγκεκριμένη χρονική περίοδο, του προσφέρεται η επιλογή να βρει όλους τους δυνατούς συνδυασμούς που μπορούν να δημιουργηθούν από τους αριθμούς που ικανοποιούν τους συγκεκριμένους στατιστικούς όρους που χρησιμοποιεί ο χρήστης.

Με το προηγούμενο, επιτρέπεται στον χρήστη να συνδυάσει την στατιστική ανάλυση που έχει προκύψει με βάσει συγκεκριμένους στατιστικούς όρους και να φιλτράρει τις στήλες. Έτσι, θα καταλήξει σε ορισμένες μόνο στήλες, των οποίων οι αριθμοί θα πρέπει να ικανοποιούν τους στατιστικούς όρους που επέλεξε ο χρήστης.

Ακόμα, σε αυτή την επιλογή της εφαρμογής δίνεται η δυνατότητα στον χρήστη να αποθηκεύσει τους αριθμούς που έχουν προκύψει από τους στατιστικούς όρους που έχει επιλέξει για κάποια χρονική περίοδο σε κάποιο αρχείο text ή excel έτσι, ώστε να τους χρησιμοποιήσει ως βασική στήλη.

Ο όρος «βασική στήλη» αναφέρεται σε μία ομάδα αριθμών από την οποία ο χρήστης μπορεί να επιλέξει πόσοι αριθμοί θα πρέπει να βρίσκονται στη τελική στήλη, ώστε να επιλεγεί ως πιθανός συνδυασμός. Συνεπώς, η βασική στήλη χωρίζεται σε τρεις βασικούς τομείς, το όνομα της ομάδας στην οποία ανήκει η στήλη, στο πλήθος των αριθμών από την στήλη που επιθυμεί ο χρήστης να ανήκουν στον συνδυασμό έτσι, ώστε να είναι αποδεκτή η συγκεκριμένη στήλη. Συνεπώς, δίνοντας τα όρια αυτά η εφαρμογή διαλέγει μόνο τους συνδυασμούς αριθμών, οι οποίοι βρίσκονται σε αυτά τα όρια που έχουν οριστεί στην βασική στήλη.

Υπάρχει η δυνατότητα , ο χρήστης να ορίζει περισσότερες από μία βασικές στήλες, οι οποίες μπορούν να ανήκουν στην ίδια ή σε διαφορετικές ομάδες. Κάθε μία βασική στήλη οριοθετεί το πλήθος των αριθμών που θέλουμε να εμφανίζονται

στον συνδυασμό. Επιπλέον, εκτός της βασικής στήλης, έχει οριστεί ακόμα μία οντότητα, η «ομάδα». Η ομάδα ορίζεται και αυτή από ένα εύρος αριθμών, το οποίο καθορίζει τα όρια του πλήθους των βασικών στηλών που θέλουμε να ικανοποιούνται έτσι, ώστε να είναι αποδεκτή η συγκεκριμένη βασική στήλη.

Με αυτόν τον τρόπο, επιτρέπεται στον χρήστη να ορίσει τους αριθμούς μέσα από τους οποίους θα δημιουργηθούν οι συνδυασμοί. Έτσι, υπάρχει καλύτερη οργάνωση της τεράστιας πληροφορίας που μπορεί να προσφέρει η δημιουργία συνδυασμών στήλης κλήρωσης χρησιμοποιώντας είτε κάποιους συγκεκριμένους αριθμούς, είτε όλους μαζί τους αριθμούς.

Ένα μεγάλο μέρος της εφαρμογής στηρίζεται στους παραπάνω όρους, την «βασική στήλη» και την «ομάδα». Συνεπώς, ήταν απαραίτητο να δίνονται όλες εκείνες οι επιλογές στον χρήστη, ώστε να διαχειριστεί σωστά και με μεγάλη λειτουργικότητα αυτούς τους όρους. Οπότε, υπάρχει η δυνατότητα ο χρήστης να ανοίξει συγκεκριμένο αρχείο text ή excel, μέσα από το οποίο θα διαβάζονται όλες οι βασικές στήλες ή ομάδες. Επίσης, υπάρχει η ευχέρεια ο χρήστης να δημιουργήσει χειροκίνητα κάποια συγκεκριμένη βασική στήλη ή ομάδα, να την επεξεργαστεί και να την καταχωρήσει στη λίστα που βρίσκονται και οι υπόλοιπες βασικές στήλες ή ομάδες αντίστοιχα, ανάλογα τι επέλεξε ο χρήστης να διαχειριστεί.

Ακόμα, μπορεί ο χρήστης να αποθηκεύσει σε κάποιο συγκεκριμένο αρχείο text ή excel τις βασικές στήλες και ομάδες έτσι, ώστε αν το επιθυμεί να είναι επαναχρησιμοποιούμενες οι προηγούμενες σε κάποιες άλλες χρονικές στιγμές, στις οποίες ο χρήστης εκτελεί την συγκεκριμένη λειτουργία.

Ως γεννήτρια τυχαίων αριθμών έχουν δημιουργηθεί δύο επιλογές, χρησιμοποιώντας και τους 45 αριθμούς, αλλά και κάποιους συγκεκριμένους αριθμούς, τους οποίους μπορεί να διαλέξει ο χρήστης μέσα από την κατάλληλη φόρμα, που έχει δημιουργηθεί για την αντίστοιχη διαδικασία. Όπως γίνεται κατανοητό, για να προκύψουν πιο ασφαλή συμπεράσματα κατά την δημιουργία των στηλών, είναι απαραίτητο να γίνονται σωστές επιλογές των ορίων που έχουν οριστεί στις βασικές στήλες και στις ομάδες. Ωστόσο, το πλήθος των διάφορων συνδυασμών που θα δημιουργηθούν από την επιλεγμένη γεννήτρια αριθμών αυξάνεται εκθετικά με το αντίστοιχο πλήθος αριθμών που βρίσκονται στις βασικές στήλες, καθώς επίσης και το πλήθος των αριθμών που έχουν επιλεγεί από το χρήστη να είναι στην γεννήτρια τυχαίων αριθμών.

Συμπερασματικά, σύμφωνα με τα προαναφερθέντα σχετικά με τις βασικές στήλες και ομάδες, η διαχείριση των βασικών στηλών και ομάδων είναι ένα πολύ δυνατό εργαλείο στην άμεση δημιουργία στηλών που μπορούν να χρησιμοποιηθούν ως τελική στήλη στο παιγνίδι τύχης, το «τζόκερ». Παρόλα αυτά δεν θα πρέπει να παραβλέπεται το ενδεχόμενο να οδηγηθεί ο χρήστης σε πολύ λάθος συμπεράσματα, αν πρώτα δεν έχει κάνει την ανάλυση των στατιστικών δεδομένων που έχουν βρεθεί για κάποια συγκεκριμένη χρονική περίοδο, που επέλεξε ο

χρήστης και δεν μελετήσει προσεχτικά τα συμπεράσματα που μπορούν να προκύψουν από την αντίστοιχη στατιστική ανάλυση των δεδομένων.

ΚΕΦΑΛΑΙΟ 1: Στοιχεία Στατιστικής Ανάλυσης και Αλγοριθμικής.

ΕΙΣΑΓΩΓΗ

Μέρος της πτυχιακής εργασίας που παρουσιάζεται αποτελεί το θεωρητικό υπόβαθρο που ακολουθείται σε όλα τα στάδια ανάπτυξης της εφαρμογής. Η άκυρη και λανθασμένη προσέγγιση στα θεωρητικά προαπαιτούμενα της εφαρμογής θα κατέληγε σε απροσδόκητα αποτελέσματα, παρέχοντας ελλιπή και ανακριβή πληροφορία κατάλληλη για χρήση.

Λόγω του υψηλού βαθμού σημαντικότητας που διακατέχεται η συγκεκριμένη εφαρμογή σχετικά με την στατιστική ανάλυση, ένα αρκετά μεγάλο χρονικό διάστημα της διάρκειας εκπόνησης της εργασίας αφιερώθηκε στην ανάλυση, μελέτη και υλοποίηση όλων εκείνων των απαραίτητων για την εγκυρότητα της εφαρμογής, στατιστικά δεδομένα.

ΥΠΟΚΕΦΑΛΑΙΟ 1.1: Συχνότητα εμφάνισης και καθυστέρησης.

Σε αυτό το σημείο οφείλουμε να τονίσουμε ότι για κάθε είδους μεταβλητή που μελετάμε στατιστικά, αυτή διαχωρίζεται σε δύο πολύ βασικές υπό κατηγορίες. Την συχνότητα εμφάνισης των τιμών της μεταβλητής και την συχνότητα καθυστέρησης των τιμών της μεταβλητής.

Αυτός ο διαχωρισμός γίνεται για να έχει ο χρήστης μια πιο ολοκληρωμένη εικόνα της συμπεριφοράς των αριθμών που μπορεί να υπάρξουν σε κάποια κλήρωση. Συνεπώς, υπολογίζοντας κάποια από τα βασικά στατιστικά μεγέθη, που αναφέρονται σε ποσοτική, διακριτή μεταβλητή, όπως είναι ο μέσος όρος και τυπική απόκλιση, υπάρχει μεγάλη πιθανότητα να καταλήξουμε στα επιθυμητά αποτελέσματα. Τα αποτελέσματα αυτά δεν μπορεί να είναι τίποτα άλλο από την επαλήθευση της πρόβλεψης της κλήρωσης στην οποία καταλήγουμε χρησιμοποιώντας την εφαρμογή. Προφανώς, ότι αναφέρεται για την μία κατηγορία της μεταβλητής, όπως π.χ. για την συχνότητα εμφάνισης του συγκεκριμένου αριθμού στην κλήρωση, αναφέρεται και για την άλλη κατηγορία.

ΥΠΟΚΕΦΑΛΑΙΟ 1.2: Μέσος όρος και τυπική απόκλιση.

Η Στατιστική είναι ο κλάδος των (εφαρμοσμένων) Μαθηματικών, ο οποίος βασίζεται σ’ ένα σύνολο αρχών και μεθοδολογιών για:

- τον σχεδιασμό της διαδικασίας συλλογής δεδομένων,
- τη συνοπτική και αποτελεσματική παρουσίαση των δεδομένων,
- την ανάλυση και εξαγωγή αντίστοιχων συμπερασμάτων.

Η στατιστική ανάλυση που πραγματοποιείται, στηρίζεται σε ένα σύνολο, του οποίου τα στοιχεία εξετάζονται ως προς ένα ή περισσότερα χαρακτηριστικά τους. Το σύνολο αυτό ονομάζεται «Πληθυσμός». Όπως γίνεται εύκολα αντιληπτό, ο πληθυσμός για την συγκεκριμένη εφαρμογή αποτελεί το σύνολο:

Για τους αριθμούς των κληρώσεων:

$$\{x \in \mathbb{N} \ / \ 1 \leq x \leq 45\} \tag{1.1}$$

Για τους αριθμούς των κληρώσεων:

$$\{x \in \mathbb{N} \ / \ 1 \leq x \leq 20\} \tag{1.2}$$

Το αντίστοιχο χαρακτηριστικό του πληθυσμού, ως προς το οποίο προκύπτουν στατιστικά συμπεράσματα ονομάζεται «Μεταβλητή». Στη εφαρμογή αυτή χρησιμοποιούνται πολλές μεταβλητές, μερικές εκ των οποίων είναι η συχνότητα εμφάνισης και καθυστέρησης εμφάνισης των αριθμών, η συχνότητα εμφάνισης και καθυστέρησης συγκεκριμένου πλήθους μονών αριθμών στις κληρώσεις, κ.ά. Διεξοδικότερα θα αναλυθούν παρακάτω όλες οι μεταβλητές του πληθυσμού, ως προς τις οποίες γίνεται στατιστική μελέτη.

Υπάρχουν δύο ειδών μεταβλητές, οι ποσοτικές ή κατηγορικές και οι ποσοτικές. Οι πρώτες αναφέρονται σε αυτές των οποίων οι τιμές δεν μπορούν να είναι αριθμοί. Αντίθετα οι ποσοτικές μεταβλητές είναι αυτές των οποίων οι τιμές είναι αριθμοί. Προφανώς, οι μεταβλητές στις οποίες θα αναφερθούμε παρακάτω είναι ποσοτικές. Ωστόσο, οι ποσοτικές μπορούν να κατηγοριοποιηθούν περισσότερο ως εξής: σε διακριτές και συνεχείς. Οι διακριτές ποσοτικές μεταβλητές ονομάζονται αυτές των οποίων οι τιμές παίρνουν μεμονωμένες (ή αλλιώς ακέραιες τιμές). Αντίθετα, συνεχείς μεταβλητές ονομάζονται αυτές που η τιμή τους μπορεί να ανήκει σε ένα διάστημα πραγματικών αριθμών (α, β).

Συχνότητα μιας συγκεκριμένη τιμής της μεταβλητής ονομάζεται ο αριθμός που αντιστοιχεί στο πλήθος των παρατηρήσεων των οποίων η τιμή είναι ίδια με την συγκεκριμένη τιμή της μεταβλητής. Αυτονόητο είναι ότι το άθροισμα όλων των συχνοτήτων για τις τιμές μιας μεταβλητής ισούται με μέγεθος του πληθυσμού.

Για την καλύτερη μελέτη και περιγραφή της στατιστικής ανάλυσης των μεταβλητών χρησιμοποιούνται κάποια αριθμητικά μεγέθη, τα οποία ονομάζονται «Μέτρα κατανομής». Χρησιμοποιώντας τα αριθμητικά μεγέθη αυτά μας προσφέρεται η δυνατότητα να περιγράψουμε σύντομα την κατανομή ενός συνόλου δεδομένων για κάποια συγκεκριμένη μεταβλητή.

Τα μέτρα μιας κατανομή είναι κυρίως δύο ειδών, τα «μέτρα θέσης», τα οποία δίνουν τη θέση του «κέντρου» των παρατηρήσεων στον οριζόντιο άξονα, και τα «μέτρα διασποράς» ή «μέτρα μεταβλητότητας», τα οποία δίνουν τη διασπορά των παρατηρήσεων, δηλαδή πόσο αυτές εκτείνονται γύρω από το «κέντρο» τους.

Τα μέτρα θέσης χρησιμοποιούνται για την περιγραφή της θέσης ενός συνόλου δεδομένων πάνω στον οριζόντιο άξονα και εκφράζουν την «κατά μέσο όρο» απόσταση των δεδομένων από την αρχή των αξόνων. Το μέτρο θέσης που θα χρησιμοποιήσουμε στην στατιστική μελέτη της εφαρμογής είναι η «μέση τιμή» ή «αριθμητικός μέσος». Σε μια κατανομή συχνοτήτων, όταν οι τιμές x_1, x_2, x_3, …, x_k της μεταβλητής X έχουν συχνότητες v_1, v_2, v_3, …, v_k αντίστοιχα, τότε η μέση τιμή της κατανομής ορίζεται από τη σχέση:

$$\bar{x} = \frac{x_1 v_1 + x_2 v_2 + x_3 v_3 + \dots + x_K v_K}{v_1 + v_2 + v_3 + \dots + v_k} = \frac{1}{v}\sum_{i=1}^{\kappa} x_i v_i \qquad (1.3)$$

Τα μέτρα διασποράς ή μεταβλητότητας εκφράζουν τις αποκλίσεις των τιμών μιας μεταβλητής γύρω από τα μέτρα κεντρικής τάσης. Τα σπουδαιότερα είναι το «εύρος», η «διακύμανση» και η «τυπική απόκλιση».

Το εύρος R ενός συνόλου παρατηρήσεων ορίζεται ως η διαφορά της ελάχιστης παρατήρησης από τη μέγιστη παρατήρηση, δηλαδή:

R = μεγαλύτερη παρατήρηση – μικρότερη παρατήρηση

Αν t_1 ,t_2 ,t_3 …,t_n είναι οι παρατηρήσεις μιας μεταβλητής X με μέση τιμή $\bar{x}$, τότε η «διακύμανση» ή «διασπορά» s^2 των παρατηρήσεων ορίζεται από τη σχέση

$$s^2 = \frac{1}{v}\sum_{i=1}^{k}(t_i - \bar{x})^2 = \frac{1}{k}\left\{\sum_{i=0}^{k} {t_i}^2 - \frac{\left(\sum_{i=1}^{k} t_i\right)^2}{k}\right\} \qquad (1.4)$$

Η «τυπική απόκλιση» s ενός συνόλου παρατηρήσεων ορίζεται ως η θετική τετραγωνική ρίζα της διακύμανσης, δηλαδή

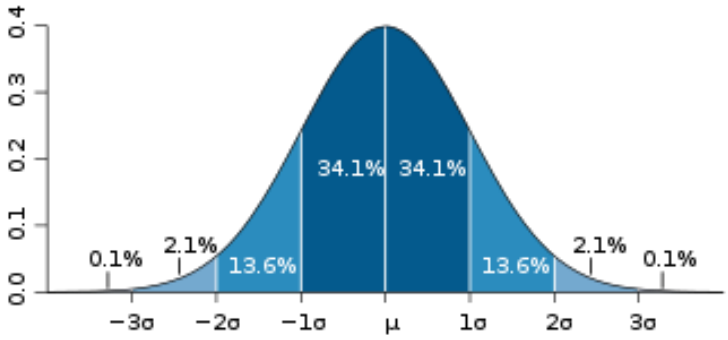
$$s = \sqrt{s^2} \qquad (1.5)$$

Η τυπική απόκλιση s πλεονεκτεί έναντι της διακύμανσης s^2 στο ότι εκφράζεται με την ίδια μονάδα μέτρησης με την οποία εκφράζονται και οι παρατηρήσεις.

Αν η καμπύλη συχνοτήτων, για το χαρακτηριστικό που εξετάζουμε, είναι κανονική ή περίπου κανονική, τότε η τυπική απόκλιση s έχει τις παρακάτω ιδιότητες:

- Το 68% περίπου των παρατηρήσεων βρίσκεται στο διάστημα $(\bar{x} - s, \bar{x} + s)$**.**
- Το 95% περίπου των παρατηρήσεων βρίσκεται στο διάστημα $(\bar{x} - 2s, \bar{x} + 2s)$**.**

- Το 99,7 % περίπου των παρατηρήσεων βρίσκεται στο διάστημα $(\bar{x} - 3s, \bar{x} + 3s)$.



Εικόνα 1. Διάγραμμα κανονικής κατανομής

Παρακάτω παραθέτουμε την μέθοδο του κώδικα που χρησιμοποιείται για τον υπολογισμό του μέσου όρου και της τυπικής απόκλισης κάποιας μεταβλητής του πληθυσμού των αριθμών.

```
public double[] getKanonikotitaOfNumber(int[] tableOfNumbers)
   {
      int[][] kanonikotitaResult;
      int max = tableOfNumbers[0];
      int min = tableOfNumbers[0];
      int Sx2v = 0;
      int Sxv = 0;
      int Sv = 0;
      double MOx = 0.0;
      double S2 = 0.0;
      double[] results = new double[2];
      try{
         for (int i = 0; i < tableOfNumbers.length; i++)
         {
            if (tableOfNumbers[i] > max)
               max = tableOfNumbers[i];
            else if (tableOfNumbers[i] < min)
               min = tableOfNumbers[i];
         }
         kanonikotitaResult = new int[max - min + 1][2];
         kanonikotitaResult[0][0] = min;
         kanonikotitaResult[0][1] = 0;
         for (int j = 1; j < kanonikotitaResult.length; j++)
         {
            kanonikotitaResult[j][0] = kanonikotitaResult[j - 1][0] + 1;
            kanonikotitaResult[j][1] = 0;
         }
         for (int k = 1; k <= tableOfNumbers.length; k++)
         {
            for (int i = 0 ; i < kanonikotitaResult.length; i++)
            {
               if (tableOfNumbers[k - 1] == kanonikotitaResult[i][0])
               {
```

```
                    kanonikotitaResult[i][1] += 1;
                    break;
                }
            }
        }
        for (int k = 0 ; k < kanonikotitaResult.length; k++)
        {
            Sv += kanonikotitaResult[k][1];
            Sxv += kanonikotitaResult[k][0] * kanonikotitaResult[k][1];
            Sx2v += (int)java.lang.Math.pow(kanonikotitaResult[k][0], 2) *
kanonikotitaResult[k][1];
        }
        if (Sv != 0)
        {
            MOx = (double)Sxv / (double)Sv;
            S2 = (Sx2v - (java.lang.Math.pow(Sxv, 2) / Sv))/ Sv;
        }
        double mesosOros = java.lang.Math.floor(MOx * 10 + 0.5d)/10;
        double typikiApoklisi = java.lang.Math.floor(java.lang.Math.sqrt(S2) * 100 +
0.5d)/100;
        results[0] = mesosOros; results[1] = typikiApoklisi;
      }
      catch(Exception e)
      {
          JOptionPane.showMessageDialog(this, String.valueOf(e.toString()),
"Προειδοποίηση!!", JOptionPane.ERROR_MESSAGE);

      }
      return results;
  }
```

Η παραπάνω συνάρτηση δέχεται ως παράμετρο έναν πίνακα με όλες τις τιμές της μεταβλητής, της οποίας μελετούμε το χαρακτηριστικό, όπως είναι η συχνότητα εμφάνισης των αριθμών στις κληρώσεις. Η συνάρτηση επιστρέφει έναν πίνακα δύο διαστάσεων, ο οποίος θα περιέχει τους αριθμούς του μέσου όρου και της τυπικής απόκλισης.

Αρχικά δημιουργείται υπολογίζεται ο μέγιστος και ο ελάχιστος αριθμός των τιμών της μεταβλητής, το χαρακτηριστικό της οποίας μελετάται. Έτσι, μπορούμε να αρχικοποιήσουμε έναν προσωρινό πίνακα, ο οποίος θα έχει μέγεθος (Μέγιστος Αριθμός – Ελάχιστος Αριθμός). Καταχωρούμε στο πρώτο κελί της πρώτης γραμμής, τον ελάχιστο αριθμό, ενώ στο δεύτερο κελί της ίδιας γραμμής θα καταχωρείται η αντίστοιχη τιμή της τυπικής απόκλισης των τιμών της μεταβλητή που μελετάται. Αρχικά θα είναι μηδέν. Στη συνέχεια σαρώνουμε τον πίνακα αυτόν και καταχωρείται στο πρώτο κελί κάθε μίας γραμμής, το νούμερο της προηγούμενης γραμμή αυξημένο κατά ένα. Αποτέλεσμα αυτού είναι να δημιουργηθεί ένας πίνακας με μέγεθος Μέγιστος Αριθμός – Ελάχιστος Αριθμός και με μία επαναληπτική μέθοδο, όπως είναι το for loop, γεμίζουμε τον πίνακα, του οποίου η πρώτη στήλη περιέχει όλους τους αριθμούς από την μικρότερη τιμή ως την μέγιστη, ενώ η δεύτερη στήλη περιέχει τον μέσο όρο των αριθμών αυτών.

Έπειτα, υπολογίζεται η διακύμανση και τυπική απόκλιση του πληθυσμού σύμφωνα με τους βασικούς τύπους που αναφέρθηκαν προηγουμένως.

Αυτά τα δύο αριθμητικά μεγέθη αποτελούν τα αποκλειστικά αριθμητικά μεγέθη, ως προς τα οποία θα γίνει η τελική στατιστική μελέτη και θα προκύψουν τα αντίστοιχα συμπεράσματα. Για αυτό, η παραπάνω συνάρτηση εκτελείται κάθε φορά που έχουμε έναν καινούριο στατιστικό όρο, τον οποίο θα μελετήσουμε.

ΥΠΟΚΕΦΑΛΑΙΟ 1.3 Ο στατιστικός όρος “Cold-Hot-Neutral” αριθμών.

Σύμφωνα με τα προηγούμενα, η συχνότητα εμφάνισης και καθυστέρησης αντίστοιχα των αριθμών που χρησιμοποιούνται στις κληρώσεις ακολουθεί την κανονική κατανομή. Στο διάγραμμα της κανονικής κατανομής (Εικόνα 1) φαίνονται τα αντίστοιχα ποσοστά των παρατηρήσεων των τιμών μιας μεταβλητής ως προς την οποία γίνεται η στατιστική μελέτη, ανά διάστημα γύρω από τον μέσο όρο.

Όπως γίνεται εύκολα αντιληπτό, όσο πιο κοντά στον μέσο όρο βρίσκεται η τιμή μιας παρατήρησης, τόσο πιο αναμενόμενο είναι να εμφανιστεί. Για αυτό τον λόγο, οι αριθμοί, των οποίων η συχνότητα εμφάνισης βρίσκονται στο διάστημα $(\bar{x} - s, \bar{x} + s)$ ονομάζονται “Neutral”, δηλαδή Φυσικοί.

Ωστόσο, η πιθανότητα εμφάνισης ενός αριθμού αυξάνεται όσο πιο δεξιά από τον μέσο όρο βρίσκεται η τιμή συχνότητας εμφάνισης αυτού του αριθμού. Συνεπώς, οι αριθμοί των οποίων οι συχνότητες βρίσκονται στο διάστημα $(\bar{x} + 2s)$ ονομάζονται “Warm”, δηλαδή Θερμοί. Αυτό σημαίνει, λόγω του ότι η συχνότητα εμφάνισης αυτών των αριθμών είναι τουλάχιστον κατά μία τυπική απόκλιση μεγαλύτερη από τον μέσο όρο εμφάνισης των αριθμών, εμφανίζονται συχνότερα από ότι ένας αριθμός που έχει χαρακτηριστεί ως “Neutral”. Αντίθετα, οι αριθμοί, των οποίων οι συχνότητες βρίσκονται στο διάστημα $(\bar{x} - 2s)$ ονομάζονται “Cool”, δηλαδή Χλιαροί. Αυτό σημαίνει, λόγω του ότι η συχνότητα εμφάνισης αυτών των αριθμών είναι τουλάχιστον κατά μία τυπική απόκλιση μικρότερη από τον μέσο όρο εμφάνισης των αριθμών, δεν εμφανίζεται τόσο συχνά όσο ένας αριθμός που έχει χαρακτηριστεί ως “Neutral”.

Τέλος, οι αριθμοί των οποίων οι τιμές των συχνοτήτων εμφάνισής τους βρίσκονται στο διάστημα $(\bar{x} + 3s)$ ονομάζονται “Hot”, δηλαδή Καυτοί. Αυτό σημαίνει, λόγω του ότι η συχνότητα εμφάνισης αυτών των αριθμών είναι τουλάχιστον κατά δύο φορές μεγαλύτερη από την τυπική απόκλιση εμφανίζονται συχνότερα από όλους τους υπόλοιπους αριθμούς. Αντίθετα, οι αριθμοί των οποίων οι συχνότητες εμφάνισης βρίσκονται στο διάστημα $(\bar{x} - 3s)$ ονομάζονται “Cold”, δηλαδή Παγωμένοι. Αυτό σημαίνει ότι η συχνότητα εμφάνισης αυτών των αριθμών είναι τόσο μικρή, που σχεδόν ποτέ δεν εμφανίζονται στις κληρώσεις.

Ο παραπάνω στατιστικός όρος είναι πάρα πολύ χρήσιμος, αφού τα συμπεράσματα που προκύπτουν από την ανάλυσή του στηρίζονται αποκλειστικά στο πλήθος των εμφανίσεων των αριθμών στις νικήτριες κληρώσεις. Συνεπώς, εύλογα κάποιος θα μπορούσε να σκεφτεί ότι μετά την ανάλυση των δεδομένων που έχουν συλλεχθεί από προηγούμενες κληρώσεις και τον χαρακτηρισμό των αριθμών αυτών ως “Cold”, “Cool”, “Neutral”, “Warm” ή “Hot”, θα διαλέξει μόνο τους αριθμούς που ανήκουν στην κατηγορία “Warm” ή “Hot”, αφού αυτοί είναι οι αριθμοί με τις μεγαλύτερες συχνότητες ή τουλάχιστον με συχνότητα μεγαλύτερη από τον μέσο όρο εμφάνισης όλων των αριθμών μαζί. Παρόλα αυτά, μια τέτοια σκέψη θα μπορούσε να οδηγήσει σε λανθασμένα αποτελέσματα και τελικά οι αριθμοί που θα διαλέγονταν από τον παίκτη να μην απέδιδαν αυτά που ήλπιζε και περίμενε. Μετά από αντίστοιχες μελέτες που έχουν πραγματοποιηθεί, έχει αποδειχθεί ότι η νικήτρια στήλη παρόμοιων με το «τζόκερ» παιγνίδια τύχης περιλαμβάνει συνδυασμών αριθμών που ανήκουν στις κατηγορίες “Neutral”, “Warm” και “Hot”.

Τα παραπάνω δεν θα πρέπει να θεωρηθούν από τον χρήστη βασικός κανόνας επιλογής αριθμών που θα μπορούσαν να αποτελέσουν την νικήτρια στήλη. Το αποτέλεσμα των κληρώσεων είναι αποτέλεσμα μιας διαδικασίας επιλογής τυχαίων αριθμών. Οπότε, δεν μπορεί να γίνει σίγουρη πρόβλεψη του αποτελέσματος με εξασφαλισμένο κέρδος.

Αλγοριθμικά ο χαρακτηρισμός κάποιου αριθμού ως “Cold”, “Cool”, “Neutral”, “Warm” ή “Hot”, στηρίζεται κατά κύριο λόγο στην εύρεση του μέσου όρου εμφάνισης των αριθμών, όπως και στην τυπική απόκλιση. Αρχικά, χρησιμοποιώντας μία ξεχωριστή διαδικασία την getFrequency(String firstDate, String secondDate) βρίσκουμε για κάθε έναν αριθμό από τους 45, που μπορούν να χρησιμοποιηθούν για την δημιουργία της νικήτριας στήλης, την συχνότητα εμφάνισής του. Στη συνέχεια, χρησιμοποιώντας την μέθοδο που υπολογίζει τον μέσο όρο και τυπική απόκλιση ενός συνόλου συχνοτήτων (βλέπε σελ.17), καταχωρούμε σε δύο μεταβλητές τον μέσο όρο και την τυπική απόκλιση. Οπότε, με κάποια επαναληπτική μέθοδο, όπως είναι η for loop, διατρέχουμε όλα τα στοιχεία του πίνακα, στον οποίο έχουν αποθηκευτεί οι αριθμοί με τις αντίστοιχες συχνότητές τους και κάνοντας τους κατάλληλους ελέγχους βρίσκουμε κάθε ένας αριθμός, με βάσει την συχνότητα εμφάνισής του σε ποια κατηγορία ανήκει.

Παρακάτω παραθέτουμε τον κώδικα υλοποίησης εύρεσης των συχνοτήτων εμφάνισης των αριθμών που μπορούν να συμμετέχουν στη δημιουργία της νικήτριας στήλης.

```
public int[] getFrequency(String firstDate, String secondDate)
   {
      int[] frequencyOfNumbers = new int[46];
      try {
           Class.forName("org.sqlite.JDBC");
         } catch (ClassNotFoundException ex) {

                  Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEV
                  ERE, null, ex);
         }
      Connection connection1 = null;
      Statement statement = null;
      try {
           connection1 =  DriverManager.getConnection("jdbc:sqlite:sample.db");
           statement = connection1.createStatement();
         } catch (SQLException ex) {

                  Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEV
                  ERE, null, ex);
         }


       try{
            for (int i = 0; i < frequencyOfNumbers.length; i++)
                frequencyOfNumbers[i] = 0;
             ResultSet freq = statement.executeQuery("SELECT n1 , n2 , n3 , n4 , n5 FROM
joker WHERE date between '" + firstDate + "' AND '" + secondDate + "';");
             while(freq.next())
               {
                for (int j = 1 ; j < 6; j++)
                    {
                       for (int k = 1; k <= 45; k++)
                       {
                          if (k == freq.getInt(j))
                             frequencyOfNumbers[k] += 1;
                          else continue;
                       }
                    }
                }
         }
         catch(SQLException ex) {
                   if (connection1 != null)
            try {
                 connection1.close();
                } catch (SQLException ex1) {

                  Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEV
                  ERE, null, ex1);
                }
         }
         finally
         {
            try {
                  if (!connection1.isClosed()) {
                     connection1.close();
```

```
                }
            }
        catch (SQLException ex) {

            Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEVERE,
            null, ex);
                }
    }
    return frequencyOfNumbers;
}
```

Όπως παρατηρείτε στον παραπάνω κομμάτι κώδικα που έχει χρησιμοποιηθεί για την εύρεση των συχνοτήτων εμφάνισης των αριθμών, υπάρχει σύνδεση του προγράμματος με κάποια βάση δεδομένων από την οποία διαβάζονται οι τιμές των αριθμών που έχουν ήδη κληρωθεί στο παρελθόν, σε κάποια χρονική περίοδο που έχει επιλεγεί από τον χρήστη. Λόγω του ότι ο όγκος των δεδομένων που μπορούν να χρησιμοποιηθούν από τις κληρώσεις όλων των ετών ύπαρξης του παιγνιδιού τύχης, του «τζόκερ» είναι πολύ μεγάλος, κρίθηκε απαραίτητο να χρησιμοποιηθεί μία τοπική βάση, στην οποία θα καταχωρούνται όλες οι κληρώσεις που έχουν πραγματοποιηθεί μέχρι σήμερα. Ωστόσο, περισσότερες λεπτομέρειες επί του θέματος θα αναφερθούν εκτενέστερα παρακάτω, στο κεφάλαιο που αναπτύσσονται θέματα σχετικά με το κατασκευαστικό μέρος της πτυχιακής εργασίας.

ΥΠΟΚΕΦΑΛΑΙΟ 1.4 Ο στατιστικός όρος Μονάδες αριθμών.

Με τον παραπάνω όρο αναφερόμαστε στους αριθμούς που χρησιμοποιούνται για την δημιουργία της νικήτριας στήλης, δηλαδή και για του 45 αριθμούς. Όπως έχει ήδη προαναφερθεί, η στατιστική ανάλυση βασίζεται σε δύο μεγάλες κατηγορίες, την συχνότητα εμφάνισης και καθυστέρησης αντίστοιχα. Έτσι, δίνεται στον χρήστη της εφαρμογής η δυνατότητα να μελετήσει με μεγαλύτερη ακρίβεια και σφαιρικότητα τα αποτελέσματα της στατιστικής ανάλυσης, όπως εξάγονται από τις λειτουργίες της εφαρμογής.

Η αλγοριθμική διαδικασία που ακολουθείται σε αυτό τον στατιστικό όρο έχει αναφερθεί παραπάνω, αφού με την μέθοδο getFrequency (βλέπε σελ. 21) υπολογίζονται και αποθηκεύονται σε έναν πίνακα οι συχνότητες εμφάνισης των αριθμών αυτών.

Ωστόσο, για την εύρεση της συχνότητας καθυστέρησης των αριθμών, χρησιμοποιείται μία διαφορετική διαδικασία, στην οποία διαβάζουμε από την τοπική βάση που έχουν καταχωρηθεί όλες οι νικήτριες κληρώσεις που έχουν ήδη πραγματοποιηθεί στο παιγνίδι τύχης, το «τζόκερ», κάθε έναν αριθμό από τους 45 που χρησιμοποιούνται στη δημιουργία των κληρώσεων και βρίσκουμε την συχνότητα της καθυστέρησης στην εμφάνισή τους, όπως επίσης και την τυπική απόκλιση για το σύνολο των καθυστερήσεων που παρουσιάζονται ανά αριθμό.

Παρακάτω παραθέτουμε τον κώδικα Java που έχει αναπτυχθεί για υλοποιήσει την στατιστική ανάλυση της καθυστέρησης στην εμφάνιση των αριθμών αυτών.

```
public java.util.ArrayList<String[]> getDelayOfNumbers(String firstDate, String
secondDate, int number)
   {
      int count = 0;
      java.util.ArrayList<String[]> delayOfNumbersTable = new
java.util.ArrayList<String[]>();
      String[] dateDelayNumberTable;
      try{
         try {
               Class.forName("org.sqlite.JDBC");
            } catch (ClassNotFoundException ex) {

         Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEVERE,
         null, ex);
            }
            Connection connection1 = null;
            Statement statement = null;
         try {
            connection1 =  DriverManager.getConnection("jdbc:sqlite:sample.db");
            statement = connection1.createStatement();
            } catch (SQLException ex) {

         Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEVERE,
         null, ex);
            }
          try{
               ResultSet DelayNumbers = statement.executeQuery("SELECT n1 , n2 , n3 ,
n4 , n5, date FROM joker WHERE date between '" + firstDate + "' AND '" + secondDate +
"'order by date desc;");
               String date = "";
               while(DelayNumbers.next())
                  {
                     date = DelayNumbers.getString("date");
                     if (number != DelayNumbers.getInt(1) && number !=
DelayNumbers.getInt(2) && number != DelayNumbers.getInt(3) &&
                           number != DelayNumbers.getInt(4) && number !=
DelayNumbers.getInt(5))
                              count += 1;
                     else
                     {
                        dateDelayNumberTable = new String[2];
                        dateDelayNumberTable[0] = DelayNumbers.getString("date");
                        dateDelayNumberTable[1] = String.valueOf(count);
                        delayOfNumbersTable.add(delayOfNumbersTable.size(),
dateDelayNumberTable);
                        count = 0;
                     }
                  }
                  if (count != 0)
                  {
                     dateDelayNumberTable = new String[2];
                     dateDelayNumberTable[0] = date;
```

```
                    dateDelayNumberTable[1] = String.valueOf(count);
                    delayOfNumbersTable.add(delayOfNumbersTable.size(),
dateDelayNumberTable);
                }
            }
            catch(SQLException ex)
            {
                if (connection1 != null)
                    try {
                        connection1.close();
                    } catch (SQLException ex1) {

Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEVERE, null,
ex1);
                    }
            }
            finally
            {
                try {
                    if (!connection1.isClosed()) {
                        connection1.close();
                    }

                    }
                catch (SQLException ex) {

          Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEVERE,
          null, ex);
                    }
            }

      }
      catch(Exception e)
      {
          JOptionPane.showMessageDialog(this, String.valueOf(e.toString()),
"Προειδοποίηση!!", JOptionPane.ERROR_MESSAGE);

      }
      return delayOfNumbersTable;
   }
```

ΥΠΟΚΕΦΑΛΑΙΟ 1.5 Ο στατιστικός όρος Επανάληψη Μονάδων αριθμών.

Με τον παραπάνω όρο αναφερόμαστε στην συχνότητα εμφάνισης ενός από τους αριθμούς αν κληρωθεί δύο συνεχόμενες φορές. Ουσιαστικά, για κάθε έναν αριθμό βρίσκουμε πόσες φορές έχει εμφανιστεί ο συγκεκριμένος αριθμός δύο συνεχόμενες φορές στον νικητήριο συνδυασμό της στήλης.

Στη συνέχεια, αφού αποθηκεύσουμε τα αποτελέσματα από την παραπάνω διαδικασία εύρεσης της συχνότητας επανάληψης δύο συνεχόμενων κληρώσεων ανά αριθμό, υπολογίζουμε τον μέσο όρο και την τυπική απόκλιση του συνόλου

των συχνοτήτων, όπως έχει δημιουργηθεί προηγουμένως (βλέπε σελ. 17) και τα εμφανίζονται στον χρήστη.

Παρακάτω παραθέτουμε το αντίστοιχο κομμάτι κώδικα Java, που υλοποιεί την εύρεση των συχνοτήτων καθυστέρησης στην εμφάνιση ανά αριθμό.

```
public int[] getEpanalipsisArithmwn(String _firstDate, String _secondDate)
   {
      java.util.ArrayList<Integer> firstDraw = new java.util.ArrayList<Integer>();
      java.util.ArrayList<Integer> secondDraw = new java.util.ArrayList<Integer>();
      int [] resultEpanalipsisCount = new int[46];
      Connection connection1 = null;
      Statement statement = null;
      for (int i = 0; i < resultEpanalipsisCount.length; i++)
         resultEpanalipsisCount[i] = 0;
      try{
         try {
             Class.forName("org.sqlite.JDBC");
         } catch (ClassNotFoundException ex) {

                  Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEV
                  ERE, null, ex);
         }
         try {
         connection1 =  DriverManager.getConnection("jdbc:sqlite:sample.db");
         statement = connection1.createStatement();
         } catch (SQLException ex) {

                  Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEV
                  ERE, null, ex);
         }
         try{
         ResultSet EpanalipsisArithmwn = statement.executeQuery("SELECT n1 , n2 , n3 ,
n4 , n5 FROM joker WHERE date between '" + _firstDate + "' AND '" + _secondDate + "'
order by date desc;");
         while(EpanalipsisArithmwn.next())
            {
            int number = 0;

               for (int index = 1; index < 6; index++)
               {
                  number = EpanalipsisArithmwn.getInt(index);
                  if (!firstDraw.isEmpty())
                  {

                     if (firstDraw.contains(number))
                     {
                        resultEpanalipsisCount[number]++;
                        firstDraw.remove((Integer)number);
                     }
                     else
                        secondDraw.add(secondDraw.size(), number);
                  }
                  else
```

```
                    secondDraw.add(secondDraw.size(), number);
                }
            firstDraw.clear();
            for (int item : secondDraw)
                {
                    firstDraw.add(firstDraw.size(), item);
                }
            secondDraw.clear();

            }
        }
        catch(SQLException ex)
        {
        if (connection1 != null)
            try {
                 connection1.close();
                } catch (SQLException ex1) {

        Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEVERE,
        null, ex1);
                }
        }
        finally
        {
            try {
                  if (!connection1.isClosed()) {
                      connection1.close();
                  }
                }
            catch (SQLException ex) {

        Logger.getLogger(AllStatisticsJFrameNew.class.getName()).log(Level.SEVERE,
        null, ex);
                }
        }
    }
    catch(Exception ex)
    {
        JOptionPane.showMessageDialog(this, String.valueOf(ex.toString()),
"Προειδοποίηση!!", JOptionPane.ERROR_MESSAGE);
    }
    return resultEpanalipsisCount;
  }
```

ΥΠΟΚΕΦΑΛΑΙΟ 1.6 Ο στατιστικός όρος Λήγοντες αριθμοί.

Στα πλαίσια της ανάπτυξης λογισμικού με χρήση αντικειμενοστραφούς γλώσσας προγραμματισμού, όπως είναι η Java, και για να υποστηρίζεται από το πρόγραμμα η τεχνική της ενθυλάκωσης, αναπτύχθηκε μία κλάση, η οποία δημιουργεί αντικείμενο για κάθε μία κλήρωση. Η κλάση αυτή ονομάζεται JokerNumbers.java, έχει ως δομητή μία συνάρτηση στην οποία αρχικοποιούνται οι

μεταβλητές που αντιστοιχούν στους αριθμούς της κλήρωσης και του αριθμού Joker. Επίσης, ταξινομούνται οι αριθμοί που βρίσκονται στον συνδυασμό της κλήρωσης.

Εκτός του δομητή, στην κλάση αυτή έχουν αναπτυχθεί ξεχωριστές συναρτήσεις, οι οποίες κάθε μία αναφέρεται σε συγκεκριμένο στατιστικό όρο. Οι συναρτήσεις αυτές έχουν βασικό στόχο για κάθε κλήρωση να υπολογίζεται το αποτέλεσμα του στατιστικού όρου.

Έτσι, για τον στατιστικό όρο «Λήγοντες» αριθμοί υπάρχει η μέθοδος getEndingNumber της κλάσης JokerNumbers.java, η οποία για κάθε μία κλήρωση υπολογίζει το πλήθος των αριθμών του συνδυασμού που ανήκουν στο σύνολο ληγόντων αριθμών που ζητείται παραμετρικά. Στη συνέχεια θα αναφέρουμε την μέθοδο αυτή έτσι, ώστε να γίνει σαφέστερη η λειτουργία της μεθόδου αυτής.

```
public ArrayList<Integer> getEndingNumber(int number)
   {
      int counter = 0;
      ArrayList<Integer> result = new ArrayList<Integer>();
      for (int i = 0; i < tempallNumbers.length; i++)
      {

         if (String.valueOf(tempallNumbers[i]).endsWith(String.valueOf(number)))
         {
            counter++;
            result.add(result.size(), tempallNumbers[i]);
         }
      }
      return result;
   }
```

Σχετικά με το κομμάτι κώδικα Java που εμφανίζεται παραπάνω θα πρέπει να αναφέρουμε ότι στην μεταβλητή tempallNumbers, έχουν αποθηκευτεί οι αριθμοί της τρέχουσας κλήρωσης, για την οποία γίνεται η κλήση της συνάρτησης. Συνεπώς, για κάθε έναν από τους αριθμούς αυτούς που βρίσκονται στη συγκεκριμένη κλήρωση υπολογίζουμε αν λήγει στον αριθμό που υπάρχει στην παράμετρο number. Αν ισχύει, τότε αποθηκεύεται σε μία δομή ArrayList, την μεταβλητή result, η οποία επιστρέφεται στο σημείο όπου κλήθηκε η συνάρτηση.

Οι υπόλοιποι στατιστικοί όροι δεν αναφέρονται μεμονωμένα στους αριθμούς που συμμετέχουν στη δημιουργία της νικήτριας στήλης, αλλά σε υποσύνολα αυτού του συνόλου των αριθμών. Συνεπώς, οι παρακάτω στατιστικοί όροι μπορούν να χρησιμοποιηθούν για την μελέτη κάποιας ομάδας αριθμών με το ίδιο χαρακτηριστικό στοιχείο, οπότε σε συνδυασμό με κάποιον άλλον στατιστικό όρο να βρούμε ένα σύνολο αριθμών που ικανοποιούν τα εξαγόμενα από την μελέτη αποτελέσματα.

Ο πρώτος από την ομάδα αυτή των στατιστικών όρων, που θα αναλύσουμε είναι ο στατιστικός όρος «Λήγοντες αριθμοί». Με αυτό τον όρο αναφερόμαστε στο σύνολο των αριθμών που λήγουν στον ίδιο αριθμό, έναν από τους αριθμούς 0,1, 2, …, 9. Συνεπώς, οι 45 αριθμοί χωρίζονται σε αυτούς που λήγουν σε 0, σε 1, σε 2, κ.ά. και η αντίστοιχη στατιστική ανάλυση γίνεται για κάθε σύνολο αριθμών ξεχωριστά. Αυτό έχει άμεσο επακόλουθο να υπολογίζεται ξεχωριστός μέσος όρος και τυπική απόκλιση ανά σύνολο.

Όπως συμβαίνει και με τους προηγούμενους στατιστικούς όρους, έτσι και εδώ η ανάλυση χωρίζεται σε δύο βασικές και κατηγορίες, την συχνότητα εμφάνισης και καθυστέρησης αριθμού στην νικήτρια στήλη, ο οποίος όμως, να ανήκει στο σύνολο των αριθμών που λήγουν στον ίδιο αριθμό. Επειδή ένας συνδυασμός της νικήτριας στήλης αποτελείται από 5 αριθμούς, υπάρχει η περίπτωση το πλήθος των αριθμών που ανήκουν στο ίδιο σύνολο ληγόντων αριθμών να είναι 0, 1, 2, 3, 4 ή 5. Για τον λόγο αυτό, η στατιστική ανάλυση γίνεται ανά σύνολο ληγόντων αριθμών και ανά πλήθος αυτών στην νικήτρια στήλη.

Για την εύρεση των συχνοτήτων καθυστέρησης στην εμφάνιση αριθμών που ανήκουν στο ίδιο σύνολο ληγόντων αριθμών έχει χρησιμοποιηθεί ξεχωριστή διαδικασία, κατά την οποία διαβάζονται όλες οι εγγραφές των κληρώσεων, που είναι καταχωρημένες στην τοπική βάση, και ανά σύνολο ληγόντων αριθμών βρίσκεται η συχνότητα καθυστέρησης στην εμφάνιση των αριθμών που ανήκουν στο συγκεκριμένο σύνολο. Έτσι γίνεται εφικτό η εύρεση του μέσου όρου και της τυπικής απόκλισης των συχνοτήτων καθυστέρησης στην εμφάνιση αριθμών.

Παρακάτω εμφανίζεται το κομμάτι κώδικα Java, που χρησιμοποιείται για την εύρεση των συχνοτήτων καθυστέρησης στην εμφάνιση αριθμών που ανήκουν σε κάποιο συγκεκριμένο σύνολο ληγόντων αριθμών.

```
public int[] getDelayCounterKlirwsis(int[] _numbers, int _count)
   {
      ArrayList<Integer> temp = new ArrayList<Integer>();
      int[] result;
      int delayCounter = 0;
         for (int i = 0; i < _numbers.length; i++)
         {
            if (_numbers[i] != _count)
               delayCounter++;
            else
            {
               temp.add(temp.size(), delayCounter);
               delayCounter = 0;
            }
         }
         if (delayCounter != 0)
            temp.add(temp.size(), delayCounter);
         result = new int[temp.size()];

         for (int i = 0; i < temp.size(); i++)
```

```
        {
            result[i] = temp.get(i);
        }
    return result;
}
```

Στην υπογραφή της παραπάνω συνάρτησης εμφανίζονται δύο παράμετροι, ένας πίνακας ακέραιων αριθμών και ένας ακέραιος αριθμός. Η δεύτερη παράμετρος, η _count, αναφέρεται στο πλήθος των αριθμών που ανήκουν στο συγκεκριμένο σύνολο ληγόντων αριθμών και εμφανίζονται στον τελικό συνδυασμό. Στην πρώτη παράμετρο αποθηκεύεται το πλήθος των αριθμών ανά κλήρωση, οι οποίοι ανήκουν στο συγκεκριμένο σύνολο ληγόντων αριθμών, για το οποίο γίνεται η στατιστική ανάλυση.

Στο σώμα της συνάρτησης υπολογίζονται και αποθηκεύονται στον πίνακα result το πλήθος των καθυστερήσεων στην εμφάνιση κάποιου αριθμού που ανήκει στο συγκεκριμένο σύνολο ληγόντων αριθμών και στο τέλος επιστρέφεται στο σημείο κλήσης της συνάρτησης ο πίνακας result.

ΥΠΟΚΕΦΑΛΑΙΟ 1.7 Ο στατιστικός όρος Ταυτάριθμοι αριθμοί.

Με τον όρο «Ταυτάριθμος» αναφερόμαστε στο μονοψήφιο αριθμό που προκύπτει αν προσθέσουμε τα ψηφία ενός αριθμού. Οπότε, το σύνολο των αριθμών που συμμετέχουν στην δημιουργία των νικητριών στηλών χωρίζονται σε υποσύνολα σύμφωνα με τον ταυτάριθμό τους. Έτσι, υπάρχει το υποσύνολο των αριθμών που έχουν ταυτάριθμο 1, αυτών που έχουν ταυτάριθμο 2, κ.ά.

Για κάθε υποσύνολο αριθμών που έχουν τον ίδιο ταυτάριθμο γίνεται ξεχωριστή στατιστική ανάλυση ως προς την συχνότητα εμφάνισης και καθυστέρησης των αριθμών αυτών.

Η αντίστοιχη μέθοδος της κλάσης JokerNumbers.java, η οποία υπολογίζει πόσοι από τους αριθμούς της κλήρωσης για την οποία γίνεται ο υπολογισμός του πλήθους των αριθμών του συνδυασμού που ανήκουν σε κάποιο συγκεκριμένο ταυτάριθμο είναι η getTautarithmoiNumber. Στη συνέχεια παραθέτουμε το κομμάτι του κώδικα Java που υλοποιεί τον αντίστοιχο υπολογισμό:

```
public ArrayList<Integer> getTautarithmoiNumber(int _number)
  {
    ArrayList<Integer> result = new ArrayList<Integer>();
    int size;
    int[] charactersOfNumber;
    int number = 0;
    int sum;
    for (int i = 0; i < tempallNumbers.length; i++)
    {
        number = tempallNumbers[i];
        do{
```

```
            sum = 0;
            size = String.valueOf(number).length();
            charactersOfNumber = new int[size];
            for (int j = 0; j < charactersOfNumber.length; j++)
               charactersOfNumber[j] = Integer.parseInt(String.valueOf(number).substring(j, j +
1));
            for (int m = 0; m < size; m++)
               sum += charactersOfNumber[m];
            number = sum;
            }while(sum > 9);

            if (sum == _number)
               result.add(tempallNumbers[i]);
         }
         return result;
      }
```

Στην μεταβλητή tempallNumbers έχουν αποθηκευτεί οι αριθμοί του συνδυασμού, για τον οποίο εκτελείται η συγκεκριμένη μέθοδος. Οπότε, για κάθε αριθμό του συνδυασμού αυτού υπολογίζεται το άθροισμα των ψηφίων του, μέχρι να γίνει το άθροισμα αυτό μονοψήφιο. Στη συνέχεια το άθροισμα αυτό ελέγχεται με τον αριθμό της παραμέτρου _number, ο οποίος καθορίζει τον ταυτάριθμο που θέλουμε. Έτσι, αν το αποτέλεσμα των προσθέσεων είναι ίσο με την παράμετρο _number, τότε αποθηκεύεται στην ArrayList result, η οποία στο τέλος θα περιλαμβάνει τους αριθμούς που έχουν ταυτάριθμο ίσο με την τιμή της παραμέτρου _number και επιστρέφεται στην κλήση της μεθόδου.

Όμως, όπως στον προηγούμενο στατιστικό όρο, σε έναν συνδυασμό πέντε αριθμών υπάρχει περίπτωση να μην υπάρχει αριθμός στη νικήτρια στήλη, ο οποίος να ανήκει στο συγκεκριμένο σύνολο ταυτάριθμων ή να υπάρχει μόνο ένας αριθμός στη νικήτρια στήλη ή δύο, κ.ά. Άρα στην αλγοριθμική επίλυση που χρησιμοποιείται για την υλοποίηση της στατιστικής μελέτης του συγκεκριμένου όρου λαμβάνεται υπόψη και το πλήθος των αριθμών που ανήκουν σε κάποιο σύνολο ταυτάριθμων και εμφανίζονται ανά κλήρωση.

Η συνάρτηση που χρησιμοποιείται για την εύρεση συχνότητας καθυστέρησης στην εμφάνιση αριθμού που ανήκει σε κάποιο σύνολο ταυτάριθμων είναι η getDelayCounterKlirwsis (βλέπε σελ. 28). Στην πρώτη παράμετρο έχουμε έναν πίνακα ακέραιων αριθμών, στον οποίο αποθηκεύονται το πλήθος των αριθμών που ανήκουν στο κάποιο σύνολο ταυτάριθμων και η δεύτερη παράμετρος αναφέρεται στο πλήθος ανά κλήρωση των αριθμών που ανήκουν στο συγκεκριμένο σύνολο ταυτάριθμων αριθμών.

ΥΠΟΚΕΦΑΛΑΙΟ 1.8 Ο στατιστικός όρος Δεκάδες αριθμοί.

Ο όρος «Δεκάδες» αριθμών αναφέρεται στην δεκάδα στην οποία ανήκει κάποιος συγκεκριμένος αριθμός. Συνεπώς, το σύνολο των αριθμών 1 – 9 ανήκει στην πρώτη δεκάδα, το σύνολο 10 – 19 ανήκει στην δεύτερη δεκάδα, το σύνολο 20 – 29 στην τρίτη, κ.ά.

Η μέθοδος της κλάσης JokerNumbers.java, η οποία χρησιμοποιείται για τον υπολογισμό του πλήθους των αριθμών ανά κλήρωση που ανήκουν σε συγκεκριμένο σύνολο δεκάδων αριθμών είναι η getStartingNumber. Το αντίστοιχο κομμάτι κώδικα Java είναι το παρακάτω:

```
public ArrayList<Integer> getStartingNumber(int number)
   {
      ArrayList<Integer> result = new ArrayList<Integer>();
      for (int i = 0; i < tempallNumbers.length; i++)
      {
         if (number == 1 && (tempallNumbers[i] > 0 && tempallNumbers[i] < 10))
            result.add(result.size(), tempallNumbers[i]);
         else
         if (number > 1 && String.valueOf(tempallNumbers[i]).length() > 1 &&
String.valueOf(tempallNumbers[i]).substring(0, 1).equals(String.valueOf(number - 1)))
            result.add(result.size(), tempallNumbers[i]);

      }
      return result;
   }
```

Όπως φαίνεται και στον κώδικα, για κάθε συνδυασμό αριθμών για τον οποίο εκτελείται η συγκεκριμένη μέθοδος, ελέγχονται όλοι οι αριθμοί του συνδυασμού, αν η δεκάδα στην οποία ανήκουν είναι ίση με την δεκάδα που είναι αποθηκευμένη στην μεταβλητή number. Αν ισχύει, τότε αποθηκεύεται στην δομή ArrayList result, ο αριθμός αυτός και στο τέλος επιστρέφεται στο σημείο που κλήθηκε η συνάρτηση το result.

Όπως συμβαίνει και στους προηγούμενους στατιστικούς όρους, η ανάλυση γίνεται τόσο για την συχνότητα εμφάνισης, τόσο και την συχνότητα καθυστέρησης στην εμφάνιση κάποιου αριθμού που ανήκει στη συγκεκριμένη δεκάδα. Επίσης, ανά δεκάδα γίνεται διαχωρισμός στο πλήθος των αριθμών που ανήκουν στη δεκάδα και εμφανίζονται στους συνδυασμούς των κληρώσεων. Η συνάρτηση που χρησιμοποιείται για την εύρεση της συχνότητας καθυστέρησης στην εμφάνιση αριθμού που ανήκει σε κάποιο σύνολο δεκάδων αριθμών είναι η getDelayCounterKlirwsis (βλέπε σελ. 28).

ΥΠΟΚΕΦΑΛΑΙΟ 1.9 Ο στατιστικός όρος Μονά – Ζυγά αριθμών.

Ο συγκεκριμένος στατιστικός όρος αναφέρεται στο πλήθος των αριθμών ενός συνδυασμού που μπορεί να είναι μονοί ή ζυγοί. Προφανώς, αν ένας συνδυασμός έχει δύο αριθμούς, οι οποίοι είναι μονοί, τότε οι άλλοι τρεις θα είναι ζυγοί. Για αυτό τον λόγο, ο υπολογισμός γίνεται μόνο για το πλήθος ανά κλήρωση που είναι μονοί αριθμοί και στη συνέχεια υπολογίζεται το πλήθος των ζυγών αριθμών αφαιρώντας από το πέντε το πλήθος των μονών αριθμών.

Η αντίστοιχη μέθοδος της κλάσης JokerNumbers.java, η οποία υπολογίζει το πλήθος των αριθμών ανά κλήρωση που είναι μονοί και πόσοι ζυγοί, είναι η getOddEvenNumber. Αρχικά θα παραθέσουμε το κομμάτι κώδικα java που χρησιμοποιείται για τον υπολογισμό και στη συνέχεια θα αναλυθούν κάποια βασικά μέρη της μεθόδου.

```
public int[] getOddEvenNumber()
   {
      int[] result = new int[2];
      int oddSum = 0;
      int evenSum = 0;
      for (int i = 0; i < tempallNumbers.length; i++)
      {
         if (tempallNumbers[i] % 2 == 0)
            oddSum += 1;
         else
            evenSum += 1;
      }
      result[0] = evenSum;
      result[1] = oddSum;
      return result;
   }
```

Στον πίνακα tempallNumebrs είναι καταχωρημένοι οι αριθμοί της κλήρωσης για την οποία γίνεται ο αντίστοιχος υπολογισμός μονών – ζυγών. Οπότε για κάθε στοιχείο του πίνακα βρίσκουμε το υπόλοιπο της διαίρεσης με το 2 αν είναι ίσο με 0, οπότε θα αυξηθεί ο αντίστοιχος μετρητής των ζυγών αριθμών, αντίθετα θα αυξηθεί ο αριθμός των μονών αριθμών. Οι δύο μετρητές αποθηκεύονται στον πίνακα result, ο οποίος επιστρέφεται στο σημείο κλήσης της μεθόδου.

Σε αυτόν τον στατιστικό όρο για την περίπτωση της συχνότητας εμφάνισης των μονών – ζυγών η ανάλυση δεν υπολογίζει τον μέσο όρο και την τυπική απόκλιση των αριθμών αυτών, παρά μόνο αναφέρεται η συχνότητα εμφάνισης των μονών – ζυγών.

Αντίθετα, στην στατιστική ανάλυση της συχνότητας της καθυστέρησης εμφάνισης μονών – ζυγών, υπολογίζεται και ο μέσος όρος μαζί με την τυπική απόκλιση τους.

ΥΠΟΚΕΦΑΛΑΙΟ 1.10 Ο στατιστικός όρος Αριστερά – Δεξιά αριθμών.

Χρησιμοποιώντας ο χρήστης τον παραπάνω στατιστικό όρο, του δίνεται η δυνατότητα να μελετήσει την στατιστική ανάλυση που πραγματοποιείται για την συχνότητα εμφάνισης και καθυστέρησης του πλήθους των αριθμών ανά κλήρωση που είναι αριστεροί και δεξιοί. Με τον όρο «Αριστερά» αναφερόμαστε στους αριθμούς των οποίων ο λήγοντας αριθμός ανήκει στο σύνολο {1, 2, 3, 4, 5}, ενώ δεξιοί είναι οι αριθμοί που ο λήγοντάς τους αριθμός ανήκει στο σύνολο {6, 7, 8, 9, 10}.

Για τον υπολογισμό και χαρακτηρισμό των αριθμών αυτών ως αριστερός ή δεξιός, έχει υλοποιηθεί στην κλάση JokerNumbers.java η μέθοδος getLeftRightNumber(), η οποία χρησιμοποιώντας την μέθοδο getEndingNumber (βλέπε σελ. 27) βρίσκει για κάθε αριθμό του τρέχοντος συνδυασμό, για τον οποίο υλοποιείται η μέθοδος getLeftRightNumber(), τον λήγοντα αριθμό του. Έτσι, ελέγχοντας τον αριθμό αυτό, αυξάνουμε τον αντίστοιχο μετρητή των αριστερών αριθμών ή των δεξιών αριθμών κατά 1. Η συνάρτηση αυτή επιστρέφει έναν πίνακα ακέραιων αριθμών, στον οποίο έχουν αποθηκευτεί οι μετρητές των αριστερών και δεξιών αριθμών που υπάρχουν ανά συνδυασμό κλήρωσης.

Παρακάτω παραθέτουμε το αντίστοιχο κομμάτι κώδικα Java, ο οποίος υλοποιεί την αλγοριθμική που υπολογίζει το πλήθος των αριστερών και δεξιών αριθμών ανά κλήρωση.

```
public int[] getLeftRightNumber()
   {
      int leftSum = 0;
      int rightSum = 0;
      int[] result = new int[2];
      ArrayList<Integer> leftrightNumbers = new ArrayList<Integer>();
      for (int i = 1; i < 6; i++)
      {
         leftrightNumbers = getEndingNumber(i);
         leftSum += leftrightNumbers.size();
         leftrightNumbers.clear();
      }

      for (int i = 6; i < 10; i++)
      {
         leftrightNumbers = getEndingNumber(i);
         rightSum += leftrightNumbers.size();
         leftrightNumbers.clear();
      }

      leftrightNumbers = getEndingNumber(0);
      rightSum += leftrightNumbers.size();
      leftrightNumbers.clear();
      result[0] = leftSum;
      result[1] = rightSum;
      return result;
   }
```

Όπως παρατηρείτε στον κώδικα, αρχικά χρησιμοποιείται μία επαναληπτική μέθοδος, όπως η for – loop , στο σώμα της οποίας εκτελείται η μέθοδος getEndingNumber για τους αριθμούς που ανήκουν στο σύνολο {1, 2, 3, 4, 5} για τον υπολογισμό του πλήθους των αριστερών αριθμών. Στη συνέχεια, εκτελείται ακόμα μία επαναληπτική μέθοδος for – loop, για τον υπολογισμό του πλήθους των δεξιών αριθμών για τον συγκεκριμένο συνδυασμό, για τον οποίο εκτελείται η παραπάνω μέθοδος. Έτσι, στο τέλος η συνάρτηση επιστρέφει στο σημείο της κλήσης της έναν πίνακα 2 εγγραφών, στον οποίο έχουν αποθηκευτεί οι μετρητές των πληθών των αριστερών και δεξιών αριθμών αντίστοιχα που βρίσκονται στο τρέχον συνδυασμό.

ΥΠΟΚΕΦΑΛΑΙΟ 1.11 Ο στατιστικός όρος Μέσα – Έξω αριθμών.

Ένας επιπλέον στατιστικός όρος που χρησιμοποιείται για την ομαδοποίηση των αριθμών των συνδυασμών είναι ο όρος «Μέσα – Έξω». Ένας αριθμός χαρακτηρίζεται ως «Μέσα» αν ανήκει στην ομάδα των εσωτερικών αριθμών, δηλαδή στο σύνολο $x \in$ [12, 19] ∪ [22, 29] ∪ [32, 35]. Αντίθετα «Έξω» αριθμός ονομάζεται ένας αριθμός που ανήκει στην ομάδα των εξωτερικών αριθμών, δηλαδή στο σύνολο $x \in$ [1, 11] ∪ [20, 21] ∪ [30, 31] ∪ [36, 45].

Ο συγκεκριμένος στατιστικός όρος υπολογίζει το πλήθος των αριθμών που έχουν χαρακτηριστεί ως Μέσα και το αντίστοιχο πλήθος των αριθμών που ανήκουν στην ομάδα των Έξω αριθμών. Συνεπώς, λόγω του ότι ένας συνδυασμός μπορεί να έχει πέντε αριθμός, υπολογίζοντας το πλήθος των μέσα αριθμών μπορούμε αφαιρώντας από το πέντε να βρούμε το πλήθος των έξω αριθμών ανά κλήρωση.

Συνεπώς, με τον παραπάνω στατιστικό όρο υπολογίζεται το πλήθος των κληρώσεων που περιέχουν κανέναν «Μέσα» αριθμό και πέντε «Έξω» αριθμούς, έναν «Μέσα» αριθμό και τέσσερις «Έξω» αριθμούς, κ.ά. Αντίθετα, στην ανάλυση της καθυστέρησης εμφάνισης των συνδυασμών των «Μέσα – Έξω» αριθμών, εκτός από το πλήθος καθυστέρησης, υπολογίζεται και ο μέσος όρος των συνόλων αυτών, όπως και η τυπική απόκλιση αυτών.

Η συνάρτηση της κλάσης JokerNumbers.java που είναι υπεύθυνη για τον χαρακτηρισμό των αριθμών ως «Μέσα» ή «Έξω» και τον υπολογισμό του πλήθους εμφάνισης έκαστου είναι η συνάρτηση getInOutNumbers(). Παρακάτω εμφανίζεται ο κώδικας java που χρησιμοποιείται για την υλοποίηση της αντίστοιχης αλγοριθμικής λογικής.

```
public int[] getInOutNumbers()
   {
      int[] result = new int[2];
      boolean flag = true;
      int inSum = 0;
      int outSum = 0;
      ArrayList<Integer> inNumber = new ArrayList<Integer>();
```

```
    ArrayList<Integer> outNumber = new ArrayList<Integer>();
    for (int i = 1; i <= 45; i++)
    {
        if (i == 12 || i == 22 || i == 32)
            flag = false;
        else if (i == 20 || i == 30 || i == 36)
            flag = true;
       if (flag) inNumber.add(i);
       else outNumber.add(i);
    }
    for (int i = 0; i < tempallNumbers.length; i++)
    {
        if (inNumber.contains(tempallNumbers[i]))
            inSum += 1;
        else
            outSum += 1;
    }

    result[0] = inSum;
    result[1] = outSum;
    return result;
}
```

Αρχικά στις δύο δομές δεδομένων ArrayList inNumber και outnumber καταχωρούμε τους αριθμούς που ανήκουν στο σύνολο των εσωτερικών και εξωτερικών αντίστοιχα αριθμών. Έτσι, για κάθε έναν αριθμό του τρέχοντος συνδυασμού για τον οποίο εκτελείται η αντίστοιχη μέθοδος, ελέγχεται σε ποια από τις δύο λίστες ανήκει και αναλόγως αυξάνουμε τον αντίστοιχο μετρητή του πλήθους. Το αποτέλεσμα αποθηκεύεται σε έναν πίνακα δύο εγγραφών ακέραιων αριθμών και επιστρέφεται στο σημείο κλήσης της συνάρτησης αυτής για την περαιτέρω στατιστική ανάλυση τους.

ΥΠΟΚΕΦΑΛΑΙΟ 1.12 Ο στατιστικός όρος Μικρά – Μεγάλα αριθμών.

Ένας αριθμός μπορεί να χαρακτηριστεί «Μικρός» αν ανήκει στο πρώτο μισό των συνολικών αριθμών, δηλαδή στο σύνολο $x \in$ [1, 22], ενώ αντίθετα «Μεγάλος» ονομάζεται ένας αριθμός που ανήκει στο δεύτερο μισό των συνολικών αριθμών, στο σύνολο $x \in$ [23, 45].

Συνεπώς, χρησιμοποιώντας την προηγούμενη έννοια μπορούμε να υπολογίσουμε το πλήθος των κληρώσεων που περιέχουν κάθε έναν συνδυασμό που μπορεί να προκύψει για τους πέντε διαφορετικούς αριθμούς της κλήρωσης. Συνεπώς, βρίσκεται το πλήθος των κληρώσεων που περιέχουν κανέναν «Μικρό» και πέντε «Μεγάλους» αριθμούς, έναν «Μικρό» και τέσσερεις «Μεγάλους» αριθμούς, κ.ά.

Η στατιστική ανάλυση της εμφάνισης «Μικρών – Μεγάλων» αριθμών υπολογίζει μόνο την συχνότητα εμφάνισης των αντίστοιχων συνδυασμών μικρών – μεγάλων αριθμών. Αντίθετα, στην στατιστική ανάλυση της καθυστέρησης της εμφάνισης

«Μικρών – Μεγάλων» αριθμών υπολογίζεται επιπλέον και ο μέσος όρος με την αντίστοιχη τυπική τους απόκλιση για κάθε συνδυασμό μικρών – μεγάλων αριθμών.

Η συνάρτηση της κλάσης JokerNumbers.java, η οποία χρησιμοποιείται για τον υπολογισμό του πλήθους των μικρών και μεγάλων αριθμών ανά κλήρωση είναι η getSmallBigNumber().

Παρακάτω εμφανίζεται το αντίστοιχο κομμάτι κώδικα java που έχει αναπτυχθεί για την υλοποίηση της αλγοριθμικής λογικής για τον συγκεκριμένο στατιστικό όρο.

```
public int[] getSmallBigNumber()
   {
      int[] result = new int[2];
      int smallSum = 0;
      int bigSum = 0;
      for (int i = 0; i < tempallNumbers.length; i++)
      {
         if (tempallNumbers[i] <= 22)
            smallSum += 1;
         else
            bigSum += 1;
      }
      result[0] = smallSum;
      result[1] = bigSum;
      return result;
   }
```

Όπως φαίνεται στον κώδικα, με την επαναληπτική διαδικασία for – loop διατρέχουμε όλους τους αριθμούς που ανήκουν στον συγκεκριμένο συνδυασμό, για τον οποίο εκτελείται η συνάρτηση αυτή, χαρακτηρίζοντας τους ως μικρό ή μεγάλο και αυξάνοντας τον αντίστοιχο μετρητή. Στο τέλος, οι δύο μετρητές που αναφέρονται στο πλήθος των αριθμών που ανήκουν στον συνδυασμό και είναι μικροί ή μεγάλοι αντίστοιχα αποθηκεύονται σε έναν πίνακα ακέραιων αριθμών δύο θέσεων και επιστρέφεται στο σημείο κλήσης της συνάρτησης για την περεταίρω ανάλυσή τους.

ΥΠΟΚΕΦΑΛΑΙΟ 1.13 Ο στατιστικός όρος Άθροισμα στήλης.

Όπως καθίσταται ξεκάθαρο από την ονομασία του στατιστικού όρου, «Άθροισμα στήλης» είναι ο όρος εκείνος, ο οποίος αναλύει στατιστικά την συχνότητα εμφάνισης και καθυστέρησης των αθροισμάτων των αριθμών ανά κλήρωση. Συνεπώς, υπολογίζεται το πλήθος των εμφανίσεων των αθροισμάτων ανά κλήρωση.

Η αλγοριθμική λογική του συγκεκριμένου στατιστικού όρου που έχει ακολουθηθεί για την υλοποίησή του στηρίζεται σε δύο βασικά στοιχεία. Αφενός υπολογίζονται όλα τα αθροίσματα που προκύπτουν ανά κλήρωση και αφετέρου η στατιστική ανάλυση αυτών των αθροισμάτων, δηλαδή εύρεση της συχνότητας εμφάνισης των

αθροισμάτων αυτών και ο μέσος όρος των συνόλων τους μαζί με την τυπική απόκλιση. Συνεπώς, αναλόγως της περιόδου για την οποία θα γίνει η αντίστοιχη στατιστική ανάλυση, θα προκύπτουν και διαφορετικά αθροίσματα των στηλών αυτών, όπως και διαφορετικοί μέσοι όροι και τυπικές αποκλίσεις.

ΥΠΟΚΕΦΑΛΑΙΟ 1.14 Ο στατιστικός όρος Εύρος στήλης.

Όπως με τον προηγούμενο στατιστικό όρο, τον «Άθροισμα» στήλης, έτσι και εδώ «Εύρος» στήλης είναι διαφορά του μεγαλύτερου αριθμού της στήλης από τον μικρότερο. Προφανώς, για την υλοποίηση αυτού, θα πρέπει για κάθε στήλη να ταξινομηθούν κατά αύξουσα τάξη οι αριθμοί κάθε μίας κλήρωσης.

Σε αυτή την περίπτωση, αρχικά υπολογίζονται για κάθε στήλη οι διαφορές που μπορούν να προκύψουν αφαιρώντας τον μεγαλύτερο αριθμό της στήλης από τον μικρότερο. Συνεπώς, ανάλογα την περίοδο για την οποία γίνεται η αντίστοιχη στατιστική μελέτη, υπολογίζονται και διαφορετικές διαφορές, με φυσικό επακόλουθο την εύρεση ανά περίοδο δεδομένων ξεχωριστών συχνοτήτων εμφάνισης και καθυστέρησης των διαφορών αυτών και επομένως, διαφορετικούς μέσους όρους και τυπικές αποκλίσεις.

ΥΠΟΚΕΦΑΛΑΙΟ 1.15 Ο στατιστικός όρος Άθροισμα ανά δύο.

Η εφαρμογή έχει αναπτυχθεί έτσι, ώστε να προσφέρεται στον χρήστη να διαλέγει δύο θέσεις στους συνδυασμούς των κληρώσεων και για αυτές τις δύο συγκεκριμένες θέσεις να γίνεται αντίστοιχη στατιστική μελέτη ως προς το άθροισμα των αριθμών των κληρώσεων που βρίσκονται σε κάθε περίπτωση σε αυτές τις δύο θέσεις.

Συνεπώς, βασικό στοιχείο για την άρτια εκτέλεση του στατιστικού όρου αυτού είναι η επιλογή των δύο θέσεων ανά στήλη, από τον χρήστη. Στη συνέχεια εκτελείται ο αντίστοιχος αλγόριθμος για τον υπολογισμό των αθροισμάτων των αριθμών που βρίσκονται στις συγκεκριμένες δύο θέσεις που επέλεξε ο χρήστης ανά στήλη. Για κάθε άθροισμα υπολογίζεται η συχνότητα εμφάνισής του στο σύνολο των στηλών που έχει επιλέξει ο χρήστης, καθώς επίσης και η εύρεση του μέσου όρου και της τυπικής απόκλισής τους.

ΥΠΟΚΕΦΑΛΑΙΟ 1.16 Ο στατιστικός όρος Εύρος ανά δύο.

Ανάλογα με τον παραπάνω στατιστικό όρο, υπάρχει η δυνατότητα για τον χρήστη να μελετήσει την αντίστοιχη στατιστική ανάλυση για τους αριθμούς που βρίσκονται σε δύο συγκεκριμένες θέσεις ανά κλήρωση και πιο συγκεκριμένα για την στατιστική ανάλυση του εύρους των αριθμών που βρίσκονται σε αυτές τις δύο θέσεις μόνο, στο σύνολο των στηλών ανά περίοδο επιλογής του χρήστη.

Έτσι, επιλέγοντας ο χρήστης κάποια χρονική περίοδο για την οποία θα συλλεχθούν τα διαθέσιμα δεδομένα, καθώς επίσης και των δύο θέσεων των αριθμών ανά στήλη, ως προς τα οποία θα γίνει στατιστική ανάλυση του εύρους τους, υπολογίζονται οι απόλυτες διαφορές των αριθμών που βρίσκονται σε αυτές τις δύο θέσεις ανά στήλη. Επίσης, υπολογίζεται ο μέσος όρος και η τυπική απόκλιση του συνόλου αυτού.

ΥΠΟΚΕΦΑΛΑΙΟ 1.17 Ο στατιστικός όρος Λήγοντας αριθμός για το Joker.

Η στατιστική ανάλυση που γίνεται στη συγκεκριμένη εφαρμογή περιλαμβάνει και την αντίστοιχη ανάλυση των αριθμών για τους αριθμούς που μπορούν να επιλεγούν για το Joker, οι οποίοι είναι το σύνολο {1, 2, 3, …,19, 20}.

Συνεπώς, όπως και στην στατιστική ανάλυση των αριθμών έτσι και για τους αριθμούς του joker οι στατιστικοί όροι στηρίζονται κατά κύριο λόγο στην συχνότητα εμφάνισης και καθυστέρησης των αντίστοιχων αριθμών joker στο σύνολο των προγενέστερων κληρώσεων που έχουν ήδη πραγματοποιηθεί.

Έχοντας ως πηγή πληροφορίας τους ήδη κληρωθέντες αριθμούς joker γίνεται καταγραφή και υπολογισμός των αντίστοιχων στατιστικών όρων. Πιο συγκεκριμένα ένας από τους στατιστικούς όρους που χρησιμοποιείται για τους αριθμούς joker είναι ο λήγοντας αριθμός. Έχει ήδη ειπωθεί η έννοια του στατιστικού όρου που αναφέρεται στους κληρωθέντες αριθμούς του «τζόκερ». Λήγοντας αριθμός αναφέρεται το ψηφίο στο οποίο λήγει ένας αριθμός.

Η εφαρμογή υπολογίζει την συχνότητα εμφάνισης και καθυστέρησης αυτών των ληγόντων αριθμών των αριθμών joker για κάποια συγκεκριμένη χρονική που έχει επιλέξει ο χρήστης. Για τον υπολογισμό του στατιστικού αυτού όρου έχει υλοποιηθεί στην κλάση JokerNumbers.java η συνάρτηση getLigontesJoker, η οποία βρίσκει για κάθε μία κλήρωση τον λήγοντα αριθμό του joker που κληρώθηκε στην τρέχουσα κλήρωση. Οπότε, στο κυρίως πρόγραμμα που γίνονται οι υπολογισμοί για την στατιστική ανάλυση του λήγοντα αριθμού του joker, υπολογίζεται η συχνότητα και τυπική απόκλιση του συνόλου των αριθμών εμφάνισης και καθυστέρησης των ληγόντων αριθμών του joker.

ΥΠΟΚΕΦΑΛΑΙΟ 1.18 Ο στατιστικός όρος Δεκάδα για το Joker.

Λόγω του ότι ο αριθμός joker των κληρώσεων μπορεί να ανήκει στο σύνολο {1, 2, 3, …, 19, 20} υπάρχουν μόνο 3 δεκάδες αριθμών. Αυτοί που βρίσκονται στην πρώτη δεκάδα, δηλαδή είναι ένας από τους αριθμούς {1, 2, 3, …, 9}, αυτοί της δεύτερης δεκάδας, δηλαδή είναι ένας από τους αριθμούς {10, 11, 12, …, 19} και οι αριθμοί της τρίτης δεκάδας, που είναι μόνο ο αριθμός 20.

Η συνάρτηση της κλάσης JokerNumbers.java, που υπολογίζει την δεκάδα στην οποία ανήκει ο αριθμό του joker των κληρώσεων είναι η getDekadesJoker. Η

λογική είναι πολύ απλή και είναι ίδια με αυτήν που αναπτύχθηκε και στην ανάλυση του στατιστικού όρου «Δεκάδα» αριθμών.

ΥΠΟΚΕΦΑΛΑΙΟ 1.19 Ο στατιστικός όρος Ταυτάριθμοι αριθμοί για το Joker.

Ταυτάριθμος αριθμός του joker είναι το μονοψήφιο άθροισμα κάποιου αριθμού που μπορεί να είναι joker. Ως γνωστόν, το σύνολο από το οποίο μπορεί να κληρωθεί ένας αριθμός joker είναι το σύνολο {1, 2, 3, …, 19, 20}. Για τον υπολογισμό του ταυτάριθμου αυτών των αριθμών θα πρέπει να υπολογίζεται συνεχώς το άθροισμα των ψηφίων του αριθμού, μέχρι το αποτέλεσμα να είναι μονοψήφιος αριθμός.

Για την εύρεση του ταυτάριθμου ανά κλήρωση joker έχει αναπτυχθεί ξεχωριστή συνάρτηση στη κλάση JokerNumbers.java, η οποία είναι η getTautarithmoiJoker. Η αλγοριθμική λογική της ανάπτυξης και υπολογισμού του συγκεκριμένου στατιστικού όρου είναι ανάλογη της λογικής που αναλύθηκε στο υποκεφάλαιο 1.7, εκεί όπου έχει επεξηγηθεί η διαδικασία υπολογισμού και εύρεσης της συχνότητας εμφάνισης και καθυστέρησης των ταυτάριθμων των αριθμών ανά κλήρωση.

ΥΠΟΚΕΦΑΛΑΙΟ 1.20 Ο στατιστικός όρος Μονά – Ζυγά για το Joker.

Αυτό που επιχειρήθηκε σε αυτόν τον στατιστικό όρο ήταν να υπολογισθεί και να αναλυθεί στατιστικώς η συχνότητα εμφάνισης και καθυστέρησης των αριθμών joker, οι οποίοι είναι μονοί ή ζυγοί. Συνεπώς, αρχικά για κάθε αριθμό joker που έχει κληρωθεί χαρακτηρίζεται μονός ή ζυγός και στη συνέχεια υπολογίζεται η αντίστοιχη συχνότητα στην εμφάνιση και καθυστέρηση των μονών και ζυγών αριθμών.

Για την ανάπτυξη του παραπάνω, χρησιμοποιείται μία άλλη συνάρτηση της κλάσης JokerNumbers.java, η οποία ύστερα από τους κατάλληλους υπολογισμούς χαρακτηρίζει κάθε joker των κληρώσεων ανά χρονική περίοδο, ως μονούς ή ζυγούς. Προφανώς, ένας αριθμός joker θα είναι μονός αν το υπόλοιπο της διαίρεσης του αριθμού αυτού με το δύο είναι ίσο με ένα, αλλιώς αν είναι μηδέν, τότε χαρακτηρίζεται ως ζυγός.

ΥΠΟΚΕΦΑΛΑΙΟ 1.21 Ο στατιστικός όρος Αριστερά – Δεξιά για το Joker.

Ένας αριθμός joker χαρακτηρίζεται ως αριστερός αν ο λήγοντας αριθμός του είναι μικρότερος ή ίσος του πέντε, αλλιώς θεωρείται δεξιός. Συνεπώς, για την στατιστική ανάλυση του όρου αυτού, αρχικά χρησιμοποιείται μία συνάρτηση της κλάσης JokerNumbers.java, η isAristeraDeksiaJoker, της οποίας ο στόχος είναι για κάθε αριθμό joker ανά κλήρωση, να βρίσκει τον λήγοντα αριθμό του, χρησιμοποιώντας τη συνάρτηση getTautarithmoiJoker, όπως αναλύθηκε στο υποκεφάλαιο 1.17 και

αν αυτός είναι μικρότερος ή ίσος του πέντε, τότε χαρακτηρίζεται ο αριθμός joker ως αριστερός, αλλιώς είναι δεξιός.

Στη συνέχεια, η ροή του προγράμματος μεταφέρεται στο κυρίως πρόγραμμα, εκεί όπου θα συλλεχθούν όλοι οι αριθμοί joker, οι οποίοι έχουν ήδη χαρακτηριστεί ως αριστεροί και δεξιοί. Με βάσει αυτές τις πληροφορίες γίνεται η στατιστική ανάλυση του όρου αυτού, δηλαδή βρίσκεται η συχνότητα εμφάνισης και καθυστέρησης των αριθμών αυτών.

ΥΠΟΚΕΦΑΛΑΙΟ 1.22 Ο στατιστικός όρος Μικρός – Μεγάλος για το Joker.

Ένας αριθμός joker θα χαρακτηρίζεται μικρός αν και μόνον εάν είναι μικρότερος ή ίσος του δέκα. Αλλιώς θεωρείται μεγάλος. Συνεπώς, η λογική είναι αρκετά απλή. Για κάθε αριθμό joker του συνόλου των κληρώσεων για κάποια συγκεκριμένη χρονική περίοδο, βρίσκουμε αν είναι μικρός ή μεγάλος και στη συνέχεια αφού το κυρίως πρόγραμμα πάρει το κύριο νήμα (thread), γίνονται όλοι οι απαραίτητοι υπολογισμοί για την εύρεση της συχνότητας εμφάνισης και καθυστέρησης των μικρών και μεγάλων αριθμών joker.

Υπάρχει μια συγκεκριμένη μέθοδος της κλάσης JokerNumbers.java, η οποία ελέγχει και χαρακτηρίζει τον αριθμό joker που έχει κληρωθεί ανά κλήρωση, ως μικρό ή μεγάλο. Στη συνέχεια το αποτέλεσμα μεταφέρεται στο σημείο από όπου κλήθηκε η συνάρτηση αυτή, που είναι το κύριο νήμα για την περεταίρω επεξεργασία των αποτελεσμάτων αυτής της μεθόδου. Το όνομα της συνάρτησης αυτής είναι isMikrosJoker.

ΕΠΙΛΟΓΟΣ

Σε κάθε ολοκληρωμένο αυτοματοποιημένο υπολογιστικό σύστημα που έχει αναπτυχθεί μέχρι τώρα στο παγκόσμιο στερέωμα, όπως και αυτά που θα υλοποιηθούν στο εγγύς μέλλον δεν μπορούν να παρουσιάσουν επιτυχίες αν το θεωρητικό υπόβαθρο και η αλγοριθμική λογική που ακολουθείται δεν βρίσκονται σε ένα πολύ καλό επίπεδο. Για τον λόγο αυτό, ο κυρίαρχος στόχος μου κατά την καταγραφή των λειτουργικών απαιτήσεων του παρόντος project ήταν να θέσουμε ο επιβλέποντας καθηγητής μου, κ. Σφέτσος και εγώ πολύ υψηλά standards για την καταγραφή, υπολογισμό, ανάλυση και διαχείριση των αποτελεσμάτων από την συνολική στατιστική ανάλυση που μπορεί να αναπτυχθεί χρησιμοποιώντας όλους ή μέρος των παραπάνω στατιστικών όρων.

Αποτέλεσμα όλων των παραπάνω ήταν ένα πάρα πολύ μεγάλο μέρος της διάρκειας που χρειάστηκα για να ολοκληρώσω την πτυχιακή μου εργασία να αφιερωθεί στην αναζήτηση, καταγραφή και κατ’ επέκταση ανάπτυξη του θεωρητικού τμήματος της στατιστικής ανάλυσης και δη στην υλοποίηση όλων

εκείνων των αλγορίθμων, που είναι απαραίτητοι για την αρτιότερη λειτουργία της εφαρμογής.

Ο σχεδιασμός που έχει πραγματοποιηθεί για να υποστηριχθεί η διαχείριση και εκμετάλλευση των αποτελεσμάτων είναι αρκετά λειτουργικός και τόσο εμπλουτισμένος με διαδικασίες και δυνατότητες ώστε η χρήση της εφαρμογής να επιτρέπει τον χρήστη να καταλήγει σε συμπεράσματα ως προς ποιοι αριθμοί σύμφωνα με την στατιστική ανάλυση και των αντίστοιχων δεδομένων που προκύπτουν έχουν τις περισσότερες πιθανότητες αν εκλεγούν την επόμενη φορά.

Η επιστήμη των μαθηματικών θεωρούνται από τις αρχαιότερες των θετικών σπουδών. Συνεπώς, το σύνολο της θεωρίας που έχει αναπτυχθεί μέχρι τις μέρες καθιστούν αδύνατο στο μεγαλύτερο μέρος της να χρησιμοποιηθούν όλες οι επιμέρους πτυχές διάφορων θεματικών ενοτήτων. Ως εκ τούτου, η στατιστική ανάλυση που ακολουθήθηκε και αναπτύχθηκε στη παρούσα πτυχιακή, ίσως να μειονεκτεί μπροστά στο σύνολο της θεωρίας της στατιστικής. Ωστόσο, καταβλήθηκε πολύ μεγάλη προσπάθεια στην όσο το δυνατόν καλύτερη και αποτελεσματικότερη εφαρμογή των θεωρητικών αποτελεσμάτων της στατιστικής επάνω στη βάση δεδομένων που δημιουργείται από το σύνολο των μέχρι και σήμερα κληρώσεων.

Ύστερα από αρκετή και προσεκτική μελέτη των αποτελεσμάτων που προκύπτουν από τη αντίστοιχη στατιστική ανάλυση όπως πραγματοποιείται από το πρόγραμμα, από πλευράς του χρήστη, του δίνεται η δυνατότητα να καταλήξει σε συμπεράσματα, τα οποία με κάποιο φυσικά μικρό σφάλμα, να μπορούν να θεωρηθούν ασφαλή.

Η εφαρμογή έχει αναπτυχθεί με τέτοιον τρόπο έτσι, ώστε να επιτρέπει στον χρήστη, αφού μελετήσει προσεχτικά όλη την στατιστική μελέτη που έχει αναπτυχθεί για την επιλεγμένη από τον χρήστη χρονική περίοδο, να επιλέγει συγκεκριμένους στατιστικούς όρους, οι οποίοι λειτουργούν ως φίλτρα στην επιλογή των αριθμών για τους τελικούς συνδυασμούς, όπως επίσης, και ως φίλτρα στην δημιουργία των αντίστοιχων συνδυασμών που μπορούν να δημιουργηθούν από τους επιλεγμένους αριθμούς.

Αποτέλεσμα του προηγούμενου στοιχείου είναι η εφαρμογή να παρέχει στον χρήστη όλες εκείνες τις δυνατότητες, οι οποίες συνδυαζόμενες μεταξύ τους σύμφωνα με τις επιθυμίες του χρήστη, τελικά να επιλέγεται μέσα από μία λίστα κάποιων συνδυασμών, των οποίων οι αριθμοί ικανοποιούν τους περιορισμούς των στατιστικών δεδομένων που έχει επιλέξει ο χρήστης, ένας ή περισσότεροι συνδυασμοί.

Ωστόσο, θα ήθελα σε αυτό το σημείο να τονίσω το συμπέρασμα στο οποίο καταλήγουν αβίαστα, όλοι οι υγιώς σκεπτόμενοι άνθρωποι, ότι η επιτυχία στο παιχνίδι τύχης του ΟΠΑΠ, το «τζόκερ» δεν είναι άμεσα εξαρτώμενη από τα στατιστικά αποτελέσματα που μπορούν να προκύψουν από κάποια στατιστική

μελέτη που εκτελεί ο χρήστης του προγράμματος. Το συγκεκριμένο παιχνίδι κατατάσσεται στα αντίστοιχα παιγνίδια τζόγου και χρειάζεται μεγάλη δόση τύχης του χρήστη για να επιτύχει το νικητήριο συνδυασμό.

Ως γνωστόν, η πιθανότητα επιτυχίας στο συγκεκριμένο είναι πάρα πολύ μικρή, ειδικά όταν ο χρήστης επιλέγει μόνο πέντε αριθμούς από τους διαθέσιμους σαράντα πέντε, όπως επίσης ένα μόνο αριθμό joker από τους είκοσι που υπάρχουν για την τελική απόφαση.

Ο τύπος που υπολογίζει το πλήθος όλων των πιθανών συνδυασμών που μπορούν να πραγματοποιηθούν μέσα από ένα σύνολο σαράντα πέντε αριθμών όταν επιλέγονται πέντε διαφορετικοί αριθμοί, είναι ο παρακάτω:

$$\binom{45}{5} = \frac{7*9*11*41*43}{1} = 1221759 \qquad (1.6)$$

ενώ το πλήθος των διαφορετικών συνδυασμών που μπορούν να δημιουργηθούν επιλέγοντας έναν αριθμό joker από το σύνολο των είκοσι διαφορετικών αριθμών joker δίνεται από τον τύπο:

$$\binom{20}{1} = \frac{20!}{19!} = 20 \qquad (1.7)$$

Όπως πολύ εύκολα μπορεί να αντιληφθεί, για να κερδίσει κάποιος το «τζόκερ» έχει πιθανότητες 1 / 1221759 * 20, δηλαδή είναι πάρα πολύ δύσκολο. Συνεπώς, χρησιμοποιώντας την στατιστική ανάλυση που προσφέρεται από την εφαρμογή αυτή, δίνεται στον χρήστη να μπορέσει κάποιον συνδυασμό αριθμών με καλύτερες πιθανότητες επιτυχίας, αφού αρκετές φορές η στατιστική επαληθεύεται σε τέτοιου είδους «τυχερών» αριθμών.

Η μελέτη, ανάπτυξη και υλοποίηση της εφαρμογής, όπως προαναφέρθηκε, στηρίζεται στα συμπεράσματα που μπορεί ένας χρήστης να καταλήξει με βάσει των θεωρημάτων και αποτελεσμάτων που προκύπτουν από την ανάλυση της θεωρίας της στατιστικής. Ωστόσο, δεν θα ήταν χρήσιμα αν δεν είχε αναπτυχθεί εξίσου εύστοχα και εύχρηστα η αντίστοιχη εφαρμογή χρησιμοποιώντας γλώσσα προγραμματισμού, την Java και έναν ακόμα συνδυασμό πολλών άλλων τεχνολογιών, οι οποίες εμπλουτίζουν αρμονικά τις λειτουργίες και δυνατότητες της Java. Με αποτέλεσμα να προσφέρεται στον χρήστη η τελική μορφή της εφαρμογής, με δυνατότητες και επιλογές στατιστικής ανάλυσης όρων και διαχείρισή τους, όπως επίσης και επιπλέον δημιουργία βασικών στηλών για το φιλτράρισμα των συνδυασμών που ορίζονται από τους σαράντα πέντε αριθμών.

Στο επόμενο κεφάλαιο αναπτύσσονται λεπτομερέστερα όλες εκείνες οι τεχνολογίες που χρησιμοποιήθηκαν στην υλοποίηση της εφαρμογής στο πλαίσιο του κώδικα.

ΚΕΦΑΛΑΙΟ 2: Δημιουργία και διαχείριση τοπικής βάσης δεδομένων με την χρήση της βιβλιοθήκης SQLite.

Σε κάθε στατιστική ανάλυση κάποιων χαρακτηριστικών ενός πληθυσμού η εγκυρότητα των αποτελεσμάτων εξαρτάται σε μεγάλο βαθμό από την σωστή επιλογή του δείγματος του πληθυσμού για το οποίο θα γίνει η ανάλυση. Όσο μεγαλύτερο είναι το δείγμα, τόσο ασφαλέστερα συμπεράσματα μπορούν να προκύψουν για διάφορα χαρακτηριστικά του πληθυσμού.

Μία από τις λειτουργικές απαιτήσεις της εφαρμογής ήταν η όσο το δυνατόν εγκυρότερη και γρήγορη απόκριση στα αποτελέσματα της στατιστικής ανάλυσης που θα πραγματοποιείται για τους διάφορους στατιστικούς όρους, όπως αυτοί αναλύθηκαν στο προηγούμενο κεφάλαιο. Για να ικανοποιηθεί η συγκεκριμένη απαίτηση, ύστερα από ανάλογη έρευνα στον κυβερνοχώρο, κρίθηκε χρήσιμο να χρησιμοποιηθεί μία πρόσθετη βιβλιοθήκη με την επωνυμία SQLite.

ΕΙΣΑΓΩΓΗ

Τον Απρίλιο του 2004 ξεκίνησε η ομάδα ανάπτυξης της αντίστοιχης βιβλιοθήκης, χρησιμοποιώντας ως γλώσσα προγραμματισμού την C να υλοποιούν την συγκεκριμένη βιβλιοθήκη, με απώτερο σκοπό την δημιουργία μιας αυτοτελούς, ανεξάρτητης από servers και ρυθμίσεις εγκατάστασης στα τερματικά και τον server μηχανής βάσης δεδομένων. Τελικά η beta έκδοση της SQLite 3 δόθηκε με ελεύθερα δικαιώματα χρήσης τέλη του Ιουνίου 2004. Αρκετά χρόνια πέρασαν εκ τότε με πολλές εκδόσεις να δίνονται στο ευρύ κοινό μέχρι την πιο πρόσφατη, που αντιστοιχεί στην έκδοση 3.7.7.1 στις 28 Ιουνίου 2011, καθώς η χρήση της βρίσκει εφαρμογές σε μεγάλα project με μεγάλους σπόνσορες, όπως ORACLE, NOKIA, ADOBE, κ.ά.

ΥΠΟΕΝΟΤΗΤΑ 2.1: Ανεξάρτητη, εύχρηστη και χωρίς ρυθμίσεις διαχειριστή, βάση δεδομένων

Αντίθετα από τις περισσότερες βάσεις δεδομένων, η SQLite δεν είναι μια ξεχωριστή διαδικασία κάποιου εξυπηρετητή. Έχει αναπτυχθεί έτσι, ώστε να διαβάζει και να γράφει απευθείας σε συνηθισμένα αρχεία που βρίσκονται στο δίσκο του τερματικού στο οποίο εκτελείται η εφαρμογή. Μία ολοκληρωμένη βάση δεδομένων που υποστηρίζει πολλαπλούς πίνακες δεδομένων, δείκτες, triggers και views συνδέονται σε ένα απλό αρχείο δίσκου με κατάληξη .dat.

Το μέγεθος του αρχείου αυτού είναι πολύ μικρό, λιγότερο από 350 Kbit αναλόγως το λειτουργικό του τερματικού, ενώ αν περιοριστούν οι λειτουργίες της βάσης δεδομένων, το μέγεθος του αρχείου μπορεί να περιοριστεί κάτω από 200 Kbit. Ένας από τους σημαντικότερους λόγους που καθιέρωσαν την βιβλιοθήκη SQLite

στο χώρο ανάπτυξης εφαρμογών με χρήση βάσεων δεδομένων ήταν η χρήση της σε μικρά gadgets, όπως smartphones, PDAs και MP3 players, λόγω του μικρού χώρου που απαιτεί για την χρήση της. Φυσικά, σε περιβάλλοντα με αρκετή μνήμη η βιβλιοθήκη SQLite δουλεύει ακόμα καλύτερα.

Οι περισσότερες βάσεις δεδομένων αποτελούν μια ξεχωριστή λειτουργία του εξυπηρετητή. Εφαρμογές που δημιουργούν σύνδεση με τη βάση, αρχικά επικοινωνούν με τον εξυπηρετητή χρησιμοποιώντας κάποια μορφή ενδοεπικοινωνίας (συνήθως TCP / IP) για να στέλνουν αιτήματα στον εξυπηρετητή και να λαμβάνουν τις αποκρίσεις. Η τεχνολογία που παρέχεται από την SQLite δεν ακολουθεί το παραπάνω μοντέλο, καθώς η εφαρμογή επικοινωνεί άμεσα με το αρχείο που έχει δημιουργηθεί και διαμορφωθεί για να υποστηρίζει την μηχανή της βάσης δεδομένων και είναι αποθηκευμένο σε κάποιο φάκελο στο δίσκο.

Φυσικά, η ανεξαρτησία της SQLite από τους εξυπηρετητές έχει τόσο πλεονεκτήματα, όσο και μειονεκτήματα. Ένα από τα πλεονεκτήματά της είναι ότι δεν χρειάζεται κάποια μορφή εγκατάστασης, ρύθμισης και αρχικοποίησης για να είναι λειτουργική. Δεν είναι απαραίτητη η βοήθεια από κάποιον διαχειριστή της βάσης για να χρησιμοποιηθεί. Όσες εφαρμογές και προγράμματα έχουν την δυνατότητα να έχουν πρόσβαση στο σύστημα των φακέλων και των αρχείων, μπορούν να συνδεθούν και να χρησιμοποιήσουν τη βάση.

Στον αντίποδα, χρησιμοποιώντας μία βάση δεδομένων μέσω ενός server προσφέρει μεγαλύτερη ασφάλεια από τις «τρύπες» στην εφαρμογή του πελάτη. Επειδή κάθε λειτουργία του server είναι και μία ξεχωριστή διαδικασία, παρέχεται η δυνατότητα για μεγαλύτερο έλεγχο των συνδέσεων στη βάση δεδομένων που βρίσκεται στον εξυπηρετητή και των περιπτώσεων που κλειδώνει κάποια βάση δεδομένων από την σύνδεση.

Οι περισσότερες βάσει δεδομένων στηρίζονται στο μοντέλο client / server. Όμως από αυτές που δεν χρησιμοποιούν κάποιον εξυπηρετητή για να λειτουργήσουν, η SQLite είναι η μοναδική που επιτρέπει πολλαπλές εφαρμογές να συνδέονται και να χρησιμοποιούν την ίδια βάση την ίδια ακριβώς στιγμή.

ΥΠΟΕΝΟΤΗΤΑ 2.2: Σύνδεση με την τοπική βάση με χρήση της SQLite βιβλιοθήκης.

Ύστερα από την ανάλυση που κάναμε παραπάνω σχετικά με τη βιβλιοθήκη SQLite και την χρήση της στην δημιουργία και διαχείριση μιας τοπικής βάσης δεδομένων, σε αυτή την υποενότητα θα εξηγηθεί η λειτουργία της SQLite αναλύοντας τις αντίστοιχες εντολές που έχουν χρησιμοποιηθεί στον κώδικα της εφαρμογής έτσι, ώστε να υλοποιηθεί.

Αρχικά θα εξηγήσουμε ποια είναι η τεχνολογία JDBC (Java DataBase Connectivity). Το βασικό στοιχείο της τεχνολογίας αυτής είναι ότι παρέχει στον

προγραμματιστή την διεπαφή προγραμματισμού εφαρμογών, κοινώς το API για την σύνδεση και χρήση σε διαφορετικών συστημάτων διαχείρισης βάσεων δεδομένων, χρησιμοποιώντας απλώς ένα εκτελέσιμο αρχείο (.jar). Το αρχείο αυτό είναι το sqlitejdbc-v056.jar, το οποίο μπορούμε να βρούμε και να κατεβάσουμε δωρεάν από την επίσημη ιστοσελίδα της SQLite, http://www.sqlite.org/ .Στη συνέχεια αφού το φορτώσουμε μέσα στο project, στο φάκελο των βιβλιοθηκών Libraries μας επιτρέπεται να χρησιμοποιήσουμε το API που προσφέρεται από την SQLite για την σύνδεσή μας με την τοπική βάση δεδομένων.

Για την σύνδεση στη βάση θα πρέπει να ακολουθήσουμε τα παρακάτω βήματα. Ξεκινώντας, θα πρέπει να εισάγουμε στο κώδικα της εφαρμογής τις κατάλληλες βιβλιοθήκες για την σύνδεση και χρήση των λειτουργιών που προσφέρονται από το αντίστοιχο αρχείο jar της SQLite για την διεπαφή της εφαρμογής με το αντίστοιχο αρχείο της βάσης. Οπότε έξω από όλες τις κλάσεις θα πρέπει να γράψουμε τις παρακάτω εντολές:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
```

Η χρησιμότητα των παραπάνω είναι πολύ μεγάλη, αφού πλέον έχουμε τη δυνατότητα να χρησιμοποιήσουμε όλες εκείνες τις εντολές του API για να αποκτήσουμε πρόσβαση σε όλες τις αντίστοιχες λειτουργίες. Μερικές από τις λειτουργίες αυτές είναι η σύνδεση με την τοπική βάση που δημιουργείται όταν εκτελείται για πρώτη φορά στο μηχάνημα, η δημιουργία παραμετρικών SqlCommands ερωτημάτων για την αλληλεπίδραση χρήστη με βάση δεδομένων. Υπάρχει η λειτουργία δημιουργία ResultSet, δηλαδή η προσωρινή δημιουργία πινάκων – views με τα αποτελέσματα των Select sql ερωτημάτων προς τη βάση. Έτσι μπορούμε να δημιουργήσουμε στην προσωρινή μνήμη του τερματικού στο οποίο εκτελείται η εφαρμογή, τους αντίστοιχους πίνακες που ορίζουν τα αποτελέσματα τέτοιων ερωτημάτων. Ακόμα η χρήση των παραπάνω εντολών μας επιτρέπουν να διαχειριστούμε όπως θέλουμε εμείς όποια τυχόν λάθη που μπορεί να προκύψουν κατά την αλληλεπίδραση της εφαρμογής μας με την βάση.

Στη συνέχεια, με το μπλοκ εντολών:

```
try {
            Class.forName("org.sqlite.JDBC");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(SplashScreenMain.class.getName()).log(Level.SEVERE,
null, ex);
        }
```

Φορτώνουμε το πρόγραμμα οδήγησης JDBC της τοπικής βάσης SQLite, το οποίο θα μας προσφέρει το κατάλληλο API για την καλύτερη αλληλεπίδραση με τις δυνατότητες της βάσης. Η εντολή που χρησιμοποιείται για αυτή την εντολή είναι Class.forName(“org.sqlite.JDBC).

Ορισμός του αντίστοιχου URL για την σύνδεση , όπου η θέση του εξυπηρετητή της βάσης δεδομένων, η θύρα και το όνομα της Βάσης δεδομένων Η εντολή που χρησιμοποιείται για την σύνδεση με κάποια βάση SQLite είναι η παρακάτω:

```
try {
            connection = DriverManager.getConnection("jdbc:sqlite:sample.db");
        } catch (SQLException ex) {
            Logger.getLogger(SplashScreenMain.class.getName()).log(Level.SEVERE,
null, ex);
        }
```

Έπειτα, για τη δημιουργία της δικτυακής σύνδεσης με τη βάση δεδομένων με χρήση του URL και της αρχικοποίησης του αντικειμένου Statement, χρησιμοποιείται τη παρακάτω εντολή:

```
Statement statement = null;

        try {
            statement = connection.createStatement();
        } catch (SQLException ex) {
            Logger.getLogger(SplashScreenMain.class.getName()).log(Level.SEVERE,
null, ex);
        }
```

Αφού έχει καθιερωθεί η σύνδεση με την τοπική βάση με χρήση του αρχείου .jar, και έχουμε αρχικοποιήσει το αντικείμενο Statement, μπορούμε να εκτελέσουμε οποιοδήποτε Sql command, χρησιμοποιώντας τις μεθόδους της κλάσης Statement

όπως το executeUpdate, executeQuery.

Αν τα αποτελέσματα υπερβαίνουν τη 1 γραμμής, τότε μπορούμε να ορίσουμε σε έναν χώρο στην μνήμη RAM την δομή των πινάκων που χρησιμοποιούνται στα ερωτήματα αυτά και να τους γεμίσουμε με τις απαραίτητες εγγραφές, που φέρνει το αντίστοιχο SqlCommand(). Η εντολή στη java για τη δημιουργία ενός ResultSet είναι η παρακάτω:

```
int thereIsTable = 0;
            ResultSet rset = statement.executeQuery("Select case name when '"+
tableNamesCreate[j][0] + "' then 1 else 0 end as 'status' from sqlite_master where name =
'"+ tableNamesCreate[j][0] + "' and type ='table'");

            while (rset.next())
            {
                thereIsTable = 1;
            }
            rset.close();
            if (thereIsTable == 0)
                statement.executeUpdate(tableNamesCreate[j][1]);
```

Παρατηρούμε ότι για να διαβάσουμε όλες τις εγγραφές, με τις οποίες γεμίζουν οι πίνακες στην προσωρινή μνήμη, χρησιμοποιείται η μέθοδος next() του αντικειμένου της κλάσης ResultSet. Έτσι, μπορούμε να διαβάσουμε τα πεδία ανά γραμμή και να υλοποιήσουμε τον αντίστοιχο κώδικα. Τέλος, θα πρέπει να γίνεται αποσύνδεση από τη βάση δεδομένων κάθε φορά που ολοκληρώνεται το πρωινό.

Πολλές φορές, αντί για ένα στατικό Sql command, χρειαζόμαστε να ορίσουμε και να αρχικοποιήσουμε ένα παραμετρικό ερώτημα, όπως π.χ. δίνεται παρακάτω:

```
PreparedStatement prep;
      prep = connection.prepareStatement("insert into joker (klirwsi, date, n1, n2, n3, n4,
n5, nJoker) values (?, ?, ?, ?, ?, ?, ?, ?);");
for (int i = 1; i < 46; i++)
      {
         int ligontas, dekada, tautarithmos, monos, mikros, mesa;
         ligontas = Ligontas(i);
         dekada = Dekada(i);
         tautarithmos = Tautarithmos(i);
         monos = isMonos(i);
         mikros = isMikros(i);
         mesa = isMesa(i);
         insertIntoNumbers.setInt(1, i);
         insertIntoNumbers.setInt(2, 0);
         insertIntoNumbers.setInt(3, ligontas);
         insertIntoNumbers.setInt(4, dekada);
         insertIntoNumbers.setInt(5, tautarithmos);
         insertIntoNumbers.setInt(6, monos);
         insertIntoNumbers.setInt(7, mikros);
         insertIntoNumbers.setInt(8, mesa);
         insertIntoNumbers.addBatch();
         screen.setProgress(progressBarValue);
         progressBarValue++;
      }
```

Παρατηρούμε ότι στη δημιουργία παραμετρικών ερωτημάτων, χρησιμοποιούνται οι ειδικοί χαρακτήρισες ‘?’, που δηλώνουν στον οδηγό JDBC ότι αυτές οι τιμές θα δεχθούν τιμή στη συνέχεια. Για την ενημέρωση κάποιων πεδίων της βάσης παραμετρικά έχουμε τη δυνατότητα να καταχωρήσουμε πρώτα όλα εκείνα τα πεδία που θα ενημερώσουν τα πεδία της βάσης όπως δηλώνει η εντολή insertIntoNumbers.addBatch() και στη συνέχεια η εκτέλεση όλων αυτών των παραμέτρων χρησιμοποιώντας τις εντολές

```
connection.setAutoCommit(false);
 insertIntoNumbers.executeBatch();
 connection.setAutoCommit(true);
```

ΕΠΙΛΟΓΟΣ

Ένα μεγάλο ποσοστό επιτυχίας της εφαρμογής ως προς την στατιστική ανάλυση και μελέτη των αποτελεσμάτων των κληρώσεων από το παιγνίδι τύχης «τζόκερ», στηρίζεται στη σωστή δομή αποθήκευσης του τεράστιου όγκου δεδομένων που

έχει συσσωρευτεί έπειτα από τόσα πολλά χρόνια ύπαρξης του αντίστοιχου παιγνιδιού.

Για να γίνει αυτό, ξοδεύτηκε μεγάλο μέρος από τον διαθέσιμο χρόνο που είχα στην εκπόνηση της εργασίας. Όπως έδειξε το αποτέλεσμα, πιστεύω ότι τελικά άξιζε αυτή η προσπάθεια που καταβλήθηκε για να λυθεί σωστά η συγκεκριμένη παράμετρος στην ανάπτυξη της εφαρμογής.

Η βιβλιοθήκη της SQLite είναι ένα αρκετά καλό εργαλείο ελεύθερου λογισμικού με μεγάλες εταιρίες που την υποστηρίζουν και χειρίζονται. Έχει χρήση σε πάρα πολλές εφαρμογές, κυρίως σε εφαρμογές που υπάρχει απαίτηση για απασχόληση μικρού μεγέθους μνήμης του τερματικού στο οποίο εκτελείται η εφαρμογή, όπως τα smartphones, PDAs, tablets, κ.ά. Συνεπώς, εφ’ όσον και στη δική μας εφαρμογή υπήρχε αυτή η λειτουργική απαίτηση κρίθηκε απαραίτητο να ακολουθηθεί αυτή η αρχιτεκτονική και το ανάλογο εργαλείο.

Ένα ακόμα κριτήριο που ικανοποιείται είναι η ανεξαρτησία της εφαρμογής από κάθε είδους εξυπηρετητή server βάσης δεδομένων. Όπως αναλύθηκε στο κεφάλαιο αυτό, η βιβλιοθήκη SQLite μπορεί να λειτουργήσει άψογα ως μηχανή βάσης δεδομένων χρησιμοποιώντας απλά ένα αρχείο τύπου .db. Συνέπεια αυτού είναι η γρήγορη απόκριση στα ερωτήματα sql που εκτελούνται από και προς τη βάση αυτή.

Φυσικά δεν θα πρέπει να παραβλεφθεί το γεγονός ότι η συγκεκριμένη βιβλιοθήκη είναι αποτέλεσμα ανοιχτού λογισμικού, πράγμα που σημαίνει ότι για την χρήση της δεν απαιτείται κάποιο οικονομικό κόστος, ούτε συγκεκριμένη άδεια. Βεβαίως, κάποιοι κακόβουλοι θα σχολίαζαν ως αρνητικό στοιχείου του παραπάνω ότι δεν υπάρχει η ανάλογη υποστήριξη που παρέχεται από εταιρίες ανάπτυξης λογισμικού. Ωστόσο, στην περίπτωση της βιβλιοθήκης αυτής δεν ισχύει το προαναφερθέν, αφού υπάρχει μεγάλη ομάδα ανάπτυξης λογισμικού που ενημερώνουν και βελτιώνουν το συγκεκριμένο project. Επίσης, η χρήση της σε πολλά άλλα μικρά και μεγάλα project ενισχύουν την ανάγκη για συντήρηση και περαιτέρω ανάπτυξης της βιβλιοθήκης αυτής.

ΚΕΦΑΛΑΙΟ 3: Ενημέρωση και διαχείριση της τοπικής βάσης

Στο προηγούμενο κεφάλαιο αναλύσαμε τον τρόπο αποθήκευσης των δεδομένων που χρησιμοποιούνται για την στατιστική ανάλυση και μελέτη που διεκπεραιώνει το πρόγραμμα. Αναφερθήκαμε στη δημιουργία τοπικής βάσης με χρήση της βιβλιοθήκης SQLite. Σε αυτό το κεφάλαιο θα αναλύσουμε τους τρόπους που προσφέρονται στον χρήστη έτσι, ώστε να ενημερώνει αυτή την τοπική βάση όποτε υπάρχει ανάγκη.

Η διαρκής ενημέρωση και συντήρηση της βάσης δεδομένων θεωρείται από τα βασικότερα μέρη της εφαρμογής. Αποτέλεσμα αυτού ήταν μεγάλος χρόνος από τη διάρκεια εκπόνησης της πτυχιακής εργασίας να αφιερωθεί στην μελέτη, ανάλυση και υλοποίηση της δημιουργίας, διαχείρισης και συντήρησης της τοπικής βάσης. Έγινε μεγάλη προσπάθεια ώστε να προσφέρεται στον χρήστη όλες εκείνες οι λειτουργίες, οι οποίες θα απλουστεύσουν τις παραπάνω διαδικασίες. Για να γίνει αυτό, η εφαρμογή υποστηρίζει τρεις διαφορετικούς τρόπους ενημέρωσης της βάσης.

ΕΙΣΑΓΩΓΗ

Ο πρώτος τρόπος χρησιμοποιείται για την καταχώρηση στη βάση μικρού πλήθους συνδυασμών, όπως έναν ή δύο. Με αυτόν τον τρόπο δίνεται στον χρήστη η δυνατότητα να δημιουργήσει τον κατάλληλο συνδυασμό αριθμών και να τον καταχωρήσει στην τοπική βάση.

Ο δεύτερος τρόπος που αναπτύχθηκε για να υποστηρίζει την συντήρηση και διαχείριση της τοπικής βάσης είναι η καταχώρηση συνδυασμών που φορτώνονται από κάποιο αρχείο κείμενο, είτε από κάποιο αρχείο excel. Με αυτόν τον τρόπο επιτρέπεται στον χρήστη να κάνει μαζική καταχώρηση συνδυασμών στην βάση απλά με ένα κουμπί.

Τέλος, ο πιο εύχρηστος και έγκυρος τρόπος που χρησιμοποιείται για την ενημέρωση της βάσης είναι η ενημέρωση απευθείας από το επίσημο ιστότοπο του ΟΠΑΠ. Με αυτή την επιλογή εδραιώνεται μια http σύνδεση με τον διακομιστή του ιστότοπου του ΟΠΑΠ, κάνουμε download τα αντίστοιχα αρχεία excel, στα οποία είναι αποθηκευμένα οι συνδυασμοί των κληρώσεων από τα τέλη του 1997 έως και σήμερα. Στη συνέχεια, η εφαρμογή διαβάζει ένα - ένα τα αρχεία αυτά και αποθηκεύει μόνο τους αριθμούς που αναφέρονται στους συνδυασμούς των κληρώσεων. Με τον συγκεκριμένο τρόπο, προσφέρεται στον χρήστη ένας πολύ εύχρηστος και λειτουργικός τρόπος έτσι, ώστε να ενημερώνεται και να συντηρείται η βάση δεδομένων, στην οποία αποθηκεύονται όλοι οι συνδυασμοί των κληρώσεων και πάνω σε αυτούς στηρίζεται η στατιστική μελέτη και ανάλυση των στατιστικών όρων. Παρακάτω θα γίνει μια πιο εκτενέστερη αναφορά και ανάλυση σχετικά με δύο από τους τρόπους ενημέρωσης και συντήρησης της τοπικής βάσης

που προαναφέρθηκαν, οι οποίοι είναι η χρήση κάποιου αρχείου και η απευθείας από τον επίσημο ιστότοπο του ΟΠΑΠ ενημέρωση.

ΥΠΟΕΝΟΤΗΤΑ 3.1: Ενημέρωση της τοπικής βάσης με χρήση αρχείου κείμενο.

Σε αυτή την υποενότητα θα γίνει ανάλυση της διαδικασίας ενημέρωσης της βάσης μέσω χρήσης κάποιου αρχείου κειμένου. Αρχικά δίνεται στον χρήστη η επιλογή μέσω της αντίστοιχης φόρμας διαλόγου, να διαλέξει και να φορτώσει το αρχείο κειμένου από το οποίο θα διαβαστούν οι αριθμοί των συνδυασμών, θα δημιουργηθούν τα κατάλληλα αντικείμενα συνδυασμών και τέλος θα καταχωρηθούν στην βάση.

Αφού διαλέξει ο χρήστης το κατάλληλο αρχείο μέσα από το οποίο θα επιλεγούν οι αντίστοιχοι αριθμοί των συνδυασμών για να καταχωρηθούν στη βάση, ορίζεται και αρχικοποιείται ένα αντικείμενο FileInputStream, το οποίο θα διαβάσει το αρχείο που επέλεξε ο χρήστης. Στη συνέχεια, αρχικοποιείται το αντικείμενο BufferedInputStream το οποίο χρησιμοποιείται για να διαβάζονται τα byte του αρχείου κειμένου ανά buffers. Τέλος ορίζεται και αρχικοποιείται ένα αντικείμενο DataInputStream με παράμετρο το αντικείμενο που αρχικοποιήθηκε πιο πριν, το BufferedInputStream.

Αφού έχουν οριστεί όλα τα παραπάνω, διαβάζεται κάθε μία γραμμή ξεχωριστά για να γίνει η διαλογή των αριθμών των συνδυασμών. Για να ξεχωρίσουν οι αριθμοί μεταξύ τους χρησιμοποιείται η μέθοδος split με φίλτρο διαχωρισμού έναν από τους χαρακτήρες ‘_’, ‘,’ , ‘ ‘, ‘;’. Έτσι, το αντίστοιχο αρχείο κειμένου θα πρέπει να έχει τέτοια μορφή που θα υποστηρίζεται από την εφαρμογή. Στο πρόγραμμα ορίζεται ακριβώς τι μορφή πρέπει να έχει το αρχείο έτσι, ώστε να δουλέψει σωστά η συγκεκριμένη λειτουργία. Εκτός των συνδυασμών ορίζεται και η αντίστοιχη ημερομηνία που έχει πραγματοποιηθεί η κλήρωση.

Εφόσον έχουν διαβαστεί οι αριθμοί μιας γραμμής, ορίζεται και αρχικοποιείται ένα αντικείμενο τύπου JokerNumbers, το οποί χρησιμοποιείται για να υποστηρίξει τις λειτουργίες της στατιστικής ανάλυσης των διάφορων στατιστικών όρων, όπως αναλύθηκαν πιο πάνω.

Στη συνέχεια, ο πίνακας με τους αριθμούς και την αντίστοιχη ημερομηνία κάθε συνδυασμού που υπάρχει στο αρχείο κειμένου εμφανίζεται στον χρήστη και ο τελευταίος, αν κρίνει απαραίτητο μπορεί να κάνει κάποιες αλλαγές πάνω στους αριθμούς ή την ημερομηνία της κλήρωσης. Όταν αποφασίσει ότι είναι σωστά τα αποτελέσματα, τότε χρησιμοποιεί το κουμπί καταχώρησης των αριθμών αυτών στη βάση δεδομένων όπως προαναφέρθηκε.

ΥΠΟΕΝΟΤΗΤΑ 3.2: Ενημέρωση της τοπικής βάσης με χρήση αρχείου excel.

Ένας επιπλέον τρόπος ενημέρωσης της βάσης καταχωρώντας μαζικά διάφορους συνδυασμούς αριθμών είναι η χρήση κάποιου αρχείου excel, το οποίο θα πρέπει να είναι τύπου 97 – 2003 κατάληξης .xls.

Για την ανάγνωση του αρχείου αυτού χρησιμοποιήθηκε μια βιβλιοθήκη ελεύθερου λογισμικού, την poi-3.8-beta3-20110606.jar. Η βιβλιοθήκη αυτή αναπτύχθηκε από την Apache POI Project, μία ομάδα ανάπτυξης ελεύθερου λογισμικού. Το αρχικό project αναπτύχθηκε από την Jakarta Project, το οποίο παρείχε βιβλιοθήκες Java για ανάγνωση και εγγραφή σε αρχεία με Microsoft Office μορφή, όπως word, powerpoint και excel. Η τελευταία σταθερή έκδοση που δόθηκε στο κοινό ήταν στις 29 Οκτωβρίου 2010, ενώ η τελευταία beta έκδοση δόθηκε στις 17 Δεκεμβρίου 2011.

Για να χρησιμοποιηθεί η παραπάνω βιβλιοθήκη, αφού ενσωματωθεί στο project το κατάλληλο jar αρχείο στον φάκελο Libraries, ενσωματώνουμε τις παρακάτω εντολές για να μπορέσουμε στον κώδικα να έχουμε πρόσβαση στο interface του αντίστοιχου jar αρχείου:

```
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
```

Στη συνέχεια, αφού έχει επιλεγεί από τον χρήστη το κατάλληλο αρχείο excel, μέσα από το οποίο θα ενημερωθεί η βάση, ορίζουμε και αρχικοποιούμε ένα αντικείμενο HSSFWorkbook, το οποίο θα διαβάσει το αρχείο excel. Έπειτα, για κάθε φύλλο excel στο συγκεκριμένο αρχείο χρησιμοποιούμε ένα αντικείμενο HSSFSheet και διαβάζουμε κάθε ένα φύλλο του αρχείου ξεχωριστά με τη βοήθεια της μεθόδου του αντικειμένου HSSFSheet, getSheetAt(k), όπου k ο αριθμός του φύλλου που διαβάζουμε.

Το αντικείμενο HSSFSheet που ορίστηκε παραπάνω έχει μία μέθοδο, την getPhysicalNumberOfRows(), η οποία διαβάζει τις γραμμές του αρχείου που περιέχουν την επιθυμητή πληροφορία. Οπότε, με μία επαναληπτική μέθοδο, όπως η for – loop μέθοδος, έχουμε τη δυνατότητα να διαβάσουμε όλες τις γραμμές του φύλλου. Για κάθε μία γραμμή που διαβάζουμε από το τρέχον φύλλο με την χρήση της μέθοδο getRow(r), όπου r ο αριθμός της τρέχουσας γραμμής, μπορούμε να διαλέξουμε τα κελιά της γραμμής αυτής και να πάρουμε την τιμή του συγκεκριμένου κελιού. Για να διαβάσουμε την πληροφορία που είναι αποθηκευμένη σε κάποιο κελί μιας γραμμής ορίζουμε και αρχικοποιούμε ένα αντικείμενο τύπου HSSFCell χρησιμοποιώντας την μέθοδο getCell(c), όπου c ο αριθμό του τρέχοντος κελιού, του αντικειμένου HSSFRow.

Με την μέθοδο getCell(c) ουσιαστικά διαλέγουμε κάποιο συγκεκριμένο κελί, το υπ' αριθμόν c. Για να διαβάσουμε τι υπάρχει στο συγκεκριμένο κελί, θα πρέπει να

ξέρουμε τι τύπου πληροφορία υπάρχει στο κελί. Υπάρχουν αρκετές περιπτώσεις, μερικές εκ των οποίων είναι να είναι αριθμητικού τύπου, αλφαριθμητικού τύπου, κάποια φόρμουλα, κ.ά. Για να βρούμε τον τύπο δεδομένων που είναι σε κάποιο κελί το χρησιμοποιούμε την μέθοδο getCellType του αντικειμένου HSSFCell που αντιστοιχεί σε κάποιο συγκεκριμένο κελί.

Αν η πληροφορία που είναι αποθηκευμένη στο κελί είναι αριθμητικού τύπου, τότε η μέθοδος του αντικειμένου HSSFCell, η οποία μπορεί να διαβάσει την πληροφορία στο συγκεκριμένο κελί είναι η getNumericCellValue. Αν η πληροφορία του κελιού είναι αλφαριθμητικού τύπου, τότε η μέθοδος getStringCellValue χρησιμοποιείται για να διαβάσουμε την πληροφορία του κελιού. Αν, όμως στο κελί υπάρχει κάποια φόρμουλα, τότε με την μέθοδο getCellFormula του αντικειμένου HSSFCell μπορούμε να διαβάσουμε την πληροφορία του κελιού.

Αφού διαβάσουμε όλα τα κελιά μιας γραμμής του αρχείου ορίζουμε και αρχικοποιούμε με αυτές τις τιμές ένα αντικείμενο της κλάσης JokerNumbers. Όπως προαναφέρθηκε, η κλάση αυτή περιλαμβάνει όλες εκείνες τις μεθόδους που χρησιμοποιούνται για την λειτουργία της στατιστικής ανάλυσης και μελέτης διάφορων στατιστικών όρων με τον τρόπο που ορίστηκαν και αναλύθηκαν στο πρώτο κεφάλαιο.

ΥΠΟΕΝΟΤΗΤΑ 3.3: Ενημέρωση της τοπικής βάσης δεδομένων απευθείας από τον επίσημο ιστότοπο του ΟΠΑΠ Α.Ε.

Ο πιο λειτουργικός και έγκυρος τρόπος για την ενημέρωση και διαχείριση της τοπικής βάσης της εφαρμογής είναι η απευθείας ενημέρωση των συνδυασμών των κληρώσεων από τον επίσημο ιστότοπο του ΟΠΑΠ Α.Ε. Με την λειτουργία αυτή δίνεται η δυνατότητα στον χρήστη με απλό και γρήγορο τρόπο, απλώς πατώντας ένα κουμπί να συγχρονίσει την τοπική του βάση με τα δεδομένα που είναι αποθηκευμένα στον επίσημο διαδικτυακό τόπο του ΟΠΑΠ Α.Ε.

Για να γίνει κάτι τέτοιο, λόγω της σύνδεσης της εφαρμογής με τον ιστότοπο μέσω διαδικτύου, προφανώς είναι απαραίτητο το τερματικό στο οποίο εκτελείται η εφαρμογή να έχει δυνατότητα πρόσβαση στο διαδίκτυο χωρίς την χρήση κάποιου proxy server.

Η δυνατότητα χρήσης όλων των λειτουργιών για την σύνδεση στο διαδίκτυο και συγκεκριμένα με τον επίσημο διαδικτυακό τόπο του ΟΠΑΠ Α.Ε. είναι απαραίτητο να εισάγουμε την παρακάτω γραμμή κώδικα:

```
import java.net.URL;
```

Στη συνέχεια, ορίζουμε ένα αντικείμενο τύπου URL με παράμετρο την διεύθυνση του ιστότοπου για να κάνουμε download τα αντίστοιχα αρχεία excel με όλες τις πληροφορίες των κληρώσεων και των συνδυασμών. Έπειτα με την μέθοδο openConnection() του αντικειμένου τύπου URL, που έχει αρχικοποιηθεί πιο πάνω,

ανοίγουμε την σύνδεση με τον εξυπηρετητή που φιλοξενεί τον αντίστοιχο ιστότοπο.

Αφού έχει εδραιωθεί η σύνδεση της εφαρμογής με τον επίσημο ιστότοπο του ΟΠΑΠ Α.Ε. όπως περιγράφηκε παραπάνω, μας επιτρέπεται καλώντας τη μέθοδο openStream() του αντίστοιχου αντικειμένου τύπου URL, που ορίστηκε παραπάνω, να διαβάσουμε τα αντίστοιχα αρχεία που κατεβάσουμε από τον εξυπηρετητή.

Δεδομένου ότι έχουν πραγματοποιηθεί όλα τα παραπάνω επιτυχώς, μπορούμε να διαβάσουμε όλη την πληροφορία που είναι αποθηκευμένη στα αρχεία αυτά. Για να διαβάσουμε την πληροφορία έχουν οριστεί και αρχικοποιηθεί δύο αντικείμενα τύπου InputStream.

Σε αυτό το σημείο θα πρέπει να τονίσουμε την διαφορετικότητα στην μορφή τους των αρχείων αυτών που γίνεται download από τον ιστότοπο του ΟΠΑΠ Α.Ε. Επειδή τα αρχεία αυτά τα βρίσκουμε από το διαδίκτυο, ορισμένα εξ αυτών έχουν μορφή html, δηλαδή έχουν μορφή xml οριζόμενα με html tags. Οπότε, το διάβασμα και η άντληση της πληροφορίας από αυτά τα αρχεία ήταν μια ιδιαίτερη περίπτωση που έχρηζε ιδιαίτερης προσοχής και έρευνας.

Θα έπρεπε χρησιμοποιώντας διάφορες κανονικές εκφράσεις (Regular expressions) να γίνει η αυτοματοποιημένη διαλογή των επιθυμητών γραμμών που περιλαμβάνουν την επεξεργάσιμη πληροφορία και στη συνέχεια, αφού γίνει η διαλογή των γραμμών, επιλέγονται τα σωστά πεδία του αρχείου που περιλαμβάνει τους αριθμούς των συνδυασμών και οι ημερομηνίες.

Για να έχουμε πρόσβαση στη χρήση των κανονικών εκφράσεων εισάγουμε στον κώδικα τις παρακάτω γραμμές κώδικα:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

Έτσι, μπορούμε να ορίσουμε και να αρχικοποιήσουμε αντικείμενα τύπου Pattern, τα οποία μέσω της μεθόδου compile() μπορούμε να συσχετίσουμε το αντικείμενο αυτό με κάποιο συγκεκριμένο αλφαριθμητικό, το οποίο θα αναζητείται στις γραμμές του αρχείου έτσι ώστε να επιλεγούν μόνο αυτές οι γραμμές που περιλαμβάνουν το συγκεκριμένο αλφαριθμητικό.

Αφού γίνει η διαλογή των επεξεργάσιμων γραμμών από ολόκληρο το αρχείο, βρίσκουμε τα πεδία κάθε μίας από αυτές τις γραμμές που περιέχουν τους συνδυασμούς των αριθμών, την ημερομηνία και την κλήρωση. Για να γίνει η σωστή διαλογή επίσης χρησιμοποιήθηκαν κανονικές εκφράσεις και διάφοροι μετρητές ώστε να επιλέγονται σε συγκεκριμένες θέσεις σε κάποια γραμμή.

Το τελευταίο βήμα είναι να οριστούν και να αρχικοποιηθούν τα αντίστοιχα αντικείμενα της κλάσης JokerNumbers.java, η οποία περιλαμβάνει όλες τις

απαραίτητες για την στατιστική ανάλυση λειτουργίες, και στη συνέχεια να ενημερωθεί η τοπική βάση με αυτές τις τιμές.

Παρακάτω, παραθέτουμε το κομμάτι κώδικα που χρησιμοποιείται για την εύρεση των αριθμών των συνδυασμών από αρχεία που έχουμε κατεβάσει από τον εξυπηρετητή του ιστότοπου του ΟΠΑΠ Α.Ε., τα οποία έχουν μορφή html.

```
URL url;
InputStream is = null;
InputStream is1 = null;
LinkedList tempList = new LinkedList();
String[] date = new String[3];
DataInputStream dis;
String line;
String finalRegex = "^\\<td align='center'\\>";
String finReg = "\\</td\\>$";
Pattern finalP = Pattern.compile(finalRegex);
Pattern finP = Pattern.compile(finReg);
StringBuffer finalSb = new StringBuffer();
StringBuffer finSb = new StringBuffer();
String regex = "^<td> </td>";
Pattern p = Pattern.compile(regex);
StringBuffer sb = new StringBuffer();
 int nowYear = java.util.Calendar.getInstance().get(java.util.Calendar.YEAR);
 for (int i = 1997; i <= nowYear; i++)
 {
 String numberOfExcel = String.format("http://media.opap.gr/Excel/5104/Joker_%d.xls", i);
 url = new URL(numberOfExcel);
 url.openConnection();
is = url.openStream();  // throws an IOException
is1=url.openStream();
contentReading(is1);
dis = new DataInputStream(new BufferedInputStream(is));
index = 0;
boolean myflag = false;
int row = 0;

while ((line = dis.readLine()) != null)
{
    if (line.toString().equals("<tr>") && index == 0)
        myflag = true;
    else if (line.toString().equals("</tr>") && index < 25)
    {
        index = 0; myflag = false;
        tempList.clear();
    }
    else
        if (myflag)
            index++;
    if (line.toString().equals("</tr>") && index == 26)
    {
        myflag = false;index = 0;row++;
        date = tempList.get(1).toString().split("/");
        StringBuilder dateValue = new StringBuilder();
```

```
dateValue.append(date[2]);
dateValue.append("-");
if (date[1].length() < 2)
   dateValue.append("0");
dateValue.append(date[1]);
dateValue.append("-");
if (date[0].length() < 2)
{
   dateValue.append("0");
}
   dateValue.append(date[0]);
prep.setString(2, dateValue.toString());

JokerNumbers stoixeiaKlirwsisAnaGrammi = new JokerNumbers(rowCounter,
                                    dateValue.toString(),
                                    Integer.parseInt(tempList.get(2).toString()),
                                    Integer.parseInt(tempList.get(3).toString()),
                                    Integer.parseInt(tempList.get(4).toString()),
                                    Integer.parseInt(tempList.get(5).toString()),
                                    Integer.parseInt(tempList.get(6).toString()),
                                    Integer.parseInt(tempList.get(7).toString()));
stoixeiaKlirwsewn.add(stoixeiaKlirwsisAnaGrammi);
for (int j = 0; j < stoixeiaKlirwsisAnaGrammi.getNumbers().length; j++)
   prep.setString(j + 3,String.valueOf(stoixeiaKlirwsisAnaGrammi.getNumbers()[j]));
prep.setString(8, String.valueOf(stoixeiaKlirwsisAnaGrammi.getJoker()));
prep.addBatch();
rowCounter++;
tempList.clear();
      }
      if (myflag && index <= 25 && index >= 1)
      {
         Matcher m = p.matcher(line);
         if (sb.length() != 0) sb.delete(0, sb.length());
         while(m.find())
         {
            m.appendReplacement(sb, "");
         }
         m.appendTail(sb);
         Matcher finM = finP.matcher(sb);
         if (finSb.length() != 0) finSb.delete(0, finSb.length());
         while (finM.find())
         {
            finM.appendReplacement(finSb, "");
         }
         finM.appendTail(finSb);
         Matcher finalM = finalP.matcher(finSb);
         if (finalSb.length() != 0) finalSb.delete(0, finalSb.length());
         while (finalM.find())
         {
            finalM.appendReplacement(finalSb, "");
         }
         finalM.appendTail(finalSb);
         tempList.add(finalSb.toString());
      }
   }
```

```
        }
        connection.setAutoCommit(false);
        prep.executeBatch();
        connection.setAutoCommit(true);
        Statement s = connection.createStatement();
        ResultSet rs = statement.executeQuery("select date from joker order by date;");
        int numberOfKlirwsi = 0;
        String insertDate = "";
        while (rs.next())
        {
            numberOfKlirwsi += 1;
            insertDate = "'" + rs.getString("date") + "'";
          s.executeUpdate("update joker set klirwsi =" + numberOfKlirwsi + " where date = "
+ insertDate + ";");
          }
      rs = statement.executeQuery("select klirwsi, date, n1, n2, n3, n4, n5, nJoker from
joker order by klirwsi");
      while (rs.next())
      {
         for (int i = 0; i < stoixeiaKlirwsewn.size(); i++)
         {
            if (stoixeiaKlirwsewn.get(i).getDate().equals(rs.getString("date")))
           {
               stoixeiaKlirwsewn.get(i).setKlirwsi(rs.getInt("klirwsi"));
               break;
            }
         }
      }
      rs.close();
  }
```

Ωστόσο, υπάρχουν περιπτώσεις που κάποια από τα αρχεία που παίρνουμε από τον εξυπηρετητή που φιλοξενεί τον ιστότοπο του ΟΠΑΠ Α.Ε. έχουν μορφή excel, οπότε αυτά μπορούμε να τα διαβάσουμε και να ορίσουμε - αρχικοποιήσουμε τα αντίστοιχα αντικείμενα της κλάσης JokerNumbers.java με τους αριθμούς που χρησιμοποιούνται στη δημιουργία των συνδυασμών των κληρώσεων όπως περιγράφηκε στην «ΥΠΟΕΝΟΤΗΤΑ 3.2: Ενημέρωση της τοπικής βάσης με χρήση αρχείου excel.».

ΕΠΙΛΟΓΟΣ

Συμπερασματικά, θα μπορούσαμε να αναφέρουμε ότι η εφαρμογή έχει σχεδιαστεί και αναπτυχθεί έτσι, ώστε να μπορεί ο κάθε χρήστης της εφαρμογής, χωρίς να έχει γνώσεις sql commands, με απλά βήματα να ενημερώνεται η τοπική βάση που διατηρεί όλους τους συνδυασμούς των κληρώσεων του παιγνιδιού τύχης «τζόκερ». Επίσης, δίνεται η δυνατότητα στον χρήστη να διαχειρίζεται τη βάση της εφαρμογής του καταχωρώντας νέους συνδυασμούς, διαγράφοντας και αλλάζοντας του παλιούς χωρίς να αναλώνει χρόνο σε πράγματα που τον δυσκολεύουν ή τον καθυστερούν.

ΚΕΦΑΛΑΙΟ 4: Εργαλεία ανάπτυξης λογισμικού.

Η επιλογή εργαλείων που βοηθάνε έναν προγραμματιστή στην ανάπτυξη μιας εφαρμογής είναι αποτέλεσμα πολλών παραγόντων, οι οποίοι θα πρέπει να εξεταστούν προσεκτικά πριν την έναρξη του έργου έτσι, ώστε όσα προβλήματα παρουσιαστούν σχετικά με διαδικαστικά και προγραμματιστικά θέματα να μην οδηγήσουν στην μεγάλη καθυστέρηση ή ακόμα και στην κατάργηση της εφαρμογής. Ανάλογα με τις ανάγκες τις εφαρμογής, τις απαιτήσεις και τις ικανότητες του προγραμματιστή πάνω στην ανάπτυξη κώδικα συνήθως καταλήγουμε σε κάποια εργαλεία χρήσιμα για την αναπαραγωγή του προγράμματος.

Ύστερα από σχετική συζήτηση που έγινε με τον επιβλέποντα καθηγητή μου, κ.Σφέτσο, και αφού έλεγξα όλες τις απαιτούμενες παραμέτρους για την επιλογή της γλώσσας ανάπτυξης του λογισμικού, καταλήξαμε ότι η εφαρμογή θα στηριχθεί στην γλώσσα προγραμματισμού JAVA, προϊόν της εταιρίας πληροφορικής Sun Microsoft και από τις 27 Απριλίου 2010 της εταιρίας λογισμικού Oracle Corporation.

ΕΙΣΑΓΩΓΗ

Στα πλαίσια εκπαίδευσής μου στο Αλεξάνδρειο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Θεσσαλονίκης, η επαφή μου με αυτή την γλώσσα προγραμματισμού ήταν καθοριστική στην επιλογή της για την ανάπτυξη της εφαρμογής. Η επιθυμία μου να ενασχοληθώ περισσότερο με την ανάπτυξη λογισμικού χρησιμοποιώντας ως γλώσσα προγραμματισμού, τη JAVA έπαιξε καθοριστικό παράγοντα αφενός στην επιλογή της συγκεκριμένη πτυχιακής εργασίας και αφετέρου στην επιλογή της συγκεκριμένης γλώσσα προγραμματισμού για την υλοποίηση της εφαρμογής.

ΥΠΟΕΝΟΤΗΤΑ 4.1: Γλώσσα ανάπτυξης αντικειμενοστραφούς προγραμματισμού JAVA

Η Java είναι μία γλώσσα προγραμματισμού που υποστηρίζει τον αντικειμενοστραφή προγραμματισμό. Ένα από τα βασικά πλεονεκτήματα έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λογισμικού συστήματος και πλατφόρμας, κάτι που σημαίνει ότι στην αλλαγή του λειτουργικού συστήματος δεν απαιτείται επαναμεταγλώττιση ή αλλαγή του πηγαίου κώδικα. Αφού γραφεί κάποιο πρόγραμμα σε Java, στη συνέχεια μεταγλωττίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode). Ο κώδικας byte είναι η μορφή που παίρνει ο πηγαίος κώδικας της Java όταν μεταγλωττιστεί. Όταν πρόκειται να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, το Java Vitual Machine που πρέπει να είναι εγκατεστημένο σε αυτό θα αναλάβει να διαβάσει τα αρχεία .class. Στη συνέχεια τα μεταφράζει σε γλώσσα

μηχανής που να υποστηρίζεται από το λειτουργικό σύστημα και τον επεξεργαστή έτσι, ώστε να εκτελεστεί. Ωστόσο, πιο σύγχρονες εφαρμογές εικονικής μηχανής μπορούν να μεταγλωττίζουν εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (εγγενή κώδικα ή native code) με αποτέλεσμα να βελτιώνεται η ταχύτητα. Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java. Πρέπει να σημειωθεί ότι η JVM είναι λογισμικό που εξαρτάται από την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Window, Linux, Unix, Macintosh, κινητά τηλέφωνα, κ.ά. Οτιδήποτε θέλει να κάνει ο προγραμματιστής γίνεται μέσω της εικονικής μηχανής. Αυτό βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη – υπολογιστή. Ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Από την άλλη μεριά, ούτε ο χρήστης μπορεί να κατεβάσει «κακό» κώδικα από το δίκτυο και να τον εκτελέσει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα, όπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως.

Ακόμα μία ιδέα που βρίσκεται πίσω από τη Java είναι η ύπαρξη του συλλέκτη απορριμμάτων (Garbage Collector). Συλλογή απορριμμάτων είναι μία κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο. Αυτή η απελευθέρωση μνήμης είναι αυτόματη και γίνεται μέσω του συλλέκτη απορριμμάτων. Υπεύθυνη για αυτό είναι και πάλι η εικονική μηχανή, η οποία μόλις «καταλάβει» ότι ο σωρός (heap) της μνήμης κοντεύει να γεμίσει ενεργοποιεί το συλλέκτη απορριμμάτων. Έτσι ο προγραμματιστής δε χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για σφάλματα δεικτών. Αυτό είναι ιδιαίτερα σημαντικό, γιατί είναι κοινά τα σφάλματα προγραμμάτων που οφείλονται σε λανθασμένο χειρισμό της μνήμης.

Παρόλο που η εικονική μηχανή προσφέρει όλα αυτά, και όχι μόνο, τα πλεονεκτήματα, η Java αρχικά ήταν πιο αργή σε σχέση με άλλες προγραμματιστικές γλώσσες υψηλού επιπέδου όπως η C και η C++. Εμπειρικές μετρήσεις στο παρελθόν είχαν δείξει ότι η C++ μπορούσε να είναι αρκετές φορές γρηγορότερη από τη Java. Ωστόσο γίνονται προσπάθειες από την Oracle για τη βελτιστοποίηση της εικονικής μηχανής, ενώ υπάρχουν και άλλες υλοποιήσεις της εικονικής μηχανής από διάφορες εταιρίες (όπως της IBM), οι οποίες μπορεί σε κάποια σημεία να προσφέρουν καλύτερα και σε κάποια άλλα χειρότερα αποτελέσματα. Επιπλέον με την καθιέρωση των μεταγλωττιστών JIT (Just In Time), οι οποίοι μετατρέπουν τον κώδικα byte απευθείας σε γλώσσα μηχανής, η διαφορά ταχύτητας από τη C++ έχει μικρύνει κατά πολύ. Οι τελευταίες εκδόσεις του javac με τη χρήση της τεχνολογίας Hot Spot έχουν καταφέρει αξιόλογες

επιδόσεις που πλησιάζουν ή και ξεπερνούν σε μερικές περιπτώσεις τον εγγενή κώδικα.

ΥΠΟΕΝΟΤΗΤΑ 4.2: Η κλάση OverlaidBarChart.java για την δημιουργία γραφικών παραστάσεων χρησιμοποιώντας Java Swing.

Η αναπαράσταση των αποτελεσμάτων κάθε στατιστικής μελέτης συνοδεύεται από κάποιο γράφημα. Συνεπώς, για την καλύτερη οπτική μελέτη των στατιστικών αναλύσεων που πραγματοποιούνται από την εφαρμογή προτιμήθηκε να χρησιμοποιηθεί μια έτοιμη βιβλιοθήκη, η JFreeChart.java.

Η συγκεκριμένη βιβλιοθήκη είναι αποτέλεσμα δουλειάς πολλών ετών μιας ομάδας προγραμματισμών, η οποία παρέχει στον προγραμματιστή να την ενσωματώσει στον κώδικα java που αναπτύσσει για την οπτική αναπαράσταση πολλών και διαφορετικών γραφικών παραστάσεων.

Είναι προϊόν του προγραμματιστή David Gilbert από τον Φεβρουάριο του 2000. Εκ τότε έχουν βγει πολλές συνεχώς εμπλουτισμένες με νέες δυνατότητες, εκδόσεις. Η τελευταία σταθερή έκδοση της βιβλιοθήκης αυτής ήταν τον Απρίλιο του 2009, με την έκδοση 1.0.13. Η βιβλιοθήκη JFreeChart κατανέμεται υπό την αιγίδα της GNU Lesser General Public Licence (LGPL), που σημαίνει ότι είναι ελεύθερο λογισμικό, δηλαδή μπορεί να χρησιμοποιηθεί και να εξελιχθεί από οποιονδήποτε χρήστη λογισμικού Java.

Η αντίστοιχη ομάδα προγραμματιστών που έχει συσταθεί για την ανάπτυξη και υποστήριξη αυτού του project παρέχουν πλούσιο documentation και το API που υπάρχει στον επίσημο ιστότοπό τους, http://www.jfree.org/jfreechart/index.html, επιτρέπουν σε κάθε χρήστη προγραμματιστή να εκμεταλλευτεί στο έπακρο όλες τις δυνατότητες της βιβλιοθήκης έτσι, ώστε να προσαρμόσει τις μεθόδους της βιβλιοθήκης στις ανάγκες του δικού του project.

Κάτι ανάλογο έγινε και στη δική μας περίπτωση. Ουσιαστικά, χρησιμοποιώντας τα αντικείμενα του αντίστοιχου τύπου γραφήματος που μας ενδιαφέρει, όπως και τα χαρακτηριστικά και τις μεθόδους αυτών μας επιτρέπεται να αναπτύξουμε και να δημιουργήσουμε γραφήματα. Παρακάτω, παραθέτουμε το κομμάτι κώδικα Java που χρησιμοποιήθηκε για την δημιουργία γραφημάτων στην αναπαράσταση των στατιστικών αποτελεσμάτων που προκύπτουν ύστερα από τους ανάλογους υπολογισμούς. Όπως προαναφέρθηκε, ο κώδικας που χρησιμοποιείται είναι γραμμένος σε γλώσσα προγραμματισμού Java.

```
import java.awt.Color;
import java.awt.Component;
import java.util.ArrayList;
import javax.swing.JScrollPane;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.AxisLocation;
import org.jfree.chart.axis.CategoryAxis;
import org.jfree.chart.axis.CategoryLabelPositions;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.axis.NumberAxis3D;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.DatasetRenderingOrder;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.BarRenderer;
import org.jfree.chart.renderer.category.CategoryItemRenderer;
import org.jfree.chart.renderer.category.LineAndShapeRenderer;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.ApplicationFrame;

public class OverlaidBarChart extends ApplicationFrame {
   final ChartPanel chartPanel;
   public OverlaidBarChart(final String title, String xLabel, String yLabel, String Secondary,
ArrayList<String[]> data1,
         ArrayList<String[]> data2, JScrollPane jscrollPane) {

      super(title);

      final CategoryDataset dataset1 = createDataset1(data1);

      // create the chart...
      final JFreeChart chart = ChartFactory.createBarChart3D(
         title,  // chart title
         xLabel,        // domain axis label
         yLabel,           // range axis label
         dataset1,          // data
         PlotOrientation.VERTICAL,
         true,             // include legend
         true,
         false
      );

      // set the background color for the chart...
      chart.setBackgroundPaint(new Color(255, 255, 235));
      // get a reference to the plot for further customisation...
      final CategoryPlot plot = chart.getCategoryPlot();
      plot.setDomainAxisLocation(AxisLocation.BOTTOM_OR_LEFT);
      plot.setRangeAxisLocation(AxisLocation.TOP_OR_LEFT);
      final CategoryItemRenderer renderer1 = plot.getRenderer();
      renderer1.setSeriesPaint(0, Color.red);
      renderer1.setSeriesPaint(1, Color.yellow);
      CategoryAxis domainAxis = plot.getDomainAxis();
```

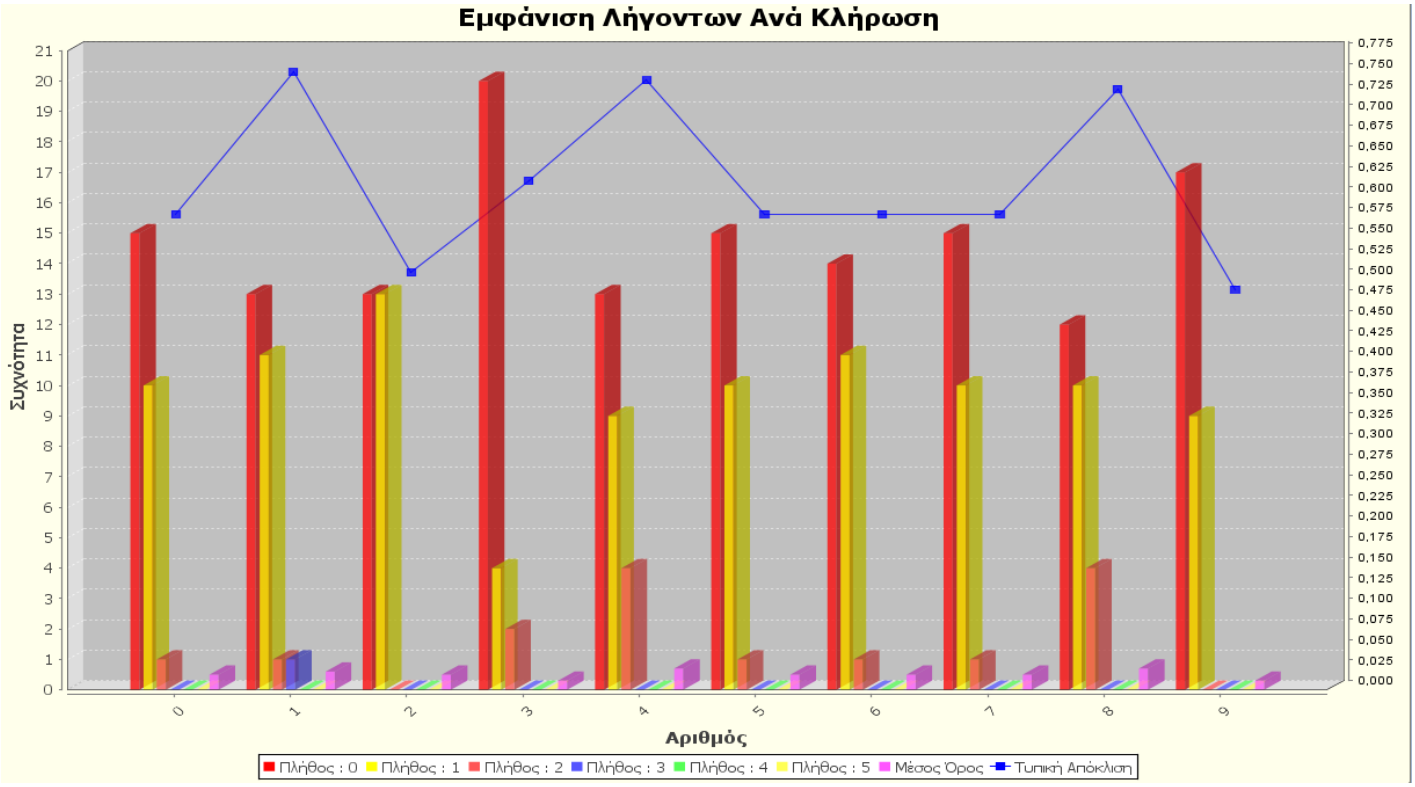
```
        domainAxis.setCategoryLabelPositions(CategoryLabelPositions.UP_45);
        if (data2 != null || !data2.isEmpty())
        {
        final CategoryDataset dataset2 = createDataset2(data2);
        final ValueAxis axis2 = new NumberAxis3D(Secondary);
        plot.setRangeAxis(1, axis2);
        plot.setDataset(1, dataset2);
        plot.mapDatasetToRangeAxis(1, 1);
        final CategoryItemRenderer renderer2 = new LineAndShapeRenderer();
        renderer2.setSeriesPaint(0, Color.blue);
        plot.setRenderer(1, renderer2);

        plot.setDatasetRenderingOrder(DatasetRenderingOrder.REVERSE);
        // OPTIONAL CUSTOMISATION COMPLETED.
        }
        // add the chart to a panel...
        chartPanel = new ChartPanel(chart);
        chartPanel.setPreferredSize(new java.awt.Dimension(500, 300));
    }
    public ChartPanel getScrollPane()
    {
        return chartPanel;
    }
    private CategoryDataset createDataset1(ArrayList<String[]> data) {
        // create the dataset...
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        for (int i = 0; i < data.size(); i++)
            dataset.addValue(Double.parseDouble(data.get(i)[0]), data.get(i)[1], data.get(i)[2]);
        return dataset;
    }
    private CategoryDataset createDataset2(ArrayList<String[]> data) {
        // create the dataset...
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        for (int i = 0; i < data.size(); i++)
            dataset.addValue(Double.parseDouble(data.get(i)[0]), data.get(i)[1], data.get(i)[2]);
            return dataset;

    }
}
```

Για να δημιουργηθεί το αντίστοιχο γράφημα, στο οποίο εμφανίζεται ο μέσος όρος και τυπική απόκλιση των συχνοτήτων εμφάνισης ή καθυστέρησης του αντίστοιχου στατιστικού όρου θα πρέπει να φτιαχτούν δύο διαφορετικά dataset, ένα για τον μέσο όρο και ένα για την τυπική απόκλιση. Η δημιουργία ενός dataset γίνεται από την μέθοδο, CategoryDataset dataset1 = createDataset1(data1), όπου για παράμετρο της συνάρτησης θέτουμε μία δομή δεδομένων τύπου ArrayList<String[]>, που περιλαμβάνει τα δεδομένα που θέλουμε να εμφανίζονται στο γράφημα που φτιάχνουμε. Τα αποτελέσματα του πρώτου dataset θα εμφανίζονται σαν κάθετες μπάρες, το ύψος των οποίων θα αντιστοιχεί στον μέσο όρο των τιμών των μεταβλητών ως προς τις οποίες γίνεται η μελέτη, ενώ τα

αποτελέσματα του δεύτερου dataset θα εμφανίζονται σαν τετράγωνες κουκίδες που αντιστοιχούν στην τυπική απόκλιση των τιμών των μεταβλητών αυτών.

Αφού οριστούν τα dataset με τις πληροφορίες που θέλουμε να φαίνεται στα γραφήματα, δημιουργούμε ένα αντικείμενο τύπου JFreeChart, με την εξής εντολή: chart = ChartFactory.createBarChart3D(). Στις παραμέτρους της μεθόδου createBarChart3D() καταχωρούμε τον τίτλο του γραφήματος, τις διαστάσεις μήκος και πλάτος του πλαισίου, στο οποίο θα εμφανίζεται το γράφημα, καθώς και τα label που εμφανίζονται σε κάθε άξονα. Φυσικά στέλνουμε παραμετρικά στη μέθοδο αυτή το dataset με την επιθυμητή πληροφορία. Ακόμα μπορούμε να ορίσουμε και κάποια άλλα χαρακτηριστικά του γραφήματος, όπως η διεύθυνση πάνω στην οποία θα αναπτύσσονται οι μπάρες. Τέλος, αρχικοποιούμε και κάποιες άλλες παραμέτρους, όπως το χρώμα στο background, τι μορφή θα έχουν οι μπάρες και ποιο χρώμα θα χρησιμοποιηθεί για κάθε είδους πληροφορία που αναπαρίσταται στο γράφημα αυτό.

Οι κανόνες σχεδιασμού λειτουργικών εφαρμογών απαιτούν η πληροφορία να εμφανίζεται στον χρήστη όσο τον δυνατόν ξεκάθαρη, χωρίς να χρειάζεται περεταίρω επεξήγηση του γραφήματος. Άρα θα πρέπει οι αντίστοιχες μπάρες της αναπαράστασης του μέσου όρου και της τυπικής απόκλισης να είναι με διαφορετικά χρώματα έτσι, ώστε να διαχωρίζεται με την αίσθηση της όρασης και μόνο η πληροφορία που αντιστοιχεί σε κάθε τιμής της μεταβλητής ως προς την οποία γίνεται η στατιστική μελέτη.



Εικόνα 2. Διάγραμμα συχνοτήτων και τυπική απόκλισης.

ΕΠΙΛΟΓΟΣ

Εν κατακλείδι, για να υλοποιηθεί ένα project όσο το δυνατόν γρηγορότερα και με λιγότερες «τρύπες» και προβλήματα στον κώδικα θα πρέπει να γίνει αρχικά καταγραφή των απαιτήσεων του πελάτη – χρήστη, όπως και του συστήματος. Στη συνέχεια να γίνει η αντίστοιχη ανάλυση αυτών των απαιτήσεων και στο τέλος να προχωρήσουμε στον σχεδιασμό και υλοποίηση της εφαρμογής.

Ωστόσο καθοριστικός παράγοντας στα παραπάνω παίζει η επιλογή των εργαλείων που θα χρησιμοποιηθούν για την ανάπτυξη του κώδικα και όλων των παρεμφερών συστατικών που απαιτούνται για την ολοκλήρωσή του. Επειδή, όμως, είναι καλό ως ιδεολογία, και όχι μόνο, να υποστηρίζουμε και να ενισχύουμε ως προγραμματιστική κοινότητα τα προϊόντα ελεύθερου λογισμικού, πιστεύω ότι βασικό προγραμματιστικό εργαλείο για την ανάπτυξη κάποιας εφαρμογής είναι η γλώσσα προγραμματισμού Java.

Η ιστορία της Java ως γλώσσα ανάπτυξης εφαρμογών είναι πολύ πλούσια και μεγάλες εταιρίες ανάπτυξης λογισμικού έχουν ασχοληθεί με την ανάπτυξη και εμπλουτισμό της με νέες τεχνολογίες και ιδέες, με αποτέλεσμα να υπάρχουν αξιόλογα προϊόντα λογισμικού που εκδίδονται υπό το καθεστώς του ελεύθερου λογισμικού. Αυτά, λοιπόν τα προϊόντα αν συνδυαστούν και εξελιχθούν από κάθε έναν προγραμματιστή μπορούν να φανούν αρκετά για την υλοποίηση μεγαλεπήβολων έργων, τα οποία θα υποστηρίζουν πλειάδα δυνατοτήτων και τεχνολογιών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Herb Schilldt: JAVA, Έτοιμες συνταγές, Εκδόσεις: Μ. Γκιούρδας.

Herb Schilldt: Οδηγός της Java 2, Τρίτη έκδοση, Εκδόσεις: Μ. Γκιούρδας.

ΗΛΕΚΤΡΟΝΙΚΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

http://www.anendotos.gr/Tzoker/Tzoker.asp?stID=91

http://docs.oracle.com/javase/tutorial/uiswing/components/index.html

http://www.jfree.org/jfreechart/

http://poi.apache.org/spreadsheet/index.html

http://www.sqlite.org/

http://www.lottomania2000.net/

http://www.lotterypowerpicks.com/tx6_hotcold.htm

http://netbeans.org/

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Το βασικό μέρος της πτυχιακής μου εργασίας αποτελεί η εφαρμογή στατιστικής ανάλυσης εξαγόμενων δεδομένων από παιχνίδια τύχης, όπως το «τζόκερ» του ΟΠΑΠ και η δημιουργία συνδυασμών χρησιμοποιώντας βασικούς όρους και ομάδες όρων. Μέσα από την στατιστική ανάλυση που πραγματοποιεί η εφαρμογή και παρουσιάζεται στον χρήστη μέσω πινάκων και κατάλληλων γραφημάτων, προκύπτουν κάποια αποτελέσματα σχετικά με τον μέσο όρο και την τυπική απόκλιση των στατιστικών όρων που μπορεί να επιλέξει ο χρήστης για την στατιστική ανάλυση που θα του παρουσιαστεί από το σύστημα. Έτσι, ο χρήστης χρησιμοποιώντας αυτά τα αποτελέσματα για φίλτρα διαλογής αριθμών, μπορεί ουσιαστικά να καταλήξει σε ένα πλήθος αριθμών λιγότερους από το σύνολο των διαθέσιμων αριθμών που προσφέρονται από την διοργανώτρια του παιγνιδιού αρχή. Με αυτό τον τρόπο ο χρήστης καθοδηγείται με χρήση των μαθηματικών και δη, της στατιστικής θεωρίας, σε περιορισμένο αριθμό συνδυασμό των αριθμών που πληρούν τα αποτελέσματα της στατιστικής ανάλυσης που πραγματοποιεί το σύστημα.

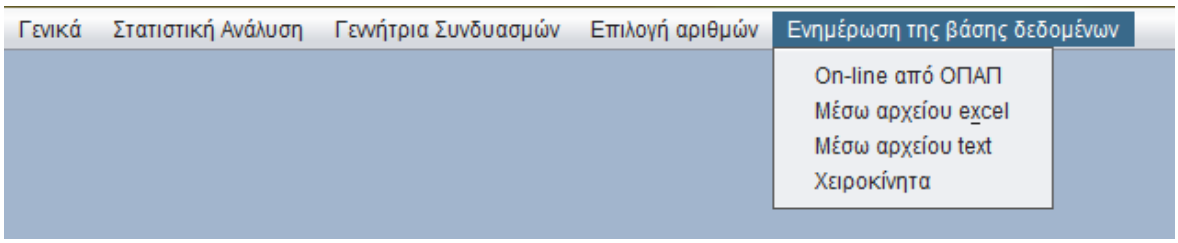
Το σύνολο του συστήματος έχει στηριχθεί στην δυναμική δημιουργία κατάλληλης για το πρόγραμμα τοπικής βάσης δεδομένων, ενημέρωση αυτής με το σύνολο των νικητριών συνδυασμών των προηγούμενων κληρώσεων και στην συντήρησή της. Στην περίπτωση που δεν συμβαίνει το παραπάνω, τα αποτελέσματα που μπορεί να εξαχθούν από το σύστημα μπορεί να είναι απρόβλεπτα. Οπότε χρήσιμη συμβουλή για την σωστή λειτουργία της εφαρμογής και τα βέλτιστα αποτελέσματα είναι να ενημερώνετε τακτικά την τοπική βάση δεδομένων.

Όπως αναφέρθηκε και αναλύθηκε παραπάνω, για ασφαλέστερα αποτελέσματα που προκύπτουν από την στατιστική ανάλυση των συνδυασμών των κληρώσεων, απαιτείται μεγάλος όγκος πληροφορίας. Βασική αρχή στην θεωρία της στατιστικής μελέτης είναι ότι το δείγμα που χρησιμοποιείται για την εξαγωγή πληροφορίας σχετικά με την μεταβλητή ως προς την οποία γίνεται η ανάλυση, θα πρέπει να είναι όσο το δυνατόν μεγαλύτερο, έγκυρο και ομοιογενές. Η φύση του δείγματος, δυστυχώς, είναι τέτοια που δεν μας αφήνει περιθώρια να επηρεάσουμε την ομοιογένεια του δείγματος. Όλες οι κληρώσεις προκύπτουν από κάποια συγκεκριμένη διαδικασία που ακολουθείται από την διοργανώτρια αρχή. Αυτό που μπορούμε να βελτιστοποιήσουμε έτσι, ώστε να εξασφαλίσουμε όσο το δυνατό περισσότερο είναι το μέγεθος και η εγκυρότητα του δείγματος.

Επειδή, όμως, ύστερα από 15 συναπτά χρόνια που λειτουργεί το συγκεκριμένο παιγνίδι τύχης, πάνω στο οποίο στηρίχθηκε ολόκληρη η πτυχιακή, η δεξαμενή της διαθέσιμης πληροφορίας είναι πολύ μεγάλη κρίθηκε ακατάλληλο η χρήση μόνο αρχείων, μέσα από τα οποία θα εξάγονται οι συνδυασμοί. Για αυτό χρησιμοποιήθηκε μια τεχνολογία ανάπτυξης βάσης δεδομένων, η οποία είναι ανεξάρτητη από servers, αφού το μόνο που χρειάζεται για να δουλέψει σωστά είναι ένα σύστημα αρχείων και το κατάλληλο αρχείο jar, το οποίο θα υλοποιήσει την τεχνολογία αυτή. Φυσικά αναφερόμαστε στη βιβλιοθήκη SQLite. Λεπτομέρειες

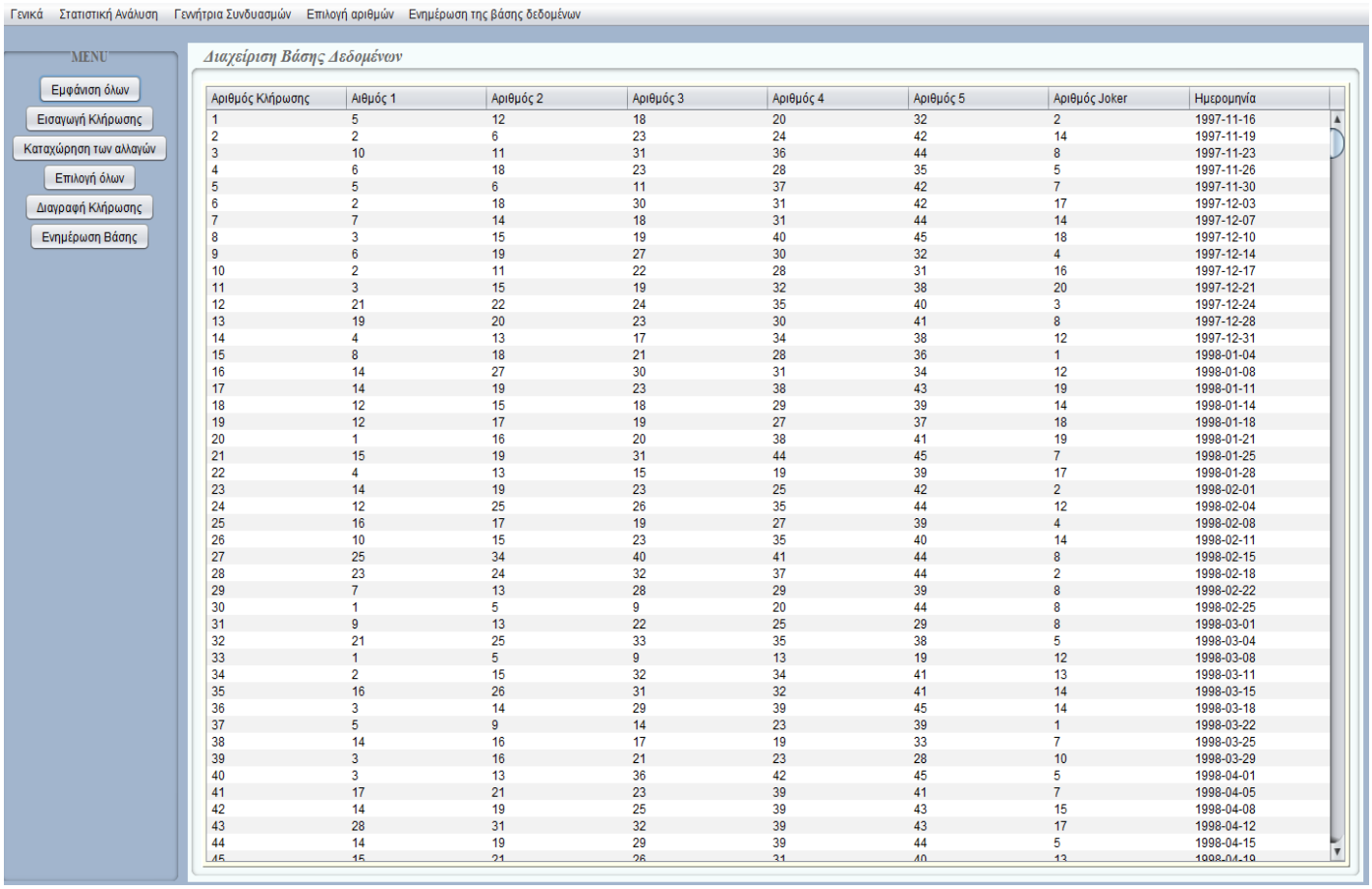
σχετικά με το θέμα έχουν αναφερθεί και αναλυθεί στα προηγούμενα κεφάλαια. Αυτό το οποίο θα πρέπει να τονιστεί είναι η σημαντικότητα της διατήρησης και συντήρησης μιας συνεχώς ενημερωμένης βάσης δεδομένων. Για τον σκοπό αυτό δημιουργήθηκαν διάφορες επιλογές. Όπως βλέπουμε και στην παρακάτω εικόνα, για την ενημέρωση της τοπικής βάσης από τον χρήστης υπάρχουν οι εξής επιλογές:

1. Απευθείας από τον εξυπηρετητή που φιλοξενεί την ιστοσελίδα του παιγνιδιού,
2. Μέσω κάποιου αρχείου excel με format 2003 – 97, δηλαδή κατάληξης .xls,
3. Μέσω κάποιου αρχείου κειμένου, δηλαδή κατάληξης .doc, .txt, .csv,
4. Χειροκίνητα, δηλαδή να εισάγει ο χρήστης έναν – έναν τους αριθμούς της κλήρωσης.



Εικόνα 3. Επιλογές ενημέρωσης της τοπικής βάσης δεδομένων.

Όλες οι προαναφερθείσες επιλογές, λόγω της σημαντικότητας της λειτουργίας που εκπονούν, αναλύθηκαν κάθε μία ξεχωριστά στα προηγούμενα κεφάλαια. Συνεπώς, σε αυτό το σημείο, απλώς θα επιδείξουμε την φόρμα στην οποία υπάρχουν οι επιλογές που παρέχονται στον χρήστη για να κάνει όποιες αλλαγές πιστεύει ότι πρέπει να γίνουν στα δεδομένα της βάσης δεδομένων.

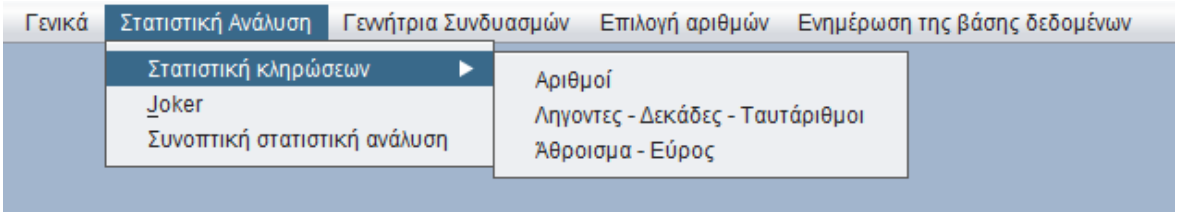


Αριθμός Κλήρωσης	Αιθμός 1	Αριθμός 2	Αριθμός 3	Αριθμός 4	Αριθμός 5	Αριθμός Joker	Ημερομηνία
1	5	12	18	20	32	2	1997-11-16
2	2	6	23	24	42	14	1997-11-19
3	10	11	31	36	44	8	1997-11-23
4	6	18	23	28	35	5	1997-11-26
5	5	6	11	37	42	7	1997-11-30
6	2	18	30	31	42	17	1997-12-03
7	7	14	18	31	44	14	1997-12-07
8	3	15	19	40	45	18	1997-12-10
9	6	19	27	30	32	4	1997-12-14
10	2	11	22	28	31	16	1997-12-17
11	3	15	19	32	38	20	1997-12-21
12	21	22	24	35	40	3	1997-12-24
13	19	20	23	30	41	8	1997-12-28
14	4	13	17	34	38	12	1997-12-31
15	8	18	21	28	36	1	1998-01-04
16	14	27	30	31	34	12	1998-01-08
17	14	19	23	38	43	19	1998-01-11
18	12	15	18	29	39	14	1998-01-14
19	12	17	19	27	37	18	1998-01-18
20	1	16	20	38	41	19	1998-01-21
21	15	19	31	44	45	7	1998-01-25
22	4	13	15	19	39	17	1998-01-28
23	14	19	23	25	42	2	1998-02-01
24	12	25	26	35	44	12	1998-02-04
25	16	17	19	27	39	4	1998-02-08
26	10	15	23	35	40	14	1998-02-11
27	25	34	40	41	44	8	1998-02-15
28	23	24	32	37	44	2	1998-02-18
29	7	13	28	29	39	8	1998-02-22
30	1	5	9	20	44	8	1998-02-25
31	9	13	22	25	29	8	1998-03-01
32	21	25	33	35	38	5	1998-03-04
33	1	5	9	13	19	12	1998-03-08
34	2	15	32	34	41	13	1998-03-11
35	16	26	31	32	41	14	1998-03-15
36	3	14	29	39	45	14	1998-03-18
37	5	9	14	23	39	1	1998-03-22
38	14	16	17	19	33	7	1998-03-25
39	3	16	21	23	28	10	1998-03-29
40	3	13	36	42	45	5	1998-04-01
41	17	21	23	39	41	7	1998-04-05
42	14	19	25	39	43	15	1998-04-08
43	28	31	32	39	43	17	1998-04-12
44	14	19	29	39	44	5	1998-04-15
45	15	21	26	31	40	13	1998-04-19

Εικόνα 4. Οθόνη επιλογών διαχείρισης και ενημέρωσης της τοπικής βάσης δεδομένων.

Όπως φαίνεται και στα αριστερά της φωτογραφίας, οι επιλογές που υποστηρίζει το σύστημα είναι όλες εκείνες οι βασικές λειτουργίες διαχείρισης μιας βάσης δεδομένων, όπως η εισαγωγή νέου συνδυασμού κάποιας κλήρωσης, διαγραφή και ενημέρωση των εγγραφών της τοπικής βάσης. Η πληροφορία που αποθηκεύεται στην βάση και επ' ακολούθως χρησιμοποιείται στην στατιστική μελέτη των όρων, είναι οι πέντε αριθμοί που ορίζουν τον συνδυασμό μαζί με το κληρωθέν αριθμό joker, όπως επίσης και η ημερομηνία κλήρωσης. Βάσει αυτής της πληροφορίας έχουν αναπτυχθεί οι επιλογές που προσφέρονται στον χρήστη για την μελέτη μιας στατιστικής ανάλυσης των αποτελεσμάτων.

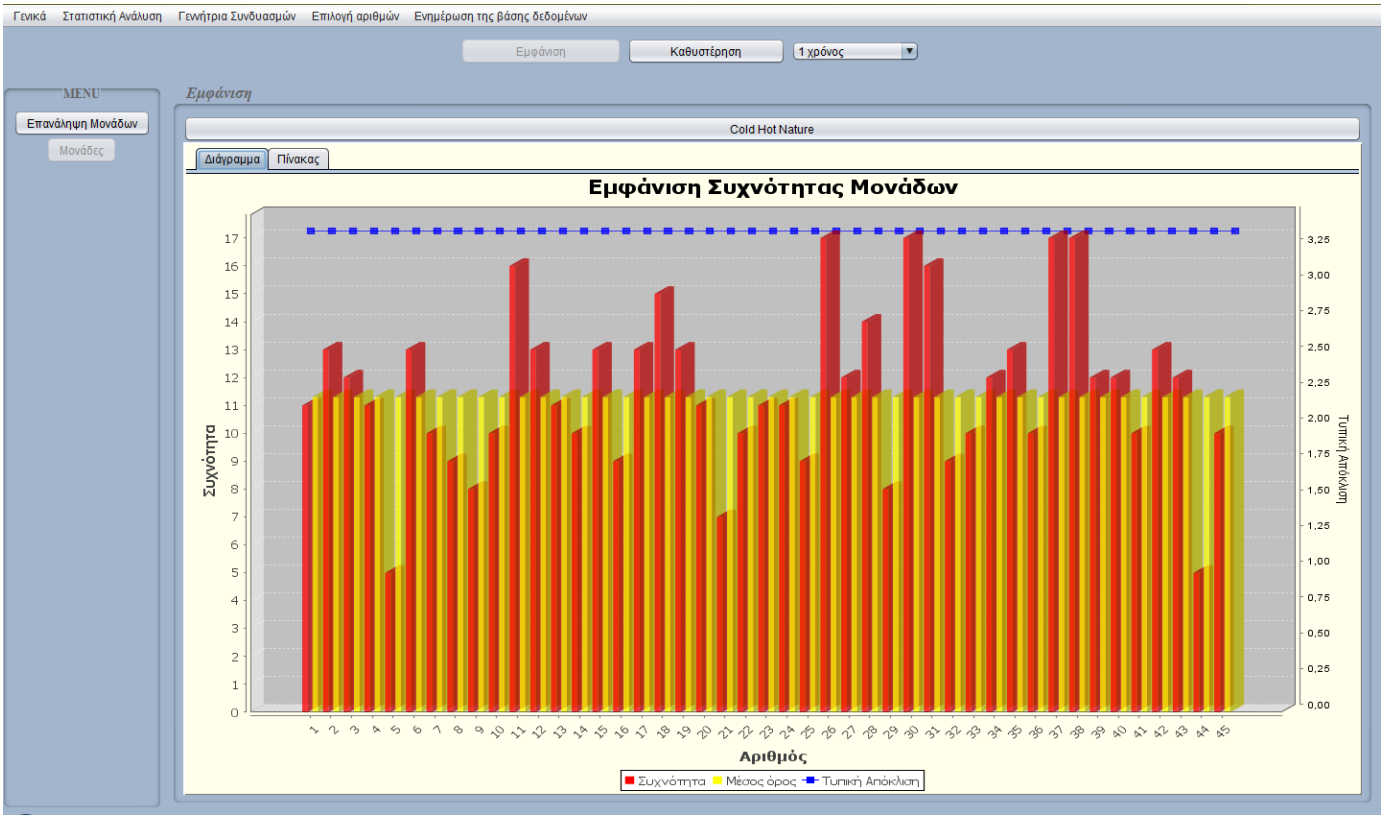
Το πρώτο βήμα έχει γίνει, δηλαδή η πηγή άντλησης πληροφοριών είναι ενημερωμένη με το σύνολο των κληρώσεων του «τζόκερ», οπότε πλέον έχουμε εξασφαλίσει ότι τα αποτελέσματα που θα προκύψουν είναι έγκυρα. Άρα στα επόμενα θα δούμε ποιους στατιστικούς όρους υποστηρίζει το σύστημα και πως εμφανίζονται τα αποτελέσματα στην οθόνη του χρήστη.



Εικόνα 5. Επιλογές ανάλυσης στατιστικών όρων αριθμών και joker

Στην παραπάνω εικόνα φαίνεται ότι η στατιστική ανάλυση χωρίζεται σε δύο βασικά μέρη, στην ανάλυση των στατιστικών όρων που αναφέρονται στους αριθμούς των πεντάδων των συνδυασμών και το δεύτερο στους όρους των αριθμών του joker που έχουν κληρωθεί. Στην τρίτη επιλογή, δίνεται η δυνατότητα στον χρήστη να τρέξει η διαδικασία μελέτης της στατιστικής ανάλυσης των κληρωθέντων αριθμών και joker αριθμών ανά περιόδους, όπως τους τελευταίους τρεις, έξι μήνες, τον τελευταίο χρόνο, δύο χρόνια και στο σύνολό των κληρώσεων από τα τέλη του 1997. Η τελευταία επιλογή ουσιαστικά εμφανίζει συγκεντρωτικά σχεδόν όλους του στατιστικούς όρους που αναλύσαμε στα προηγούμενα με σκοπό να μπορέσει ο χρήστης βλέποντας διάφορα στατιστικά αποτελέσματα ανά περιόδους να συμπεράνει ποιοι αριθμοί έχουν μεγαλύτερες πιθανότητες να κληρωθούν την επόμενη φορά.

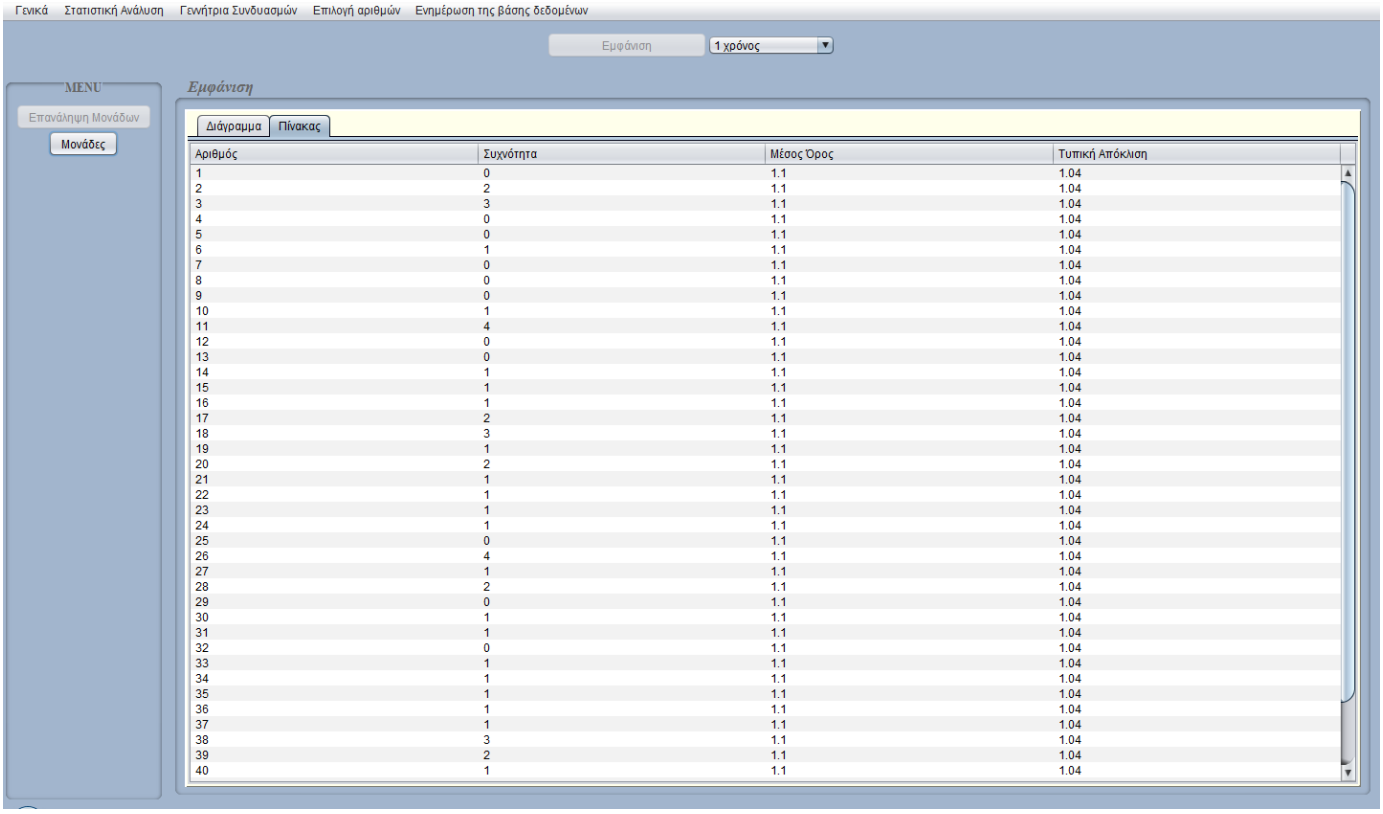
Παρακάτω, θα δείξουμε την οθόνη που χρησιμοποιείται για την αναπαράσταση των αποτελεσμάτων της στατιστικής ανάλυσης για συγκεκριμένο στατιστικό όρο, όπως η συχνότητα εμφάνισης των αριθμών των συνδυασμών.



Εικόνα 6. Διάγραμμα συχνοτήτων των αριθμών των συνδυασμών.

Παρατηρούμε στην προηγούμενη φωτογραφία το πεδίο επιλογής της χρονικής διάρκειας συλλογής πληροφορίας των κληρωθέντων συνδυασμών, η οποία θα αποτελέσει το δείγμα της στατιστικής ανάλυσης. Γίνεται εύκολα αντιληπτό σε αυτό το σημείο η σημασία χρήσης βάσης δεδομένων για την αποθήκευση και διαχείριση της πληροφορίας των κληρώσεων μέσω της βιβλιοθήκης SQLite.

Σε όλους τους στατιστικούς όρους όπως αναφέρθηκαν και αναλύθηκαν προηγουμένως, εμφανίζεται το αντίστοιχο διάγραμμα συχνοτήτων. Συνεπώς, για λογούς εξοικονόμησης χώρου θα εμφανίσουμε μόνο μία ακόμα φωτογραφία, στην οποία θα εμφανίζει ο δεύτερος τρόπος παρουσίασης της πληροφορίας στον χρήστη, μέσω πίνακα.

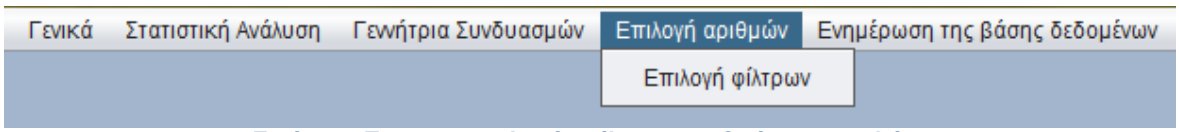


Αριθμός	Συχνότητα	Μέσος Όρος	Τυπική Απόκλιση
1	0	1.1	1.04
2	2	1.1	1.04
3	3	1.1	1.04
4	0	1.1	1.04
5	0	1.1	1.04
6	1	1.1	1.04
7	0	1.1	1.04
8	0	1.1	1.04
9	0	1.1	1.04
10	1	1.1	1.04
11	4	1.1	1.04
12	0	1.1	1.04
13	0	1.1	1.04
14	1	1.1	1.04
15	1	1.1	1.04
16	1	1.1	1.04
17	2	1.1	1.04
18	3	1.1	1.04
19	1	1.1	1.04
20	2	1.1	1.04
21	1	1.1	1.04
22	1	1.1	1.04
23	1	1.1	1.04
24	1	1.1	1.04
25	0	1.1	1.04
26	4	1.1	1.04
27	1	1.1	1.04
28	2	1.1	1.04
29	0	1.1	1.04
30	1	1.1	1.04
31	1	1.1	1.04
32	0	1.1	1.04
33	1	1.1	1.04
34	1	1.1	1.04
35	1	1.1	1.04
36	1	1.1	1.04
37	1	1.1	1.04
38	3	1.1	1.04
39	2	1.1	1.04
40	1	1.1	1.04

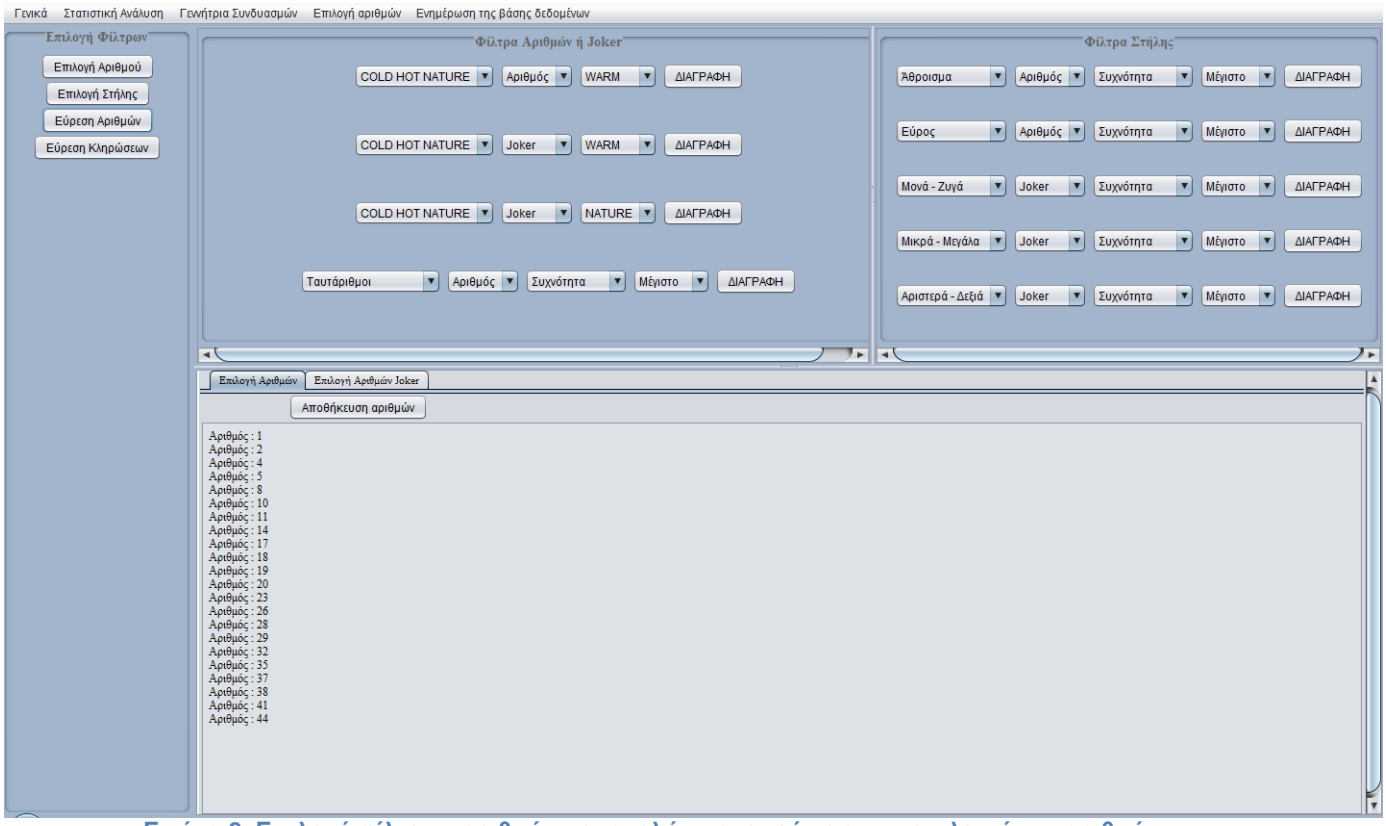
Εικόνα 7. Πίνακας ανάλυσης του στατιστικού όρου "Επανάληψη Μονάδων".

Αφού ο χρήστης πλοηγηθεί ανάμεσα στους στατιστικούς όρους που υποστηρίζει το σύστημα και μελετήσει τα αποτελέσματα αυτά, θα έχει καταλήξει σε κάποιους συγκεκριμένους στατιστικούς όρους, του οποίους θέλει να χρησιμοποιήσει ως φίλτρα στην επιλογή των αριθμών, από τους οποίους θα δημιουργηθούν οι συνδυασμοί. Τα φίλτρα χωρίζονται σε δύο βασικές κατηγορίες, στα φίλτρα επιλογής αριθμών για τους συνδυασμούς και στα φίλτρα επιλογής στήλης, τα οποία φίλτρα χρησιμοποιούνται για να εξεταστεί ο συνδυασμός που προκύπτει από τους αριθμούς που επιλέχθηκαν από την προηγούμενη κατηγορία φίλτρων αν ικανοποιεί τους στατιστικούς όρους της επιλογής του χρήστη ή όχι. Προφανώς, τελικά επιλέγονται οι στήλες που ικανοποιούν τα φίλτρα της δεύτερης κατηγορίας και έχουν δημιουργηθεί από τους αριθμούς που ικανοποιούν τα φίλτρα της

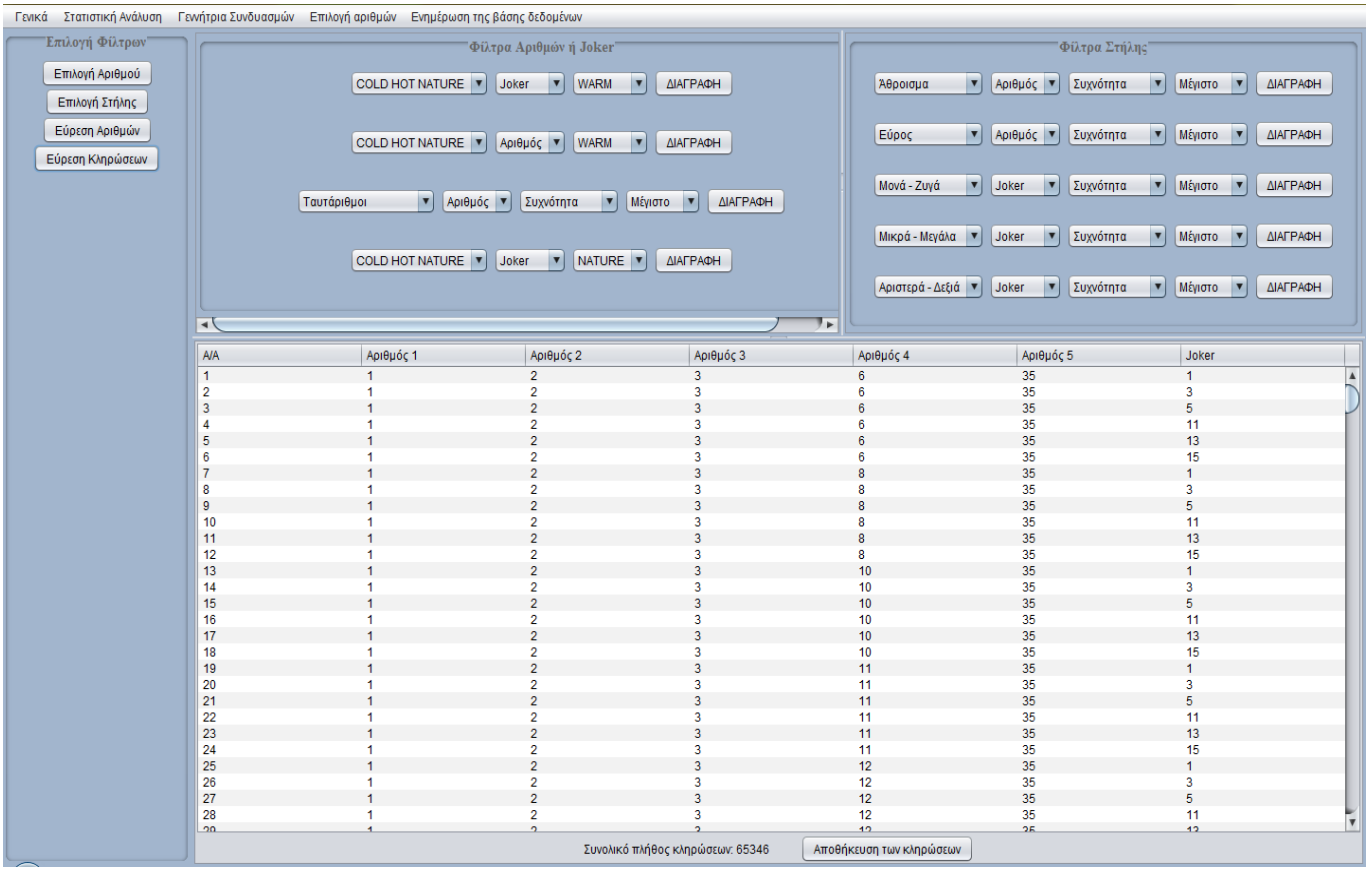
πρώτης κατηγορίας. Παρακάτω εμφανίζονται σε τρεις διαφορετικές φωτογραφίες η επιλογή των φίλτρων και οι αντίστοιχες οθόνες επιλογής των επιθυμητών στατιστικών όρων και εμφάνιση των αποτελεσμάτων.



Εικόνα 8. Το menu επιλογής φίλτρων αριθμών και στηλών.



Εικόνα 9. Επιλογή φίλτρων αριθμών και στηλών και εμφάνιση των επιλεγμένων αριθμών.



Α/Α	Αριθμός 1	Αριθμός 2	Αριθμός 3	Αριθμός 4	Αριθμός 5	Joker
1	1	2	3	6	35	1
2	1	2	3	6	35	3
3	1	2	3	6	35	5
4	1	2	3	6	35	11
5	1	2	3	6	35	13
6	1	2	3	6	35	15
7	1	2	3	8	35	1
8	1	2	3	8	35	3
9	1	2	3	8	35	5
10	1	2	3	8	35	11
11	1	2	3	8	35	13
12	1	2	3	8	35	15
13	1	2	3	10	35	1
14	1	2	3	10	35	3
15	1	2	3	10	35	5
16	1	2	3	10	35	11
17	1	2	3	10	35	13
18	1	2	3	10	35	15
19	1	2	3	11	35	1
20	1	2	3	11	35	3
21	1	2	3	11	35	5
22	1	2	3	11	35	11
23	1	2	3	11	35	13
24	1	2	3	11	35	15
25	1	2	3	12	35	1
26	1	2	3	12	35	3
27	1	2	3	12	35	5
28	1	2	3	12	35	11

Εικόνα 10. Πίνακας συνδυασμών βάσει των στατιστικών όρων.

Μία ακόμα, τελευταία λειτουργία που έχει αναπτυχθεί στην εφαρμογή αυτή είναι ο ορισμός βασικών όρων και ομάδων βασικών όρων, από τους οποίους θα δημιουργηθούν οι συνδυασμοί.

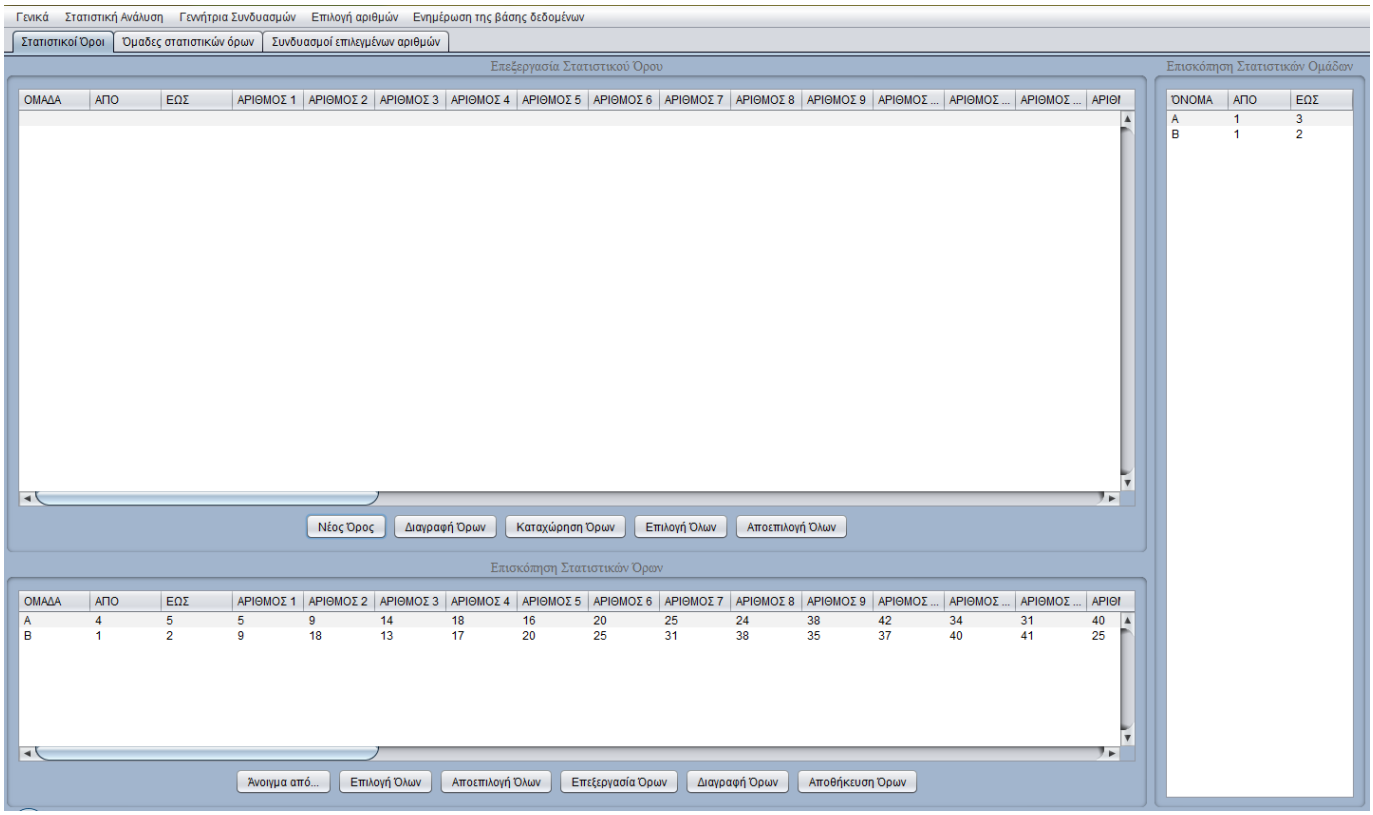
Με την λέξη “βασικός όρος” εννοούμε ένα πλήθος αριθμών από το σύνολο {1, 2, 3, …, 45}, από τους οποίους θα δημιουργηθούν οι συνδυασμοί. Ωστόσο, για να είναι καλώς ορισμένος ο βασικός όρος, θα πρέπει να οριστούν τα όρια του πλήθους των αριθμών του όρου που θέλουμε να βρίσκονται σε κάποιον συνδυασμό έτσι, ώστε ο συνδυασμός αυτός να είναι αποδεκτός. Ακόμα, θα πρέπει να καταχωρήσουμε το όνομα της ομάδας στην οποία ανήκει ο όρος αυτός. Έτσι, παραδείγματος χάριν, ένα αποδεκτός βασικός όρος θα ήταν αν είχαμε ορίσει ότι ο συνδυασμό που έχει δημιουργηθεί περιέχει από 4 έως 5 αριθμούς από τη λίστα των αριθμών που βρίσκεται στον συγκεκριμένο βασικό όρο. Αν το πλήθος των αριθμών που ανήκουν στον όρο και βρίσκονται σε κάποιον συνδυασμό είναι μικρότερος του τέσσερα, τότε αυτός ο συνδυασμός είναι απαράδεκτος.

Ακόμα, υπάρχει δυνατότητα για τον χρήστη να συμπληρώσει ομάδες βασικών όρων. Μια ομάδα βασικών όρων είναι καλώς ορισμένη αν έχει δοθεί το όνομά της και έχουν οριστεί τα όρια μέσα στα οποία θα πρέπει να κινείται το πλήθος των ομάδων από τις οποίες έχουν δημιουργηθεί οι συνδυασμοί. Έστω, λοιπόν, ότι

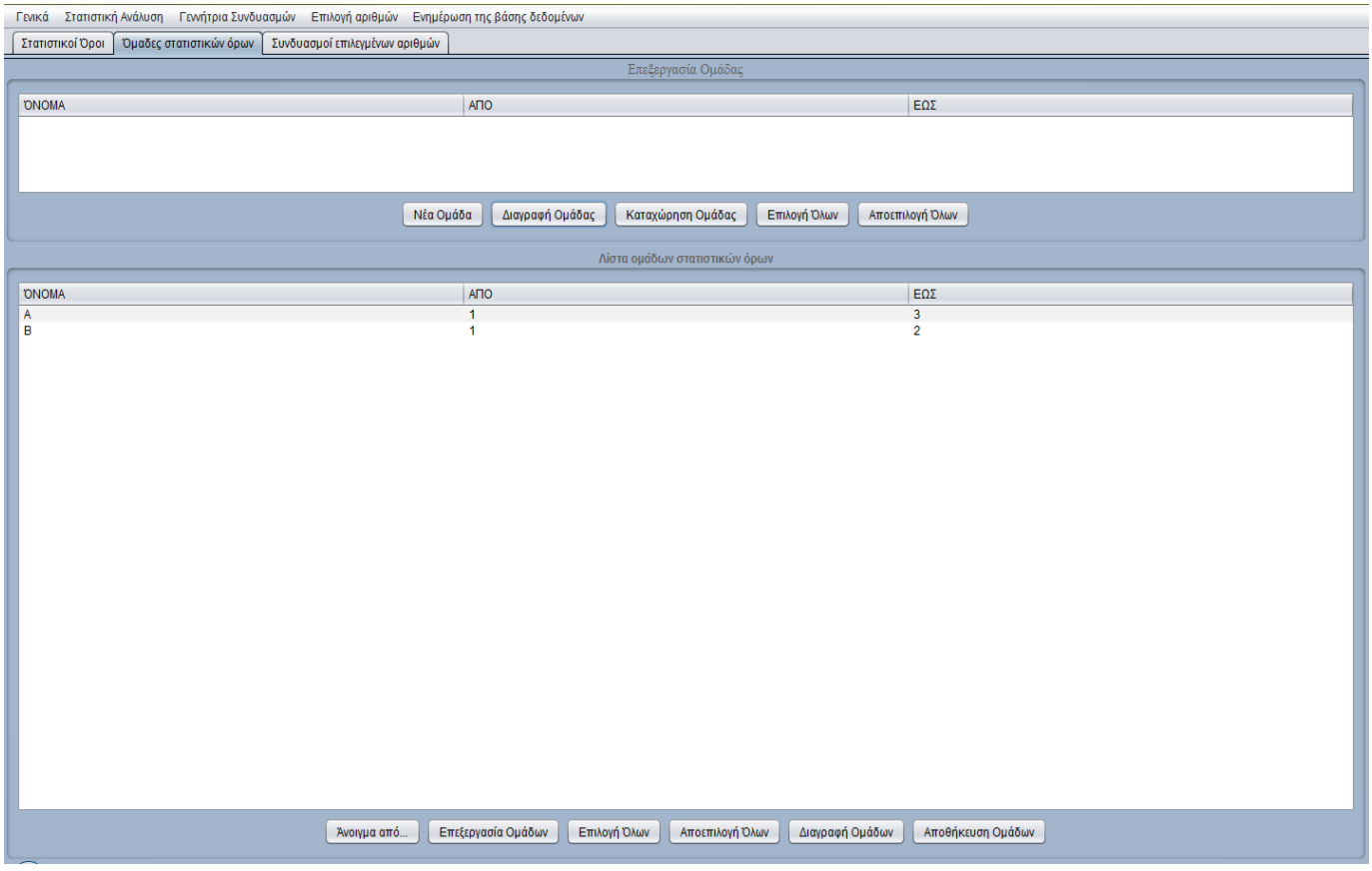
έχουν επιλεγεί οι συνδυασμοί αριθμών, οι οποίοι είναι τέτοιοι ώστε να ικανοποιούν τρεις βασικούς όρους της ομάδας όρων “Α” και μία από την ομάδα όρων “Β”. Τότε αν έχουμε ορίσει δύο ομάδες όρων, την Α με όρια από 1 έως 2 και την ομάδα “Β” με όρια από 1 έως 1, θα επιλεγούν μόνο οι συνδυασμοί που ικανοποιούν την “Β” ομάδα, αφού οι βασικοί όροι που ικανοποιούνται από τους συνδυασμούς των αριθμών είναι τρεις, ενώ η αντίστοιχη ομάδα όρων είναι αποδεκτή μόνο αν το πλήθος των βασικών όρων που ανήκουν στη συγκεκριμένη ομάδα και ικανοποιούνται είναι από 1 βασικός όρος έως 2.

Η γεννήτρια αριθμών αποτελεί ένα εργαλείο με το οποίο ο χρήστης μπορεί να ορίσει το σύνολο των αριθμών μέσα από το οποίο δημιουργούνται οι συνδυασμοί των αριθμών. Υπάρχουν δύο δυνατότητες. Η πρώτη δημιουργεί συνδυασμούς αριθμών μέσα από το σύνολο των αριθμών {1, 2, 3, …, 45}, ενώ η δεύτερη επιλογή επιτρέπει στον χρήστη να διαλέξει συγκεκριμένο υποσύνολο του προηγούμενου συνόλου, μέσα από το οποίο θα δημιουργηθούν οι κατάλληλοι συνδυασμοί.

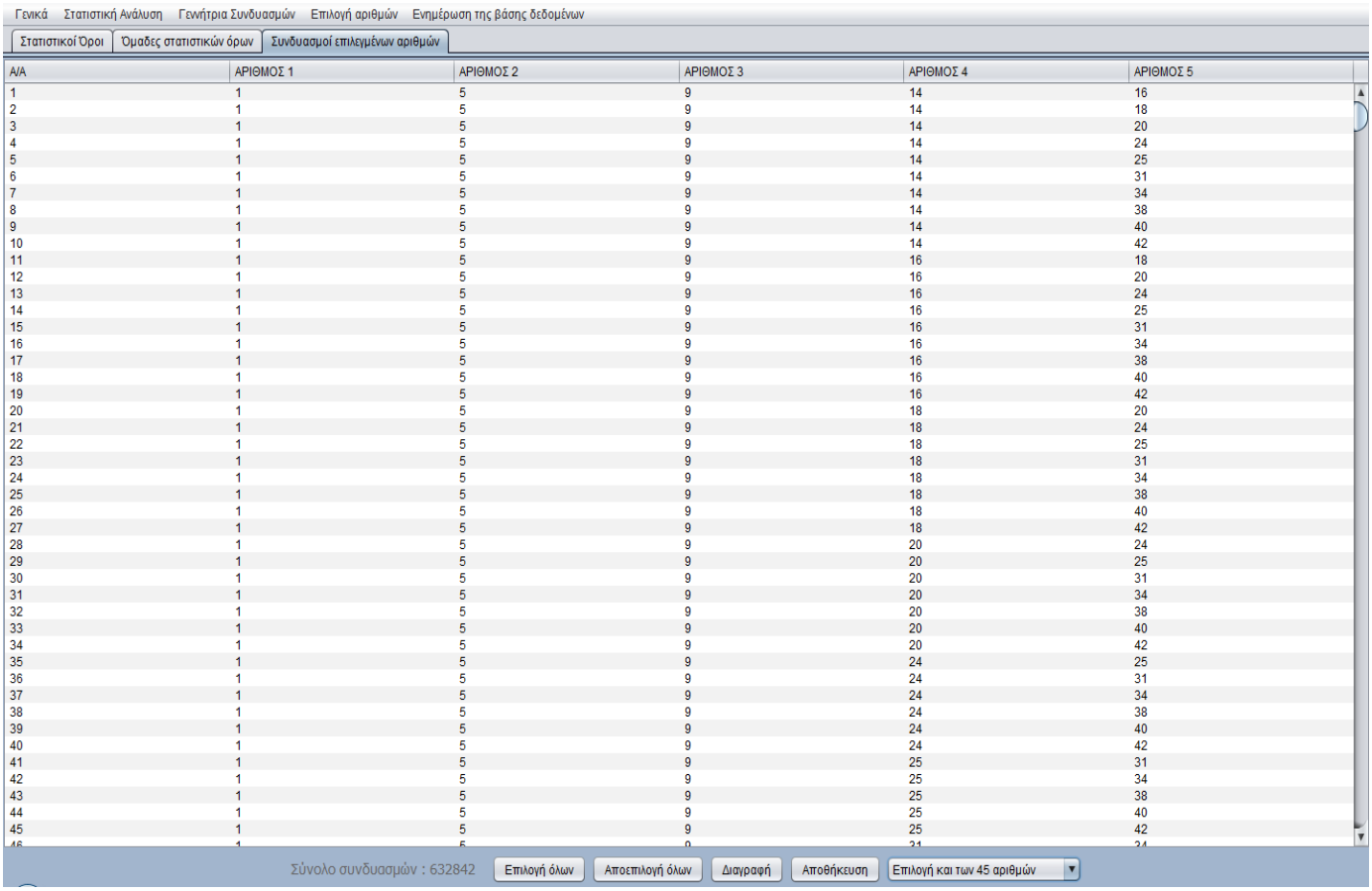
Παρακάτω εμφανίζουμε κάποιες φωτογραφίες των αντίστοιχων οθονών και επιλογών που μπορεί να εκμεταλλευτεί ο χρήστης έτσι, ώστε να δημιουργήσει διάφορους συνδυασμούς σύμφωνα με τις δικές του επιλογές και εμπνεύσεις. Με αυτόν τον τρόπο, ο χρήστης αφού έχει μελετήσει διεξοδικά όλη την στατιστική ανάλυση που πραγματοποιεί το σύστημα, μπορεί να επιλέξει κάποιους αριθμούς και να δημιουργήσει συνδυασμούς ξεκινώντας αποκλειστικά και μόνο επιλέγοντας από αυτούς τους αριθμούς.



Εικόνα 11. Εισαγωγή και επεξεργασία βασικών όρων.

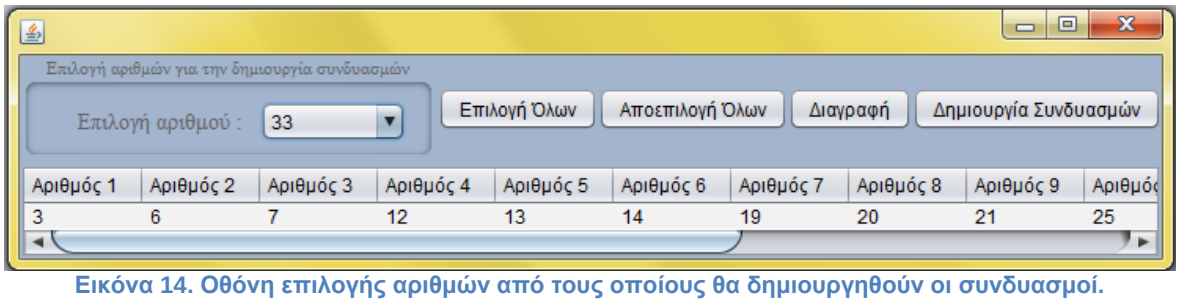


Εικόνα 12. Εισαγωγή και επεξεργασία ομάδων βασικών όρων.



Α/Α	ΑΡΙΘΜΟΣ 1	ΑΡΙΘΜΟΣ 2	ΑΡΙΘΜΟΣ 3	ΑΡΙΘΜΟΣ 4	ΑΡΙΘΜΟΣ 5
1	1	5	9	14	16
2	1	5	9	14	18
3	1	5	9	14	20
4	1	5	9	14	24
5	1	5	9	14	25
6	1	5	9	14	31
7	1	5	9	14	34
8	1	5	9	14	38
9	1	5	9	14	40
10	1	5	9	14	42
11	1	5	9	16	18
12	1	5	9	16	20
13	1	5	9	16	24
14	1	5	9	16	25
15	1	5	9	16	31
16	1	5	9	16	34
17	1	5	9	16	38
18	1	5	9	16	40
19	1	5	9	16	42
20	1	5	9	18	20
21	1	5	9	18	24
22	1	5	9	18	25
23	1	5	9	18	31
24	1	5	9	18	34
25	1	5	9	18	38
26	1	5	9	18	40
27	1	5	9	18	42
28	1	5	9	20	24
29	1	5	9	20	25
30	1	5	9	20	31
31	1	5	9	20	34
32	1	5	9	20	38
33	1	5	9	20	40
34	1	5	9	20	42
35	1	5	9	24	25
36	1	5	9	24	31
37	1	5	9	24	34
38	1	5	9	24	38
39	1	5	9	24	40
40	1	5	9	24	42
41	1	5	9	25	31
42	1	5	9	25	34
43	1	5	9	25	38
44	1	5	9	25	40
45	1	5	9	25	42

Εικόνα 13. Δημιουργία και διαχείριση συνδυασμών με βάσει βασικών όρων και ομάδων.



Εικόνα 14. Οθόνη επιλογής αριθμών από τους οποίους θα δημιουργηθούν οι συνδυασμοί.