**Αποτελέσματα των παιγνίων**

**της ΟΠΑΠ Α.Ε.**

**ΕΑΠ – ΠΛΗ 24**

**3η ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ 2021-2022**

ΠΕΡΙΕΧΟΜΕΝΑ

[1 ΕΙΣΑΓΩΓΗ 3](#_Toc97672203)

[1.1 Περιγραφή Της Εφαρμογής 3](#_Toc97672204)

[1.2 Στόχος του έργου 4](#_Toc97672205)

[1.3 Παραδοχές 5](#_Toc97672206)

[1.4 Χαρακτηριστικά Και Λειτουργίες Της Εφαρμογής 6](#_Toc97672207)

[2 ΥΛΟΠΟΙΗΣΗ ΕΡΓΟΥ 8](#_Toc97672216)

[2.1 Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java 8](#_Toc97672217)

[2.2 Δημιουργία GUI Εφαρμογής 36](#_Toc97672218)

[2.3 Παρουσίαση Στατιστικών Στοιχείων και Γραφημάτων 73](#_Toc97672219)

[3 ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ 88](#_Toc97672221)

[4 ΑΝΑΦΟΡΕΣ 102](#_Toc97672230)

# ΕΙΣΑΓΩΓΗ

## Περιγραφή Της Εφαρμογής

Η παρούσα εργασία αφορά στην ανάλυση, σχεδίαση και υλοποίηση εφαρμογής που αντλεί δεδομένα από υπηρεσία ιστού (web service), τα αποθηκεύει σε τοπική βάση δεδομένων, και τα επεξεργάζεται έτσι ώστε να τα παρουσιάσει με εύληπτο τρόπο στον τελικό χρήστη του συστήματος. Τα δεδομένα που διαχειρίζεται η εφαρμογή *TzokerStats* αντιστοιχούν στο γνωστό παιχνίδι ΤΖΟΚΕΡ του ΟΠΑΠ και πιο συγκεκριμένα αφορούν στα αποτελέσματα των κληρώσεων αλλά και στα στατιστικά του παιχνιδιού από το λανσάρισμα του (2000) έως και σήμερα.

Εν κατακλείδι το παρεχόμενο σύστημα θα πρέπει να επιλύει το πρόβλημα της άντλησης και παρουσίασης δεδομένων που αφορούν σε κληρώσεις, τις ιστορικές αλλά και την τρέχουσα, του παιχνιδιού ΤΖΟΚΕΡ του ΟΠΑΠ. Επιπλέον θα πρέπει να έχει την δυνατότητα επέκτασης και στα υπόλοιπα παιχνίδια του ΟΠΑΠ.

## Στόχος του έργου

Αν αποσυνθέσουμε την γενική περιγραφή της λύσης, της προηγούμενης παραγράφου, στις συνιστώσες της προκύπτει ότι:

Αρχικά η παρεχόμενη λύση θα πρέπει να αντλεί δεδομένα από το REST API του ΟΠΑΠ. Η εν λόγω υπηρεσία ιστού περιγράφεται στην ιστοσελίδα <https://www.opap.gr/web-services>. Στην εμβέλεια (scope) του έργου, ανήκουν τα δεδομένα που αφορούν τις κληρώσεις αλλά και τα στατιστικά του παιχνιδιού ΤΖΟΚΕΡ. Ποιο συγκεκριμένα, τα endpoints που επιστρέφουν τα απαιτούμενα δεδομένα είναι τα:

1. [https://api.opap.gr/draws/v3.0/{gameId}/draw-date/{fromDate}/{toDate}](https://api.opap.gr/draws/v3.0/%7bgameId%7d/draw-date/%7bfromDate%7d/%7btoDate%7d)
2. [https://api.opap.gr/draws/v3.0/{gameId}/{drawId}](https://api.opap.gr/draws/v3.0/%7bgameId%7d/%7bdrawId%7d)
3. [https://api.opap.gr/games/v1.0/{gameId}/statistics](https://api.opap.gr/games/v1.0/%7bgameId%7d/statistics)

Με κλήση του (1) αντλούνται  τα αποτελέσματα των κληρώσεων του επιλεγμένου παιχνιδιού για το δοσμένο εύρος ημερομηνιών. Ενώ με κλήση του (2) αντλούνται τα αντίστοιχα δεδομένα για μεμονωμένη κλήρωση. Τέλος με κλήση του (3) επιστρέφονται στατιστικά στοιχεία του παιχνιδιού όπως  πόσο συχνά έχουν εμφανιστεί ή καθυστερήσει να εμφανιστούν οι αριθμοί το Τζόκερ από την αρχή του παιχνιδιού μέχρι σήμερα. Τα κριτήρια αναζήτησης παρέχονται από τον χρήστη κάθε φορά μέσα από το αντίστοιχο GUI της εφαρμογής.

Εν συνεχεία οι συμβολοσειρές JSON (JSON literals) που επιστρέφονται θα μετατρέπονται σε αντικείμενα JAVA μέσω τον οποίων γίνεται η επεξεργασία και παρουσίαση τους, καθώς και η αποθήκευση τους στην σχεσιακή βάση δεδομένων (μέσω του JPA) που στην συγκεκριμένη περίπτωση είναι η Derby.

Επιπλέον η εφαρμογή θα πρέπει να έχει την δυνατότητα να επεξεργάζεται τα δεδομένα που αποθήκευσε στην βάση αλλά και να τα παρουσιάζει στον τελικό χρήστη.

Οι λειτουργίες που προαναφέρθηκαν θα πρέπει να επιτελούνται με εύκολο και αξιόπιστο τρόπο έτσι ώστε να θεωρηθεί ότι η παρεχόμενη λύση καλύπτει τους βασικούς στόχους του έργου και πετυχαίνει υψηλό ποσοστό αποδοχής από την κοινότητα των τελικών χρηστών (user adaptation).

Για να καλυφθούν οι παραπάνω στόχοι αναπτύχθηκε ένα γραφικό περιβάλλον χρήσης (GUI) το οποίο επιτρέπει την εύκολη διαχείριση των λειτουργιών της εφαρμογής. Επιπλέον η διαχείριση των σφαλμάτων, που είναι πιθανό να προκύψουν, έγινε με στόχο την αποφυγή καταστάσεων ασυνέπειάς (inconsistency) στα δεδομένα που χειρίζεται η παρεχόμενη λύση αλλά και την ορθή καθοδήγηση του χρήστη.

## Παραδοχές

Όπως προαναφέρθηκε, κατά την σχεδίαση της εφαρμογής λήφθηκε υπόψιν ότι θα πρέπει δυνητικά να υποστηρίξει όλα τα παιχνίδια που προσφέρονται από τον ΟΠΑΠ. Για το λόγο αυτό ο κωδικός παιχνιδιού (GAME\_ID) αποτυπώνεται σε όλα τα επίπεδα του συστήματος. Από την βάση δεδομένων και τις κλάσεις έως τις κλήσεις προς το API Και το GUI. Εδώ θα πρέπει να σημειωθεί ότι σε περίπτωση επέκταση της εφαρμογής σε νέο παιχνίδι θα πρέπει να προστεθούν και/ή να τροποποιηθούν φόρμες αφού κάθε παιχνίδι έχει διαφορετικούς μηχανισμούς που το διέπουν (game mechanics) και επομένως τα δεδομένα και οι κανόνες του διαφέρουν.

Επιπλέον η ανάπτυξη του λογισμικού έγινε με την παραδοχή ότι θα πρέπει να καλυφθούν αδυναμίες του παρεχόμενου API έτσι ώστε να διευκολυνθεί ο τελικός χρήστης αλλά και να πάρει σωστό τελικό αποτέλεσμα. Για τον ίδιο σκοπό έχουν προστεθεί λειτουργίες στην εφαρμογή όπως η μερική εισαγωγή δεδομένων (partial insert) και η διαγραφή όλων των δεδομένων της βάσης που περιγράφονται παρακάτω.

Τέλος, θεωρήθηκε ότι εργασία αποτελείται από δραστηριότητες που έχουν να κάνουν με τη συγγραφή κώδικα αλλά και την τεκμηρίωση και καταγραφή των δραστηριοτήτων στις αντίστοιχες ενότητες του πρότυπου συγγραφής της εργασίας.

## Χαρακτηριστικά Και Λειτουργίες Της Εφαρμογής

Τα χαρακτηριστικά που διέπουν την εφαρμογή είναι απόρροια των απαιτήσεων της λύσης. Έπεται ότι η εφαρμογή θα πρέπει:

|  |
| --- |
| **R1 – Κυρίως Μενού** |
| Κατά την εκκίνηση της η εφαρμογή θα εμφανίζει το κυρίως παράθυρο. Το τελευταίο περιέχει το βασικό μενού, το οποίο παρέχει εύκολη πλοήγηση στις δυνατότητες που προσφέρει η εφαρμογή. Κάθε επιλογή παρουσιάζει νέα φόρμα μέσα σε ξεχωριστό πλαίσιο της αρχικής οθόνης. Οι παρεχόμενες επιλογές είναι:   1. Διαχείριση δεδομένων ΤΖΟΚΕΡ 2. Προβολή δεδομένων ΤΖΟΚΕΡ 3. Προβολή στατιστικών δεδομένων ΤΖΟΚΕΡ 4. Έξοδος. |
| |  | | --- | | **R2 - Διαχείριση δεδομένων ΤΖΟΚΕΡ** | | Με την συγκεκριμένη επιλογή πρέπει να παρουσιάζεται φόρμα που επιτρέπει στον χρήστη να εκτελέσει τις παρακάτω λειτουργίες:   1. Άντληση και εμφάνιση δεδομένων κληρώσεων Τζόκερ για κάποιον κωδικό κλήρωσης 2. Άντληση και εμφάνιση δεδομένων κληρώσεων Τζόκερ σε εύρος ημερομηνιών 3. Εισαγωγή των επιλεγμένων δεδομένων κληρώσεων Τζόκερ στη βάση δεδομένων (ΒΔ). 4. Διαγραφή των δεδομένων κληρώσεων Τζόκερ από τη ΒΔ. 5. Διαγραφή όλων των δεδομένων κληρώσεων Τζόκερ από την ΒΔ. 6. *Επιπλέον* θα δοθεί η δυνατότητα μερικής εισαγωγής δεδομένων έτσι ώστε να διευκολυνθεί ο χειρισμός της εφαρμογής από τον τελικό χρήστη. | |
| |  | | --- | | **R3 - Οθόνη προβολής δεδομένων ΤΖΟΚΕΡ** | | Αρχικά ο χρήστης θα μπορεί να επιλέγει έτος και εν συνεχεία να βλέπει συνολικά (aggregated) δεδομένα που αφορούν τις κληρώσεις ανά μήνα για το συγκεκριμένο έτος. Αναλυτικότερα, παρέχονται πληροφορίες για τον αριθμό των κληρώσεων τα χρήματα που διανεμήθηκαν καθώς και τον αριθμό των ΤΖΑΚ-ΠΟΤ που πραγματοποιήθηκαν στο επιλεγμένο έτος αλλά και τα διανεμόμενα κέρδη ανά μήνα. Τα τελευταία παρουσιάζονται με γραφικό τρόπο υπό την μορφή bar chart. Σε περίπτωση που για το επιλεγμένο έτος λείπουν πληροφορίες από την βάση, τότε αυτές θα αντλούνται και θα εισάγονται στην βάση προτού παρουσιαστούν τα αποτελέσματα στον χρήστη.  *Επιπλέον*, για να προωθηθεί η αίσθηση (look and feel) του dashboard με κλικ στα αποτελέσματα κάποιου μήνα θα γίνεται ανάλυση (drill down) του μήνα ανά κλήρωση. Η πληροφορία αυτή θα εμφανίζεται σε νέα φόρμα η οποία επιπλέον θα παρέχει την δυνατότητα αλλαγής του έτους αλλά και του μήνα. Με αυτόν τον τρόπο δεν θα απαιτείται από τον χρήστη να μεταβεί πίσω στην αρχική φόρμα, διευκολύνοντας έτσι την ανάλυση των δεδομένων αφού απλοποιείται η πλοήγηση (navigation) στην εφαρμογή. | | **R4 - Προβολή στατιστικών δεδομένων ΤΖΟΚΕΡ και εκτύπωση σε PDF** | | Με την επιλογή “Προβολή Στατιστικών Τζόκερ” και επιλογή του αντίστοιχου υπο-μενού, ο χρήστης θα μπορεί να επιλέγει ένα εύρος ημερομηνιών και πληροφορείται για τα στατιστικά των αριθμών που έχουν κληρωθεί. Η πληροφορία θα εμφανίζεται υπό την μορφή πίνακα και επιπλέον θα δίνεται η δυνατότητα εκτύπωσης του πίνακα σε αρχείο PDF.  Επιπλέον το παράθυρο επιλογής της τοποθεσίας αποθήκευσης του pdf θα πρέπει να παρέχει κάποια βοηθητικά χαρακτηριστικά όπως:  Να επιτρέπει στον χρήστη να ανοίξει το παραγόμενο αρχείο αυτόματα μετά την παραγωγή του pdf στο πρόγραμμα που έχει συσχετιστεί με αυτόν τον τύπο του αρχείου.  Να θυμάται την τελευταία τοποθεσία αποθήκευσης  Να εμφανίζει τα αντίστοιχα μηνύματα σε περίπτωση που το αρχείο υπάρχει ήδη στην τοποθεσία αποθήκευσης και θα πρέπει να αντικατασταθεί. | | **R5 - Προβολή στατιστικών δεδομένων ΤΖΟΚΕΡ σε γραφική μορφή** | | Μέσα από την επιλογή “Προβολή Στατιστικών Τζόκερ” και επιλογή του αντίστοιχου υπο-μενού, η εφαρμογή θα πρέπει να επιτρέπει στον χρήστη να πάρει πληροφορίες σχετικές με την συχνότητα εμφάνισης των αριθμών της κάθε κλήρωσης καθώς και τον μέσο όρο κερδών ανά κατηγορία. Ο χρήστης θα πρέπει και εδώ να μπορεί να επιλέξει το εύρος ημερομηνιών και επιπλέον τα στατιστικά στοιχεία που επιθυμεί. | |

# ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2 - ΥΛΟΠΟΙΗΣΗ ΕΡΓΟΥ

## Διάγραμμα Κλάσεων και Υλοποίηση Κλάσεων σε Java

α) Το πρώτο βήμα ήταν να δημιουργηθούν τα απαραίτητα Entity-Relationship (ER) καθώς και Class diagrams. Τα διαγράμματα αυτά σχεδιάστηκαν με την βοήθεια του CASE Tool Visual Paradigm. Το αντίστοιχο αρχείο που παράχθηκε από το ανωτέρω εργαλείο ανήκει στο σύνολο των παραδοτέων του έργου, άρα ο αναγνώστης μπορεί να ανατρέξει εκεί για περισσότερες λεπτομέρειες.

Βασική αρχιτεκτονική επιλογή ήταν η υλοποίηση της λογική της εφαρμογής στις συνοριακές κλάσεις (πακέτο gui) και όχι της προσέγγισης που ξεχωρίζει τη λογική της εφαρμογής από την παρουσίαση, όπως για παράδειγμα το πρότυπο Model-View-Controller (MVC) ή το Model-View-ViewModel (MVVM).

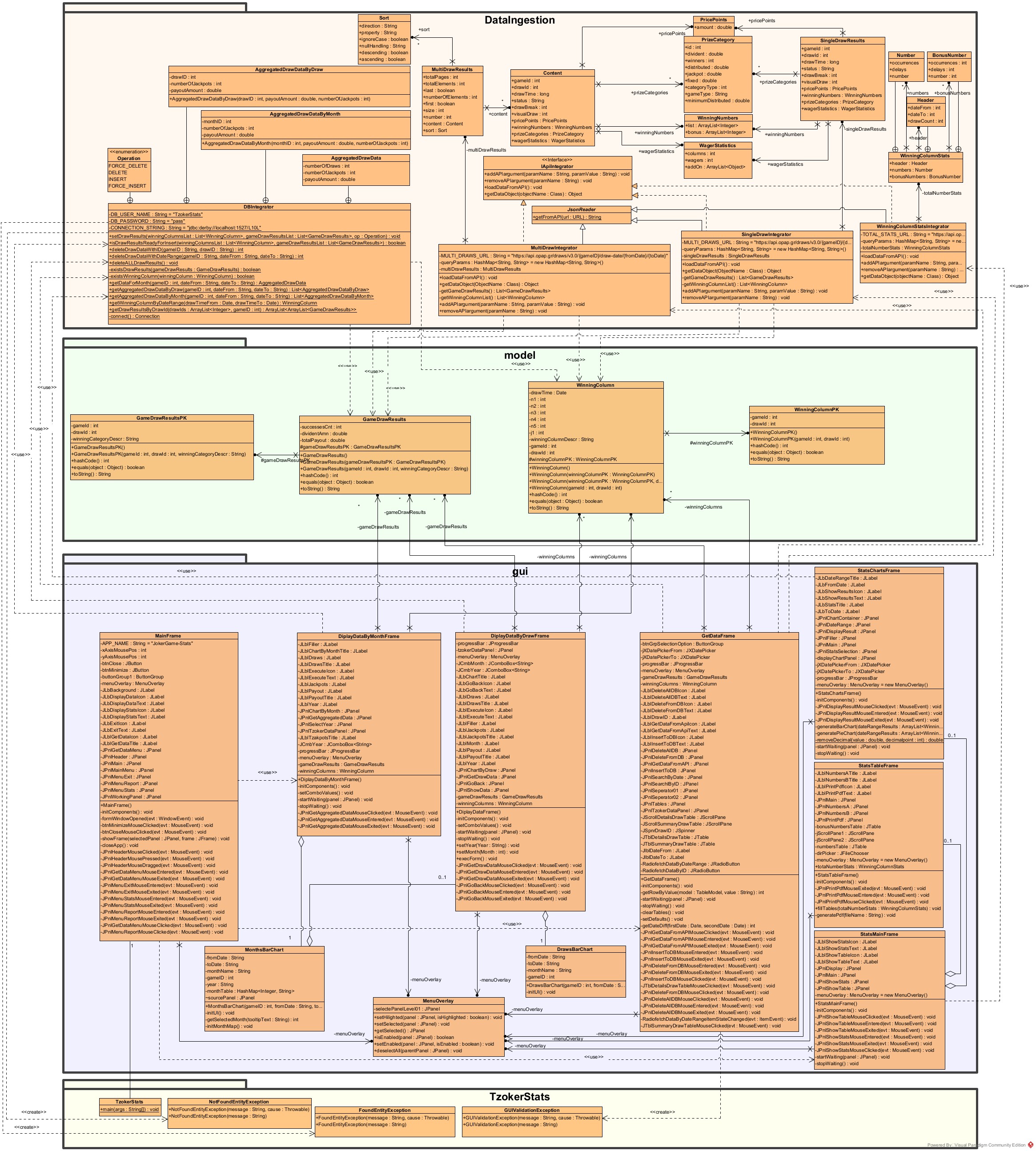
Ακόμη μια αρχιτεκτονική επιλογή που διέπει την εφαρμογή, είναι η ανάθεση των εργασιών που σχετίζονται με API Calls σε εξειδικευμένες κλάσεις. Γι’ αυτό το σκοπό υλοποιήθηκε interface (*IApiIntegrator*) το οποίο ορίζει τις απαραίτητες λειτουργίες που πρέπει να υλοποιεί κάθε κλάση που παρέχει πρόσβαση στο OPAP API. Έτσι δίνεται η δυνατότητα στις κλάσεις πελάτες (gui) να στέλνουν μηνύματα στις κλάσεις που υλοποιούν το interface, δίχως οι τελευταίες να λαμβάνουν υπόψη τον συγκεκριμένο τύπο της εκάστοτε κλάσης (πολυμορφισμός). Επιπλέον η επέκταση της εφαρμογής απλοποιείται, αφού αρκεί να προστεθεί ένας επιπλέον Integrator για κάθε νέο παιχνίδι που πρέπει να υποστηρίξει η εφαρμογή. Με την ίδια λογική έχει σχεδιαστεί και η κλάση που αναλαμβάνει όλες τις CRUD λειτουργίες με την βάση δεδομένων της εφαρμογής (*DBIntegrator*).

Θα πρέπει να αναφερθεί ακόμα ότι δημιουργήθηκε μία POJO κλάση για κάθε απάντηση του τύπου JSON που επεξεργάζεται η εφαρμογή. Με αυτόν τον τρόπο δεν απαιτούνται πολύπλοκες τεχνικές επεξεργασίας αλφαριθμητικών για να εξαχθεί η πληροφορία από το JSON και επιπλέον οι παραπάνω κλάσεις δημιουργούνται αυτόματα με βάση την δομή του JSON (για παράδειγμα <https://json2csharp.com/json-to-pojo>).

Οι κλάσεις τις εφαρμογής έχουν διακριθεί σε τέσσερεις λογικές ομάδες. Αυτός ο λογικός διαχωρισμός έγινε με βάση την λειτουργική συνάφεια των κλάσεων και είχε ως απώτερο σκοπό την καλύτερη διαχείριση της βάσης του κώδικα (codebase) της εφαρμογής. Αναλυτικότερα τα πακέτα που απαρτίζουν την εφαρμογή TzokerStats είναι:

* Το πακέτο ***tzokerstats***, που είναι το σημείο εκκίνησης (entry point) της εφαρμογής. Είναι εκείνο το κομμάτι της εφαρμογής που προετοιμάζει το περιβάλλον εκτέλεσης του κώδικα που πρόκειται να ακολουθήσει. Επιπλέον περιέχει τις νέες εξαιρέσεις που θεωρήθηκαν απαραίτητες κατά την ανάπτυξη της εφαρμογής (custom exceptions).
* Το πακέτο ***model***, το οποίο ομαδοποιεί τις POJO κλάσεις που δημιουργήθηκαν αυτόματα κάνοντας χρήση του Java Persistence API (JPA). To JPA αφορά μια προγραμματιστική βιβλιοθήκη που χρησιμοποιείται για την εύκολη και αποδοτική διαχείριση μιας βάσης δεδομένων μέσα από κώδικα Java.
* Το πακέτο ***gui***, που περιέχει τις οθόνες της εφαρμογής. Επιπλέον περιέχει βοηθητικές κλάσεις (*MenuOverlay*) που έχουν να κάνουν με το *look and feel* της τελευταίας. Οι κλάσεις του συγκεκριμένου πακέτου κάνουν χρήση των βοηθητικών κλάσεων των υπολοίπων πακέτων, για να ικανοποιήσουν τις απαιτήσεις της εφαρμογής.
* Τέλος το πακέτο ***DataIngestion***, το οποίο περιέχει όλες εκείνες τις βοηθητικές κλάσεις που αναλαμβάνουν να αποκρύψουν την πολυπλοκότητα της επικοινωνίας με τα εξωτερικά συστήματα (OPAP API, Derby Database). Έτσι, οι εν λόγω κλάσεις επεξεργάζονται και επιστρέφουν αντικείμενα, τα οποία μπορούν να καταναλώσουν οι κλάσεις που ανήκουν στο πακέτο gui. Επιπλέον περιέχει τις POJO κλάσεις που δημιουργήθηκαν για την υποδοχή των απαντήσεων των API Calls.

Στην εικόνα 2.1 που ακολουθεί, βρίσκεται το διάγραμμα κλάσεων της εφαρμογής TzokerStats.



Εικόνα 2.1-TzokerStats Class Diagram

β) Το διάγραμμα Οντοτήτων Συσχετίσεων (ER) που ακολουθεί παρουσιάζει με εύληπτο τρόπο τις οντότητες που διέπουν την βάση δεδομένων του συστήματος. Επιπλέον διακρίνονται σε αυτό οι ιδιότητες τους, όπως και οι σχέσεις αναμεσά τους.

Ποιο συγκεκριμένα, παρουσιάζεται η οντότητα αναφοράς (master data) GAME. Η οντότητα αυτή περιέχει πληροφορίες για τα παιχνίδια του ΟΠΑΠ όπως *κωδικός*, *περιγραφή* και *οικογένεια* *παιχνιδιού*. Επιπλέον έχουμε την οντότητα WINNING\_COLUMN που αντιπροσωπεύει την *νικητήρια* *στήλη* για τον συνδυασμό *παιχνιδιού*, και *κωδικού* *κλήρωσης*. Τέλος η οντότητα GAME\_DRAW\_RESULTS αναφέρεται στα *οικονομικά* *αποτελέσματα* της εκάστοτε κλήρωσης και παιχνιδιού ανά *κατηγορία* *κερδών*. Η προηγούμενη οντότητα συνδέεται με την τελευταία μέσω του ξένου σύνθετου κλειδιού (composite foreign key) GAME\_ID και DRAW\_ID. Με την σειρά της η οντότητα GAME συνδέεται με την WINNING\_ COLUMN μέσω του foreign key GAME\_ID. Όλες οι παραπάνω συσχετίσεις έχουν πληθικότητα 1 προς N και είναι προαιρετικές (ή μερικές). Και αυτό το διάγραμμά βρίσκεται στο vpp file που επισυνάπτεται.

|  |
| --- |
|  |

Εικόνα 2.2-JokerStats ER Diagram

Tέλος, τα credentials για την βάση είναι “TzokerStats” & ”pass” και ακολουθεί το SQL Script που ορίζει το σχήμα της βάσης δεδομένων.

|  |
| --- |
| **Database Schema DDL** |
| /\*  Generate tables  \*/  **CREATE** **TABLE** GAME **(**  GAME\_ID **integer**  **,**GAME\_FAMILY\_ID **integer**  **,**GAME\_NAME **varchar(**128**)**  **,PRIMARY** **KEY** **(**GAME\_ID**)**  **);**  **CREATE** **TABLE** WINNING\_COLUMN **(**  GAME\_ID **integer** **NOT** **NULL**  **,**DRAW\_ID **integer** **NOT** **NULL**  **,**DRAW\_TIME **timestamp** **NOT** **NULL**  **,**N\_1 **integer** **NOT** **NULL**  **,**N\_2 **integer** **NOT** **NULL**  **,**N\_3 **integer** **NOT** **NULL**  **,**N\_4 **integer** **NOT** **NULL**  **,**N\_5 **integer** **NOT** **NULL**  **,**J\_1 **integer** **NOT** **NULL**  **,**WINNING\_COLUMN\_DESCR **varchar(**128**)** **NOT** **NULL**  **,PRIMARY** **KEY** **(**GAME\_ID**,** DRAW\_ID**)**  **);**  **CREATE** **TABLE** GAME\_DRAW\_RESULTS **(**  GAME\_ID **integer** **NOT** **NULL**  **,**DRAW\_ID **integer** **NOT** **NULL**  **,**WINNING\_CATEGORY\_DESCR **varchar(**128**)** **NOT** **NULL**  **,**SUCCESSES\_CNT **integer**  **,**DIVIDENT\_AMN **double**  **,**TOTAL\_PAYOUT **double**  **,PRIMARY** **KEY** **(**GAME\_ID**,** DRAW\_ID**,** WINNING\_CATEGORY\_DESCR**)**  **);**  /\*  Generate Relationships  \*/  **ALTER** **TABLE** GAME\_DRAW\_RESULTS  **ADD** **CONSTRAINT** FKGAME\_DRAW **FOREIGN** **KEY** **(**GAME\_ID**,** DRAW\_ID**)**  **REFERENCES** WINNING\_COLUMN **(**GAME\_ID**,** DRAW\_ID**);**    **ALTER** **TABLE** WINNING\_COLUMN  **ADD** **CONSTRAINT** FKWINNING **FOREIGN** **KEY** **(**GAME\_ID**)**  **REFERENCES** GAME **(**GAME\_ID**);** |

(γ) Ακολουθεί ο κώδικας για εκείνες τις κλάσεις της εφαρμογής που δεν εμπίπτουν στις κατηγορίες των παραγράφων 3.2 (GUI) και 3.3 (εμφάνιση στατιστικών).

|  |
| --- |
| **Πακέτο:TzokerStats**  **Κλάση:TzokerStats.Java** |
| package tzokerstats**;**  **import** gui**.**MainFrame**;**  **import** java**.**sql**.**Connection**;**  **import** java**.**sql**.**DriverManager**;**  **import** java**.**sql**.**SQLException**;**  **import** java**.**io**.**File**;**  **import** java**.**io**.**IOException**;**  **import** javax**.**swing**.**JOptionPane**;**  public class TzokerStats **{**  /\*\*  \* **@param** args the command line arguments  \*/  public static void main**(**String**[]** args**)** **{**  /\* Set the Nimbus look and feel \*/  //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  \*/  **try** **{**  **for** **(**javax**.**swing**.**UIManager**.**LookAndFeelInfo info **:** javax**.**swing**.**UIManager**.**getInstalledLookAndFeels**())** **{**  **if** **(**"Nimbus"**.**equals**(**info**.**getName**()))** **{**  javax**.**swing**.**UIManager**.**setLookAndFeel**(**info**.**getClassName**());**  **break;**  **}**  **}**  **}** **catch** **(**ClassNotFoundException ex**)** **{**  java**.**util**.**logging**.**Logger**.**getLogger**(**MainFrame**.**class**.**getName**()).**log**(**java**.**util**.**logging**.**Level**.**SEVERE**,** **null,** ex**);**  **}** **catch** **(**InstantiationException ex**)** **{**  java**.**util**.**logging**.**Logger**.**getLogger**(**MainFrame**.**class**.**getName**()).**log**(**java**.**util**.**logging**.**Level**.**SEVERE**,** **null,** ex**);**  **}** **catch** **(**IllegalAccessException ex**)** **{**  java**.**util**.**logging**.**Logger**.**getLogger**(**MainFrame**.**class**.**getName**()).**log**(**java**.**util**.**logging**.**Level**.**SEVERE**,** **null,** ex**);**  **}** **catch** **(**javax**.**swing**.**UnsupportedLookAndFeelException ex**)** **{**  java**.**util**.**logging**.**Logger**.**getLogger**(**MainFrame**.**class**.**getName**()).**log**(**java**.**util**.**logging**.**Level**.**SEVERE**,** **null,** ex**);**  **}**  //</editor-fold>  //</editor-fold>  //</editor-fold>  //</editor-fold>  **try** **{**  //Start DerbyDB engine before launching app  Process process **=**  Runtime**.**getRuntime**().**exec**(**"cmd /c start /b startNetworkServer.bat"  **,** **null,** **new** File**(**"db"**));**  process**.**waitFor**();**  **}** **catch** **(**InterruptedException **|** IOException e**)** **{**  JOptionPane**.**showMessageDialog**(null,**  "Παρουσιάστηκε σφάλμα κατα την εκκίνηση της βάσης δεδομένων. "  **+** e**.**getMessage**(),**  "Σφάλμα κατα την εκκίνηση της βάσης δεδομένων"**,**  JOptionPane**.**ERROR\_MESSAGE**);**  **}**    /\* Create and display the main form \*/  java**.**awt**.**EventQueue**.**invokeLater**(new** Runnable**()** **{**  public void run**()** **{**  **new** MainFrame**().**setVisible**(true);**  **}**  **});**    **}**  **}** |

|  |
| --- |
| **Πακέτο:TzokerStats**  **Κλάση:** **FoundEntityException.Java** |
| package tzokerstats**;**  //Exception that denotes error due to existance of a data item  public class FoundEntityException **extends** Exception **{**  public FoundEntityException**(**String message**,** Throwable cause**)** **{**  **super(**message**,** cause**);**  **}**  public FoundEntityException**(**String message**)** **{**  **super(**message**);**  **}**  **}** |

|  |
| --- |
| **Πακέτο:TzokerStats**  **Κλάση:** **GUIValidationException.Java** |
| package tzokerstats**;**  //Exception that corresponds to error during a form validation  public class GUIValidationException **extends** Exception **{**  public GUIValidationException**(**String message**,** Throwable cause**)** **{**  **super(**message**,** cause**);**  **}**  public GUIValidationException**(**String message**)** **{**  **super(**message**);**  **}**  **}** |

|  |
| --- |
| **Πακέτο:TzokerStats**  **Κλάση:** **NotFoundEntityException.Java** |
| package tzokerstats**;**  //Exception that denotes error due to non existance of a data item  public class NotFoundEntityException **extends** Exception **{**  public NotFoundEntityException**(**String message**,** Throwable cause**)** **{**  **super(**message**,** cause**);**  **}**  public NotFoundEntityException**(**String message**)** **{**  **super(**message**);**  **}**  **}** |

|  |
| --- |
| **Πακέτο:** **DataIngestion**  **Interface:** **IApiIntegrator.Java** |
| package DataIngestion**;**  **import** java**.**io**.**IOException**;**  /\*  Interface that denotes the basic functionality that all  Classes that provide access to the OPAP API must provide.  It is used so that the client classes (the gui) don't  have to take care of the specifics of each API call. Thus  we can add more Integrator objects (i.e API calls for other games)  without too much effort (loose coupling with the GUI).  For example MultiDrawIntegrator and SingleDrawIntegrator, both  implement this Interface and provide a way of:  - calling the corresponding endpoint,  - transform the JSON response to Java Object and  - return it to the caller.  \*/  public interface IApiIntegrator **{**  public void addAPIargument**(**String paramName**,** String paramValue**);**  public void removeAPIargument**(**String paramName**);**  public void loadDataFromAPI**()** **throws** IOException**;**  public Object getDataObject**(**Class objectName**);**  **}** |

|  |
| --- |
| **Πακέτο:** **DataIngestion**  **Κλάση:** **JsonReader.Java** |
| package DataIngestion**;**  **import** java**.**io**.**BufferedReader**;**  **import** java**.**io**.**IOException**;**  **import** java**.**io**.**InputStreamReader**;**  **import** java**.**net**.**URL**;**  public abstract class JsonReader **{**  public final static String getFromAPI**(**URL url**)** **throws** IOException **{**  StringBuilder result **=** **new** StringBuilder**();**  //Reads text from a character-input stream, and removes not applicable words,  //that is punctuation characters and whitespaces.  //buffering characters so as to provide for the efficient reading of characters, arrays, and lines.  //An InputStreamReader is a bridge from byte streams to character streams:  //It reads bytes and decodes them into characters using a specified charset.  //The charset that it uses may be specified by name or may be given explicitly,  //or the platform's default charset may be accepted.  **try** **{**  BufferedReader in **=** **new** BufferedReader**(new** InputStreamReader**(**url**.**openStream**()));**  String inputLine**;**  **while** **((**inputLine **=** in**.**readLine**())** **!=** **null)** **{**  result**.**append**(**inputLine**);**  result**.**append**(**" "**);**  **}**  **}** **catch** **(**IOException e**)** **{**  System**.**out**.**println**(**e**.**getMessage**());**  **throw** e**;**  **}**  **return** result**.**toString**();**  **}**  **}** |
| **Πακέτο:** **DataIngestion**  **Κλάση:** **DBIntegrator.Java** |
| package DataIngestion**;**  **import** java**.**sql**.**Connection**;**  **import** java**.**sql**.**DriverManager**;**  **import** java**.**sql**.**PreparedStatement**;**  **import** java**.**sql**.**ResultSet**;**  **import** java**.**sql**.**SQLException**;**  **import** javax**.**persistence**.**Query**;**  **import** java**.**util**.**ArrayList**;**  **import** java**.**util**.**Date**;**  **import** java**.**util**.**List**;**  **import** javax**.**persistence**.**EntityManager**;**  **import** javax**.**persistence**.**EntityManagerFactory**;**  **import** javax**.**persistence**.**Persistence**;**  **import** model**.**GameDrawResults**;**  **import** model**.**WinningColumn**;**  **import** tzokerstats**.**FoundEntityException**;**  **import** tzokerstats**.**NotFoundEntityException**;**  /\*  Class that abstracts all db related operations:  - Create  - Delete  - Update  - Delete    Reads from db and uses the POJO Classes in order to  manipulate the data.  Furthermore it executes direct SQL Queries against  the target server.    Important! It contains hardcoded values for the  -DB connection string  -DB Credentials  that should be migrated in an ecrypted external config file (TODO)  \*/  public class DBIntegrator **{**  //TODO: encrypt in a outside file  private final static String DB\_USER\_NAME **=** "TzokerStats"**;**  private final static String DB\_PASSWORD **=** "pass"**;**  private final static String CONNECTION\_STRING **=** "jdbc:derby://localhost:1527/L10L"**;**  static public void setDrawResults**(**List**<**WinningColumn**>** winningColumnsList**,** List**<**GameDrawResults**>** gameDrawResultsList**,** Operation op**)** **throws** FoundEntityException **{**  //Create entity manager factory  EntityManagerFactory emf **=** Persistence**.**createEntityManagerFactory**(**"TzokerStatsPU"**);**  // Create EntityManager  EntityManager em **=** emf**.**createEntityManager**();**  //We should write both tables to one transaction  //to avoid inconsistancies.  //Begin transaction  **try** **{**  em**.**getTransaction**().**begin**();**  //Loop through current winning columns  **for** **(**WinningColumn winningColumn **:** winningColumnsList**)** **{**  **if** **(!**existsWinningColumn**(**winningColumn**))** **{**  //Write to DB Winning Columns  em**.**persist**(**winningColumn**);**  **}** **else** **{**  //throw Found Exception per user choice  **if** **(!**op**.**equals**(**Operation**.**FORCE\_INSERT**))** **{**  **throw** **new** FoundEntityException**(**"Τα επιλεγμένα δεδομένα υπάρχουν ήδη στην βάση."**);**  **}**  **}**  **}**  //Loop through Draw Details  **for** **(**GameDrawResults gameDrawResults **:** gameDrawResultsList**)** **{**  **if** **(!**existsDrawResults**(**gameDrawResults**))** **{**  //Write to DB Winning Columns  em**.**persist**(**gameDrawResults**);**  **}** **else** **{**  //throw Found Exception per user choice  **if** **(!**op**.**equals**(**Operation**.**FORCE\_INSERT**))** **{**  **throw** **new** FoundEntityException**(**"Τα επιλεγμένα δεδομένα υπάρχουν ήδη στην βάση."**);**  **}**  **}**  **}**  //Commit all changes at once  em**.**getTransaction**().**commit**();**  //destroy EntityManager and EntityManagerFactory (why wait for the GC?)  **}** **catch** **(**FoundEntityException e**)** **{**  System**.**out**.**println**(**"Draw data allready exists"**);**  em**.**getTransaction**().**rollback**();**  **throw** **(**e**);**  **}** **catch** **(**Exception e**)** **{**  System**.**out**.**println**(**"Error during DB Insert"**);**  em**.**getTransaction**().**rollback**();**  **throw** **(**e**);**  **}** **finally** **{**  em**.**close**();**  emf**.**close**();**  **}**  **}**  /\*  Checks draws in the input arguments and  returns true if no rows found  false if some draws are found.  If all rows are found raises error  \*/  static public boolean isDrawResultsReadyForInsert**(**List**<**WinningColumn**>** winningColumnsList**,** List**<**GameDrawResults**>** gameDrawResultsList**)** **throws** FoundEntityException **{**  //First check for existance  int howManyEntriesFound **=** 0**;**  int howManyEntries **=** winningColumnsList**.**size**();**  **for** **(**WinningColumn winningColumn **:** winningColumnsList**)** **{**  **if** **(**existsWinningColumn**(**winningColumn**))** **{**  howManyEntriesFound**++;**  **}**  **}**  **if** **(**howManyEntriesFound **==** 0**)** **{**  **return** **true;**  **}** **else** **if** **(**howManyEntriesFound **<** howManyEntries**)** **{**  **return** **false;**  **}** **else** **if** **(**howManyEntriesFound **==** howManyEntries**)** **{**  //All rows found raise error  **throw** **new** FoundEntityException**(**"Τα επιλεγμένα δεδομένα υπάρχουν ήδη στην βάση."**);**  **}**  **return** **false;**  **}**  /\*  Delete for selected Draw ID  Using transaction control to avoid inconsistency among the tables!  \*/  public static int deleteDrawDataWithID**(**String gameID**,** String drawID**)** **throws** SQLException**,** NotFoundEntityException **{**  AggregatedDrawData aggregatedDrawData **=** **null;**  int deletedDraws **=** 0**;**  String deleteGameDrawResultsSQL **=** "Delete From TZOKERSTATS.GAME\_DRAW\_RESULTS Where DRAW\_ID = ? and GAME\_ID = ?"**;**  String deleteWinningColumnSQL **=** "Delete From TZOKERSTATS.WINNING\_COLUMN Where DRAW\_ID = ? and GAME\_ID = ?"**;**  Connection connection **=** **null;**  **try** **{**  connection **=** connect**();**  connection**.**setAutoCommit**(false);**  //Delete statement for GAME\_DRAW\_RESULTS table  PreparedStatement deleteGameDrawResults **=** connection**.**prepareStatement**(**deleteGameDrawResultsSQL**);**  deleteGameDrawResults**.**setString**(**1**,** drawID**);**  deleteGameDrawResults**.**setString**(**2**,** gameID**);**  deleteGameDrawResults**.**executeUpdate**();**  //Delete statement for WINNING\_COLUMN table  PreparedStatement deleteWinningColumn **=** connection**.**prepareStatement**(**deleteWinningColumnSQL**);**  deleteWinningColumn**.**setString**(**1**,** drawID**);**  deleteWinningColumn**.**setString**(**2**,** gameID**);**  deletedDraws **=** deleteWinningColumn**.**executeUpdate**();**  connection**.**commit**();**  connection**.**close**();**  **if** **(**deletedDraws **<** 1**)** **{**  **throw** **new** NotFoundEntityException**(**"Δεν βρέθηκαν δεδομένα στην βάση με τα επιλεγμένα κριτήρια."**);**  **}**  **}** **catch** **(**SQLException ex**)** **{**  **if** **(**connection **!=** **null)** **{**  System**.**err**.**print**(**"Transaction is being rolled back"**);**  connection**.**rollback**();**  deletedDraws **=** 0**;**  **}**  **throw** **(**ex**);**  **}**  **return** deletedDraws**;**  **}**  /\*  Delete for selected Draw ID  Using transaction control to avoid inconsistency among the tables!  \*/  public static int deleteDrawDataWithDateRange**(**String gameID**,** String dateFrom**,** String dateTo**)** **throws** SQLException**,** NotFoundEntityException **{**  AggregatedDrawData aggregatedDrawData **=** **null;**  int deletedDraws **=** 0**;**  String deleteGameDrawResultsSQL **=** "Delete\n"  **+** "From TZOKERSTATS.GAME\_DRAW\_RESULTS\n"  **+** "Where game\_id = ?\n"  **+** "and DRAW\_ID in (\n"  **+** " SELECT DRAW\_ID \n"  **+** " FROM TZOKERSTATS.WINNING\_COLUMN \n"  **+** " where game\_id = ? \n"  **+** " and cast(draw\_time as date) between ? and ?\n"  **+** ")"**;**  String deleteWinningColumnSQL **=** "Delete \n"  **+** " FROM TZOKERSTATS.WINNING\_COLUMN \n"  **+** " where game\_id = ? \n"  **+** " and cast(draw\_time as date) between ? and ?"**;**  Connection connection **=** **null;**  **try** **{**  connection **=** connect**();**  connection**.**setAutoCommit**(false);**  //Delete statement for GAME\_DRAW\_RESULTS table  PreparedStatement deleteGameDrawResults **=** connection**.**prepareStatement**(**deleteGameDrawResultsSQL**);**  deleteGameDrawResults**.**setString**(**1**,** gameID**);**  deleteGameDrawResults**.**setString**(**2**,** gameID**);**  deleteGameDrawResults**.**setString**(**3**,** dateFrom**);**  deleteGameDrawResults**.**setString**(**4**,** dateTo**);**  deleteGameDrawResults**.**executeUpdate**();**  //Delete statement for WINNING\_COLUMN table  PreparedStatement deleteWinningColumn **=** connection**.**prepareStatement**(**deleteWinningColumnSQL**);**  deleteWinningColumn**.**setString**(**1**,** gameID**);**  deleteWinningColumn**.**setString**(**2**,** dateFrom**);**  deleteWinningColumn**.**setString**(**3**,** dateTo**);**  deletedDraws **=** deleteWinningColumn**.**executeUpdate**();**  connection**.**commit**();**  connection**.**close**();**  **if** **(**deletedDraws **<** 1**)** **{**  **throw** **new** NotFoundEntityException**(**"Δεν βρέθηκαν δεδομένα στην βάση με τα επιλεγμένα κριτήρια."**);**  **}**  **}** **catch** **(**SQLException ex**)** **{**  **if** **(**connection **!=** **null)** **{**  System**.**err**.**print**(**"Transaction is being rolled back"**);**  connection**.**rollback**();**  deletedDraws **=** 0**;**  **}**  **throw** **(**ex**);**  **}**  **return** deletedDraws**;**  **}**  static public void deleteALLDrawResults**()** **{**  //Create entity manager factory  EntityManagerFactory emf **=** Persistence**.**createEntityManagerFactory**(**"TzokerStatsPU"**);**  // Create EntityManager  EntityManager em **=** emf**.**createEntityManager**();**  //We should write both tables to one transaction  //to avoid inconsistencies.  //Begin transaction  **try** **{**  em**.**getTransaction**().**begin**();**  em**.**createNativeQuery**(**"truncate table TZOKERSTATS.WINNING\_COLUMN"**).**executeUpdate**();**  em**.**createNativeQuery**(**"truncate table TZOKERSTATS.GAME\_DRAW\_RESULTS"**).**executeUpdate**();**  em**.**getTransaction**().**commit**();**  **}** **catch** **(**Exception e**)** **{**  System**.**out**.**println**(**"Error during DB delete"**);**  em**.**getTransaction**().**rollback**();**  **throw** **(**e**);**  **}** **finally** **{**  em**.**close**();**  emf**.**close**();**  **}**  **}**  //Check for rows in table by draw ID  //It is being used inside this class for checking the existance  //of a record before the execution of an insert operation.  static private boolean existsDrawResults**(**GameDrawResults gameDrawResults**)** **{**  //Create entity manager factory  EntityManagerFactory emf **=** Persistence**.**createEntityManagerFactory**(**"TzokerStatsPU"**);**  // Create EntityManager  EntityManager em **=** emf**.**createEntityManager**();**  int drawID **=** gameDrawResults**.**getGameDrawResultsPK**().**getDrawId**();**  int gameID **=** gameDrawResults**.**getGameDrawResultsPK**().**getGameId**();**  //prepare query  Query q **=** em**.**createNamedQuery**(**"GameDrawResults.findByDrawId"**,** GameDrawResults**.**class**);**  q**.**setParameter**(**"drawId"**,** drawID**);**  q**.**setParameter**(**"gameId"**,** gameID**);**  ArrayList**<**GameDrawResults**>** eList **=** **new** ArrayList**<**GameDrawResults**>(**q**.**getResultList**());**  em**.**close**();**  emf**.**close**();**  **if** **(**eList**.**size**()** **<** 1**)** **{**  **return** **false;**  **}** **else** **{**  **return** **true;**  **}**  **}**  //Check for rows in table by draw ID  static private boolean existsWinningColumn**(**WinningColumn winningColumn**)** **{**  //Create entity manager factory  EntityManagerFactory emf **=** Persistence**.**createEntityManagerFactory**(**"TzokerStatsPU"**);**  // Create EntityManager  EntityManager em **=** emf**.**createEntityManager**();**  int drawID **=** winningColumn**.**getWinningColumnPK**().**getDrawId**();**  int gameID **=** winningColumn**.**getWinningColumnPK**().**getGameId**();**  //prepare query  Query q **=** em**.**createNamedQuery**(**"WinningColumn.findByDrawId"**,** WinningColumn**.**class**);**  q**.**setParameter**(**"drawId"**,** drawID**);**  q**.**setParameter**(**"gameId"**,** gameID**);**  ArrayList**<**WinningColumn**>** eList **=** **new** ArrayList**<**WinningColumn**>(**q**.**getResultList**());**  em**.**close**();**  emf**.**close**();**  **if** **(**eList**.**size**()** **<** 1**)** **{**  **return** **false;**  **}** **else** **{**  **return** **true;**  **}**  **}**  //Get Aggregated data for a game in date range  public static AggregatedDrawData getDataForMonth**(**int gameID**,** String dateFrom**,** String dateTo**)** **throws** SQLException **{**  AggregatedDrawData aggregatedDrawData **=** **null;**  **try** **{**  Connection connection **=** connect**();**  String selectSQL **=** "SELECT count(DISTINCT t1.DRAW\_ID) As DRAW\_COUNT\n"  **+** " ,SUM(SUCCESSES\_CNT \* DIVIDENT\_AMN) As PAYOUT\_AMN\n"  **+** " ,SUM(CASE When WINNING\_CATEGORY\_DESCR in ('1') And SUCCESSES\_CNT = 0 Then 1 Else 0 END) As JACKPOTS\_COUNT\n"  **+** " From TZOKERSTATS.GAME\_DRAW\_RESULTS t1\n"  **+** " Inner\n"  **+** " Join TZOKERSTATS.WINNING\_COLUMN t2\n"  **+** " On t1.DRAW\_ID = t2.DRAW\_ID \n"  **+** " And t1.GAME\_ID = t2.GAME\_ID\n"  **+** " Where Cast(t2.DRAW\_TIME as Date) Between ? And ? And t2.GAME\_ID = ?"**;**  PreparedStatement preparedStatement **=** connection**.**prepareStatement**(**selectSQL**);**  preparedStatement**.**setString**(**1**,** dateFrom**);**  preparedStatement**.**setString**(**2**,** dateTo**);**  preparedStatement**.**setString**(**3**,** String**.**valueOf**(**gameID**));**  ResultSet resultSet **=** preparedStatement**.**executeQuery**();**  int drawCount **=** 0**;**  double payout **=** 0**;**  int jackpotCount **=** 0**;**  **if** **(**resultSet**.**next**())** **{**  drawCount **=** resultSet**.**getInt**(**"DRAW\_COUNT"**);**  payout **=** resultSet**.**getDouble**(**"PAYOUT\_AMN"**);**  jackpotCount **=** resultSet**.**getInt**(**"JACKPOTS\_COUNT"**);**  **}**  preparedStatement**.**close**();**  connection**.**close**();**  aggregatedDrawData **=** **new** AggregatedDrawData**(**drawCount**,** payout**,** jackpotCount**);**  **}** **catch** **(**SQLException ex**)** **{**  **throw** **(**ex**);**  **}**  **return** aggregatedDrawData**;**  **}**  //Get Aggregated data for a game by drawid in date range  public static List**<**AggregatedDrawDataByDraw**>** getAggregatedDrawDataByDraw**(**int gameID**,** String dateFrom**,** String dateTo**)** **throws** SQLException **{**  List**<**AggregatedDrawDataByDraw**>** aggregatedDrawDataByDrawList **=** **new** ArrayList**<>();**  **try** **{**  Connection connection **=** connect**();**  String selectSQL **=** "SELECT t1.DRAW\_ID As DRAW\_ID\n"  **+** " ,SUM(SUCCESSES\_CNT \* DIVIDENT\_AMN) As PAYOUT\_AMN\n"  **+** " ,SUM(CASE When WINNING\_CATEGORY\_DESCR in ('1') And SUCCESSES\_CNT = 0 Then 1 Else 0 END) As JACKPOTS\_COUNT\n"  **+** " From TZOKERSTATS.GAME\_DRAW\_RESULTS t1\n"  **+** " Inner\n"  **+** " Join TZOKERSTATS.WINNING\_COLUMN t2\n"  **+** " On t1.DRAW\_ID = t2.DRAW\_ID \n"  **+** " And t1.GAME\_ID = t2.GAME\_ID\n"  **+** " Where Cast(t2.DRAW\_TIME as Date) Between ? And ? And t2.GAME\_ID = ? \n"  **+** " Group\n"  **+** " By t1.DRAW\_ID\n"  **+** " Order \n"  **+** " By t1.DRAW\_ID\n"**;**  PreparedStatement preparedStatement **=** connection**.**prepareStatement**(**selectSQL**);**  preparedStatement**.**setString**(**1**,** dateFrom**);**  preparedStatement**.**setString**(**2**,** dateTo**);**  preparedStatement**.**setString**(**3**,** String**.**valueOf**(**gameID**));**  ResultSet resultSet **=** preparedStatement**.**executeQuery**();**  int drawID **=** 0**;**  double payout **=** 0**;**  int jackpotCount **=** 0**;**  **while** **(**resultSet**.**next**())** **{**  drawID **=** resultSet**.**getInt**(**"DRAW\_ID"**);**  payout **=** resultSet**.**getDouble**(**"PAYOUT\_AMN"**);**  jackpotCount **=** resultSet**.**getInt**(**"JACKPOTS\_COUNT"**);**  AggregatedDrawDataByDraw aggregatedDrawDataByDraw **=** **new** AggregatedDrawDataByDraw**(**drawID**,** payout**,** jackpotCount**);**  aggregatedDrawDataByDrawList**.**add**(**aggregatedDrawDataByDraw**);**  **}**  preparedStatement**.**close**();**  connection**.**close**();**  **}** **catch** **(**SQLException ex**)** **{**  **throw** **(**ex**);**  **}**  **return** aggregatedDrawDataByDrawList**;**  **}**  //Get Aggregated data for a game in date range  public static List**<**AggregatedDrawDataByMonth**>** getAggregatedDrawDataByMonth**(**int gameID**,** String dateFrom**,** String dateTo**)** **throws** SQLException **{**  List**<**AggregatedDrawDataByMonth**>** aggregatedDrawDataByMonthList **=** **new** ArrayList**<>();**  **try** **{**  Connection connection **=** connect**();**  String selectSQL **=** "SELECT month(t2.draw\_time) As MONTH\_ID\n"  **+** " ,SUM(SUCCESSES\_CNT \* DIVIDENT\_AMN) As PAYOUT\_AMN\n"  **+** " ,SUM(CASE When WINNING\_CATEGORY\_DESCR in ('1') And SUCCESSES\_CNT = 0 Then 1 Else 0 END) As JACKPOTS\_COUNT\n"  **+** " From TZOKERSTATS.GAME\_DRAW\_RESULTS t1\n"  **+** " Inner\n"  **+** " Join TZOKERSTATS.WINNING\_COLUMN t2\n"  **+** " On t1.DRAW\_ID = t2.DRAW\_ID \n"  **+** " And t1.GAME\_ID = t2.GAME\_ID\n"  **+** " Where Cast(t2.DRAW\_TIME as Date) Between ? And ? And t2.GAME\_ID = ? \n"  **+** " Group\n"  **+** " By month(t2.draw\_time)\n"  **+** " Order \n"  **+** " By month(t2.draw\_time)\n"**;**  PreparedStatement preparedStatement **=** connection**.**prepareStatement**(**selectSQL**);**  preparedStatement**.**setString**(**1**,** dateFrom**);**  preparedStatement**.**setString**(**2**,** dateTo**);**  preparedStatement**.**setString**(**3**,** String**.**valueOf**(**gameID**));**  ResultSet resultSet **=** preparedStatement**.**executeQuery**();**  int monthID **=** 0**;**  double payout **=** 0**;**  int jackpotCount **=** 0**;**  **while** **(**resultSet**.**next**())** **{**  monthID **=** resultSet**.**getInt**(**"MONTH\_ID"**);**  payout **=** resultSet**.**getDouble**(**"PAYOUT\_AMN"**);**  jackpotCount **=** resultSet**.**getInt**(**"JACKPOTS\_COUNT"**);**  AggregatedDrawDataByMonth aggregatedDrawDataByMonth **=** **new** AggregatedDrawDataByMonth**(**monthID**,** payout**,** jackpotCount**);**  aggregatedDrawDataByMonthList**.**add**(**aggregatedDrawDataByMonth**);**  **}**  preparedStatement**.**close**();**  connection**.**close**();**  **}** **catch** **(**SQLException ex**)** **{**  **throw** **(**ex**);**  **}**  **return** aggregatedDrawDataByMonthList**;**  **}**  static public ArrayList**<**WinningColumn**>** getWinningColumnByDateRange**(**Date drawTimeFrom**,** Date drawTimeTo**)** **{**  //Create entity manager factory  EntityManagerFactory emf **=** Persistence**.**createEntityManagerFactory**(**"TzokerStatsPU"**);**  // Create EntityManager  EntityManager em **=** emf**.**createEntityManager**();**  //prepare query  Query q **=** em**.**createNamedQuery**(**"WinningColumn.findByDrawTimeRange"**,** WinningColumn**.**class**);**  q**.**setParameter**(**"drawTimeFrom"**,** drawTimeFrom**);**  q**.**setParameter**(**"drawTimeTo"**,** drawTimeTo**);**  ArrayList**<**WinningColumn**>** resultsList **=** **new** ArrayList**<**WinningColumn**>(**q**.**getResultList**());**  em**.**close**();**  emf**.**close**();**  **return** resultsList**;**  **}**    static public ArrayList**<**ArrayList**<**GameDrawResults**>>** getDrawResultsByDrawId**(**ArrayList**<**Integer**>** drawIds**,** int gameID**){**  ArrayList**<**ArrayList**<**GameDrawResults**>** **>** totalGameDrawResults **=** **new** ArrayList**<**ArrayList**<**GameDrawResults**>** **>();**  //Create entity manager factory  EntityManagerFactory emf **=** Persistence**.**createEntityManagerFactory**(**"TzokerStatsPU"**);**  // Create EntityManager  EntityManager em **=** emf**.**createEntityManager**();**  **for** **(**Integer drawId **:** drawIds**){**  //prepare query  Query q **=** em**.**createNamedQuery**(**"GameDrawResults.findByDrawId"**,** GameDrawResults**.**class**);**  q**.**setParameter**(**"drawId"**,**drawId**);**  q**.**setParameter**(**"gameId"**,** gameID**);**  ArrayList**<**GameDrawResults**>** singleGameDrawResults **=** **new** ArrayList**<**GameDrawResults**>(**q**.**getResultList**());**  totalGameDrawResults**.**add**(**singleGameDrawResults**);**  **}**  em**.**close**();**  emf**.**close**();**    **return** totalGameDrawResults**;**  **}**    public enum Operation **{**  FORCE\_DELETE**,**  DELETE**,**  INSERT**,**  FORCE\_INSERT  **}**  public static class AggregatedDrawData **{**  private int numberOfDraws**;**  private int numberOfJackpots**;**  private double payoutAmount**;**  public AggregatedDrawData**(**int numberOfDraws**,** double payoutAmount**,** int numberOfJackpots**)** **{**  **this.**numberOfDraws **=** numberOfDraws**;**  **this.**numberOfJackpots **=** numberOfJackpots**;**  **this.**payoutAmount **=** payoutAmount**;**  **}**  public int getNumberOfDraws**()** **{**  **return** numberOfDraws**;**  **}**  public void setNumberOfDraws**(**int numberOfDraws**)** **{**  **this.**numberOfDraws **=** numberOfDraws**;**  **}**  public int getNumberOfJackpots**()** **{**  **return** numberOfJackpots**;**  **}**  public void setNumberOfJackpots**(**int numberOfJackpots**)** **{**  **this.**numberOfJackpots **=** numberOfJackpots**;**  **}**  public double getPayoutAmount**()** **{**  **return** payoutAmount**;**  **}**  public void setPayoutAmount**(**double payoutAmount**)** **{**  **this.**payoutAmount **=** payoutAmount**;**  **}**  **}**  public static class AggregatedDrawDataByDraw **{**  private int drawID**;**  private int numberOfJackpots**;**  private double payoutAmount**;**  public AggregatedDrawDataByDraw**(**int drawID**,** double payoutAmount**,** int numberOfJackpots**)** **{**  **this.**drawID **=** drawID**;**  **this.**numberOfJackpots **=** numberOfJackpots**;**  **this.**payoutAmount **=** payoutAmount**;**  **}**  public int getDrawID**()** **{**  **return** drawID**;**  **}**  public void setDrawID**(**int drawID**)** **{**  **this.**drawID **=** drawID**;**  **}**  public int getNumberOfJackpots**()** **{**  **return** numberOfJackpots**;**  **}**  public void setNumberOfJackpots**(**int numberOfJackpots**)** **{**  **this.**numberOfJackpots **=** numberOfJackpots**;**  **}**  public double getPayoutAmount**()** **{**  **return** payoutAmount**;**  **}**  public void setPayoutAmount**(**double payoutAmount**)** **{**  **this.**payoutAmount **=** payoutAmount**;**  **}**  **}**  public static class AggregatedDrawDataByMonth **{**  private int monthID**;**  private int numberOfJackpots**;**  private double payoutAmount**;**  public AggregatedDrawDataByMonth**(**int monthID**,** double payoutAmount**,** int numberOfJackpots**)** **{**  **this.**monthID **=** monthID**;**  **this.**numberOfJackpots **=** numberOfJackpots**;**  **this.**payoutAmount **=** payoutAmount**;**  **}**  public int getMonthID**()** **{**  **return** monthID**;**  **}**  public void setMonthID**(**int drawID**)** **{**  **this.**monthID **=** drawID**;**  **}**  public int getNumberOfJackpots**()** **{**  **return** numberOfJackpots**;**  **}**  public void setNumberOfJackpots**(**int numberOfJackpots**)** **{**  **this.**numberOfJackpots **=** numberOfJackpots**;**  **}**  public double getPayoutAmount**()** **{**  **return** payoutAmount**;**  **}**  public void setPayoutAmount**(**double payoutAmount**)** **{**  **this.**payoutAmount **=** payoutAmount**;**  **}**  **}**  //Create and return a new connection to the L10DB  private static Connection connect**()** **throws** SQLException **{**  Connection conn **=** **null;**  **try** **{**  conn **=** DriverManager**.**getConnection**(**CONNECTION\_STRING**,** DB\_USER\_NAME**,** DB\_PASSWORD**);**  **}** **catch** **(**SQLException ex**)** **{**  System**.**out**.**println**(**ex**.**getMessage**());**  **throw** **(**ex**);**  **}**  **return** conn**;**  **}**  **}** |

|  |
| --- |
| **Πακέτο:** **DataIngestion**  **Κλάση:** **MultiDrawIntegrator.Java** |
| package DataIngestion**;**  **import** com**.**google**.**gson**.**Gson**;**  **import** java**.**io**.**IOException**;**  **import** java**.**net**.**URL**;**  **import** java**.**util**.**ArrayList**;**  **import** java**.**util**.**Date**;**  **import** java**.**util**.**HashMap**;**  **import** java**.**util**.**List**;**  **import** model**.**GameDrawResults**;**  **import** model**.**WinningColumn**;**  **import** model**.**WinningColumnPK**;**  /\*  Utility class that is designed to  foster easy comunication with external APIs  regarding the multiple draw results data  \*/  public class MultiDrawIntegrator **extends** JsonReader **implements** IApiIntegrator**{**  private final String MULTI\_DRAWS\_URL **=** "https://api.opap.gr/draws/v3.0/{gameID}/draw-date/{fromDate}/{toDate}"**;**  private MultiDrawResults multiDrawResults**;**  private HashMap**<**String**,** String**>** queryParams **=** **new** HashMap**<**String**,** String**>();**    //Fetch data from api to API Object  @Override  public void loadDataFromAPI**()** **throws** IOException**{**  //Prepare endpoint URL  String finalURL **=** MULTI\_DRAWS\_URL**;**  **for** **(**String key **:** queryParams**.**keySet**())** **{**  String formatedKey **=** "{" **+** key **+** "}"**;**  finalURL **=** finalURL**.**replace**(** formatedKey**,** queryParams**.**get**(**key**));**  **}**  URL DrawResultsWithDateURL **=** **new** URL**(**finalURL**);**  //Get Results by calling the REST API  String ApiResponse **=** getFromAPI**(**DrawResultsWithDateURL**);**  //Convert to SingleDrawResults Object  //and return to caller  Gson gson **=** **new** Gson**();**  multiDrawResults **=** gson**.**fromJson**(**ApiResponse**,** MultiDrawResults**.**class**);**  **}**      @Override  public Object getDataObject**(**Class ObjectName**)** **{**  **if** **(**ObjectName **==** GameDrawResults**.**class**){**  **return** getGameDrawResults**();**  **}** **else** **if** **(**ObjectName **==** WinningColumn**.**class**)** **{**  **return** getWinningColumnList**();**  **}**  **return** **null;**  **}**    private List**<**GameDrawResults**>** getGameDrawResults**()** **{**    List**<**GameDrawResults**>** gameDrawResultsList **=** **new** ArrayList**<>();**  //iterate through draws  **for** **(**MultiDrawResults**.**Content content**:**multiDrawResults**.**content**){**  //iterate through prizecategories in draw  **for** **(**MultiDrawResults**.**PrizeCategory prizeCategory **:** content**.**prizeCategories**)** **{**  //Create GameDrawResults entry  GameDrawResults gameDrawResults **=** **new** GameDrawResults**(**content**.**gameId**,**  content**.**drawId**,**  String**.**valueOf**(**prizeCategory**.**id**));**  //And continuing with the rest of the fields  gameDrawResults**.**setDividentAmn**(**prizeCategory**.**divident**);**  gameDrawResults**.**setSuccessesCnt**(**prizeCategory**.**winners**);**  gameDrawResults**.**setTotalPayout**(**prizeCategory**.**distributed**);**  gameDrawResultsList**.**add**(**gameDrawResults**);**  **}**  **}**  **return** gameDrawResultsList**;**  **}**      private List**<**WinningColumn**>** getWinningColumnList**()** **{**    List**<**WinningColumn**>** winningColumnList **=** **new** ArrayList**<>();**    //iterate through draws  **for** **(**MultiDrawResults**.**Content content**:**multiDrawResults**.**content**){**  WinningColumnPK winningColumnPK **=** **new** WinningColumnPK**(**content**.**gameId**,**content**.**drawId**);**  WinningColumn WinningColumn **=** **new** WinningColumn**(**winningColumnPK**);**  WinningColumn**.**setN1**(**content**.**winningNumbers**.**list**.**get**(**0**));**  WinningColumn**.**setN1**(**content**.**winningNumbers**.**list**.**get**(**0**));**  WinningColumn**.**setN2**(**content**.**winningNumbers**.**list**.**get**(**1**));**  WinningColumn**.**setN3**(**content**.**winningNumbers**.**list**.**get**(**2**));**  WinningColumn**.**setN4**(**content**.**winningNumbers**.**list**.**get**(**3**));**  WinningColumn**.**setN5**(**content**.**winningNumbers**.**list**.**get**(**4**));**  WinningColumn**.**setJ1**(**content**.**winningNumbers**.**bonus**.**get**(**0**));**  String winningColumnDescr **=** WinningColumn**.**getN1**()** **+** ","  **+** WinningColumn**.**getN2**()** **+** ","**+** WinningColumn**.**getN3**()** **+** ","  **+** WinningColumn**.**getN4**()** **+** ","**+** WinningColumn**.**getN5**()** **+** "/"  **+** WinningColumn**.**getJ1**();**  WinningColumn**.**setWinningColumnDescr**(**winningColumnDescr**);**  //Get Draw Datetime  long drawTime **=** **(**long**)** content**.**drawTime**;**  //and convert from unix timestamp  Date drawDateTime **=** **new** java**.**util**.**Date**((**long**)**drawTime**);**  WinningColumn**.**setDrawTime**(**drawDateTime**);**  //add winning column objectg to list  winningColumnList**.**add**(**WinningColumn**);**  **}**  **return** winningColumnList**;**  **}**  @Override  public void addAPIargument**(**String paramName**,** String paramValue**)** **{**  queryParams**.**put**(**paramName**,** paramValue**);**  **}**  @Override  public void removeAPIargument**(**String paramName**)** **{**  queryParams**.**remove**(**paramName**);**  **}**  **}** |

|  |
| --- |
| **Πακέτο:** **DataIngestion**  **Κλάση:** **SingleDrawIntegrator.Java** |
| package DataIngestion**;**  **import** com**.**google**.**gson**.**Gson**;**  **import** java**.**io**.**IOException**;**  **import** java**.**net**.**URL**;**  **import** java**.**util**.**ArrayList**;**  **import** java**.**util**.**Date**;**  **import** java**.**util**.**HashMap**;**  **import** java**.**util**.**List**;**  **import** model**.**GameDrawResults**;**  **import** model**.**WinningColumn**;**  **import** model**.**WinningColumnPK**;**  /\*  Utility class that is designed to  foster easy comunication with external APIs  regarding the single draw data  \*/  public class SingleDrawIntegrator **extends** JsonReader **implements** IApiIntegrator**{**  private final String MULTI\_DRAWS\_URL **=** "https://api.opap.gr/draws/v3.0/{gameID}/{drawID}"**;**  private SingleDrawResults singleDrawResults**;**  private HashMap**<**String**,** String**>** queryParams **=** **new** HashMap**<**String**,** String**>();**    //Fetch data from api to API Object  @Override  public void loadDataFromAPI**()** **throws** IOException**{**  //Prepare endpoint URL  String finalURL **=** MULTI\_DRAWS\_URL**;**  **for** **(**String key **:** queryParams**.**keySet**())** **{**  String formatedKey **=** "{" **+** key **+** "}"**;**  finalURL **=** finalURL**.**replace**(** formatedKey**,** queryParams**.**get**(**key**));**  **}**  URL DrawResultsWithDateURL **=** **new** URL**(**finalURL**);**  //Get Results by calling the REST API  String ApiResponse **=** getFromAPI**(**DrawResultsWithDateURL**);**  //Convert to SingleDrawResults Object  //and return to caller  Gson gson **=** **new** Gson**();**  singleDrawResults **=** gson**.**fromJson**(**ApiResponse**,** SingleDrawResults**.**class**);**  **}**    @Override  public Object getDataObject**(**Class ObjectName**)** **{**  **if** **(**ObjectName **==** GameDrawResults**.**class**){**  **return** getGameDrawResults**();**  **}** **else** **if** **(**ObjectName **==** WinningColumn**.**class**)** **{**  **return** getWinningColumnList**();**  **}**  **return** **null;**  **}**    private List**<**GameDrawResults**>** getGameDrawResults**()** **{**    List**<**GameDrawResults**>** gameDrawResultsList **=** **new** ArrayList**<>();**  //iterate through prizecategories in draw  **for** **(**SingleDrawResults**.**PrizeCategory prizeCategory **:** singleDrawResults**.**prizeCategories**)** **{**  //Create GameDrawResults entry  GameDrawResults gameDrawResults **=** **new** GameDrawResults**(**singleDrawResults**.**gameId**,**  singleDrawResults**.**drawId**,**  String**.**valueOf**(**prizeCategory**.**id**));**  //And continuing with the rest of the fields  gameDrawResults**.**setDividentAmn**(**prizeCategory**.**divident**);**  gameDrawResults**.**setSuccessesCnt**(**prizeCategory**.**winners**);**  gameDrawResults**.**setTotalPayout**(**prizeCategory**.**distributed**);**  gameDrawResultsList**.**add**(**gameDrawResults**);**  **}**  **return** gameDrawResultsList**;**  **}**      private List**<**WinningColumn**>** getWinningColumnList**()** **{**  List**<**WinningColumn**>** winningColumnList **=** **new** ArrayList**<>();**  WinningColumnPK winningColumnPK **=** **new** WinningColumnPK**(**singleDrawResults**.**gameId**,**  singleDrawResults**.**drawId**);**  WinningColumn WinningColumn **=** **new** WinningColumn**(**winningColumnPK**);**  WinningColumn**.**setN1**(**singleDrawResults**.**winningNumbers**.**list**.**get**(**0**));**  WinningColumn**.**setN2**(**singleDrawResults**.**winningNumbers**.**list**.**get**(**1**));**  WinningColumn**.**setN3**(**singleDrawResults**.**winningNumbers**.**list**.**get**(**2**));**  WinningColumn**.**setN4**(**singleDrawResults**.**winningNumbers**.**list**.**get**(**3**));**  WinningColumn**.**setN5**(**singleDrawResults**.**winningNumbers**.**list**.**get**(**4**));**  WinningColumn**.**setJ1**(**singleDrawResults**.**winningNumbers**.**bonus**.**get**(**0**));**  String winningColumnDescr **=** WinningColumn**.**getN1**()** **+** ","  **+** WinningColumn**.**getN2**()** **+** ","**+** WinningColumn**.**getN3**()** **+** ","  **+** WinningColumn**.**getN4**()** **+** ","**+** WinningColumn**.**getN5**()** **+** "/"  **+** WinningColumn**.**getJ1**();**  WinningColumn**.**setWinningColumnDescr**(**winningColumnDescr**);**  //Get Draw Datetime  long drawTime **=** singleDrawResults**.**drawTime**;**  //and convert from unix timestamp  Date drawDateTime **=** **new** java**.**util**.**Date**((**long**)**drawTime**);**  WinningColumn**.**setDrawTime**(**drawDateTime**);**  //add winning column objectg to list  winningColumnList**.**add**(**WinningColumn**);**    **return** winningColumnList**;**  **}**    @Override  public void addAPIargument**(**String paramName**,** String paramValue**)** **{**  queryParams**.**put**(**paramName**,** paramValue**);**  **}**  @Override  public void removeAPIargument**(**String paramName**)** **{**  queryParams**.**remove**(**paramName**);**  **}**  **}** |

|  |
| --- |
| **Πακέτο:** **DataIngestion**  **Κλάση:** **WinningColumnStatsIntegrator.Java** |
| package DataIngestion**;**  **import** com**.**google**.**gson**.**Gson**;**  **import** java**.**io**.**IOException**;**  **import** java**.**net**.**URL**;**  **import** java**.**util**.**HashMap**;**  /\*  Class that abstracts the communication  with the OPAP API and the data preparation  process for game statistics related data.  \*/  public class WinningColumnStatsIntegrator **extends** JsonReader **implements** IApiIntegrator **{**  private final String TOTAL\_STATS\_URL **=** "https://api.opap.gr/games/v1.0/{gameID}/statistics"**;**  private WinningColumnStats totalNumberStats**;**  private HashMap**<**String**,** String**>** queryParams **=** **new** HashMap**<**String**,** String**>();**  //Fetch data from api to API Object  @Override  public void loadDataFromAPI**()** **throws** IOException **{**  //Prepare endpoint URL  String finalURL **=** TOTAL\_STATS\_URL**;**  **for** **(**String key **:** queryParams**.**keySet**())** **{**  String formatedKey **=** "{" **+** key **+** "}"**;**  finalURL **=** finalURL**.**replace**(**formatedKey**,** queryParams**.**get**(**key**));**  **}**  URL TotalNumberStats **=** **new** URL**(**finalURL**);**  //Get Results by calling the REST API  String ApiResponse **=** getFromAPI**(**TotalNumberStats**);**  //Convert to SingleDrawResults Object  //and return to caller  Gson gson **=** **new** Gson**();**  totalNumberStats **=** gson**.**fromJson**(**ApiResponse**,** WinningColumnStats**.**class**);**  **}**  public WinningColumnStats getTotalNumberStats**()** **{**  **return** totalNumberStats**;**  **}**  @Override  public void addAPIargument**(**String paramName**,** String paramValue**)** **{**  queryParams**.**put**(**paramName**,** paramValue**);**  **}**  @Override  public void removeAPIargument**(**String paramName**)** **{**  queryParams**.**remove**(**paramName**);**  **}**  @Override  public Object getDataObject**(**Class objectName**)** **{**  **if** **(**objectName **==** WinningColumnStats**.**class**)** **{**  **return** totalNumberStats**;**  **}**  **return** **null;**  **}**  **}** |

(δ) Οι κλάσεις που ανήκουν στο πακέτο ***model***, δημιουργήθηκαν αυτόματα με την βοήθεια του αντίστοιχου οδηγού που παρέχει το NetBeans. Ακολουθούν τα κομμάτια εκείνα το κώδικα που προστέθηκαν εκ των υστέρων από την ομάδα ανάπτυξης. Επίσης θα πρέπει να αναφερθεί ότι δεν παρατίθεται ο κώδικας για τις κλάσεις *WinningColumnStats*, *SingleDrawResults* και *MultiDrawResults* αφού και αυτές, όπως αναφέρθηκε προηγουμένως, δημιουργήθηκαν μηχανικά για την μετατροπή (casting) των JSON σε αντικείμενα JAVA.

|  |
| --- |
| **Πακέτο:** **model**  **Κλάση:** **GameDrawResults.Java** |
| //Custom Query to introduce game ID condition in search  **,** @NamedQuery**(**name **=** "GameDrawResults.findByDrawId"**,** query **=** "SELECT g FROM GameDrawResults g WHERE g.gameDrawResultsPK.drawId = :drawId and g.gameDrawResultsPK.gameId = :gameId"**)**  //Custom Query to retreive Draw Results info based on Winning category  **,** @NamedQuery**(**name **=** "GameDrawResults.findByWinningCategoryDescr"**,** query **=** "SELECT g FROM GameDrawResults g WHERE g.gameDrawResultsPK.winningCategoryDescr = :winningCategoryDescr"**)** |

|  |
| --- |
| **Πακέτο:** **model**  **Κλάση:** **GameDrawResults.Java** |
| //Custom Query to fetch Winning Column by Draw Id AND Game ID  **,** @NamedQuery**(**name **=** "WinningColumn.findByDrawId"**,** query **=** "SELECT w FROM WinningColumn w WHERE w.winningColumnPK.drawId = :drawId and w.winningColumnPK.gameId = :gameId"**)**  //Custom Query to retrieve all Winning Column info in given date range  **,** @NamedQuery**(**name **=** "WinningColumn.findByDrawTimeRange"**,** query **=** "SELECT w FROM WinningColumn w WHERE w.drawTime BETWEEN :drawTimeFrom AND :drawTimeTo"**)})** |

## Ερώτημα Β – Δημιουργία GUI Εφαρμογής

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:** **MenuOverlay.Java** |
| package gui**;**  **import** java**.**awt**.**Color**;**  **import** java**.**awt**.**Component**;**  **import** javax**.**swing**.**JLabel**;**  **import** javax**.**swing**.**JPanel**;**  public class MenuOverlay **{**    private JPanel selectePanelLevel01**;**    //Highlight on/off Main Menu upon mouseover  public void setHilighted**(**JPanel panel**,** boolean isHighlighted**){**  //Ignore highligt for selected panel  **if** **(!**panel**.**equals**(**selectePanelLevel01**)** **&&** panel**.**isEnabled**()){**  **if(**isHighlighted**){**  panel**.**setBackground**(new** Color**(**255**,**195**,**20**));**  **((**JLabel**)**panel**.**getComponent**(**0**)).**setForeground**(new** Color**(**51**,**51**,**51**));**  **}**  **else** **{**  panel**.**setBackground**(new** Color**(**192**,**139**,**6**));**  **((**JLabel**)**panel**.**getComponent**(**0**)).**setForeground**(new** Color**(**255**,**255**,**255**));**  **}**  **}**  **}**    //set selected Menu item  public void setSelected**(**JPanel panel**){**  selectePanelLevel01 **=** panel**;**  **}**  //get selected Menu item  public JPanel getSelected**(){**  **return** selectePanelLevel01**;**  **}**      public boolean isEnabled**(**JPanel panel**){**  **return** panel**.**isEnabled**();**  **}**      public void setEnabled**(**JPanel panel**,** boolean isEnabled**){**    **if** **(!**isEnabled**){**  panel**.**setBackground**(new** Color**(**192**,**139**,**6**));**  **}**    panel**.**setEnabled**(**isEnabled**);**  Component**[]** components **=** panel**.**getComponents**();**  **for(**Component comp **:** components**){**  comp**.**setEnabled**(**isEnabled**);**  **}**  **(**panel**.**getComponent**(**0**)).**setForeground**(new** Color**(**255**,**255**,**255**));**  **}**    public void deselectAll**(**JPanel parentPanel**){**    Component**[]** components **=** parentPanel**.**getComponents**();**  **for(**Component comp **:** components**){**  **if** **(**comp **instanceof** JPanel **){**  setHilighted**((**JPanel**)**comp**,** **false);**  **}**  **}**  **}**  **}** |

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:** **MainFrame.Java** |
| package gui**;**  **import** java**.**awt**.**Color**;**  **import** java**.**awt**.**Dimension**;**  **import** java**.**awt**.**Toolkit**;**  **import** java**.**io**.**File**;**  **import** java**.**io**.**IOException**;**  **import** java**.**net**.**InetAddress**;**  **import** java**.**sql**.**Connection**;**  **import** java**.**sql**.**DriverManager**;**  **import** java**.**sql**.**SQLException**;**  **import** java**.**util**.**Objects**;**  **import** java**.**util**.**logging**.**Level**;**  **import** java**.**util**.**logging**.**Logger**;**  **import** javax**.**swing**.**JDialog**;**  **import** javax**.**swing**.**JLabel**;**  **import** javax**.**swing**.**JOptionPane**;**  **import** javax**.**swing**.**JPanel**;**  public class MainFrame **extends** javax**.**swing**.**JFrame **{**  //Application Name  private final String APP\_NAME **=** "JokerGame-Stats"**;**  //Store Current mouse cursor position  private int xAxisMousePos**;**  private int yAxisMousePos**;**  private MenuOverlay menuOverlay **=** **new** MenuOverlay**();**      /\*\*  \* Creates new form MainMenu  \*/  public MainFrame**()** **{**  initComponents**();**  //Check DB connection during startup  String connectionString **=** "jdbc:derby://localhost/L10L;user=TzokerStats;password=pass"**;**  Connection connection **=** **null;**    **try** **{**  connection **=** DriverManager**.**getConnection**(**connectionString**);**  **}** **catch** **(**SQLException ex**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα κατα την σύνδεση στην βάση δεδομένων.\n" **+** ex**,**  "Πρόβλημα κατα την σύνδεση στον derby Server"**,** JOptionPane**.**ERROR\_MESSAGE**);**  **}**  **}**  private void formWindowOpened**(**java**.**awt**.**event**.**WindowEvent evt**)** **{**  setTitle**(**APP\_NAME**);**  getAccessibleContext**().**setAccessibleName**(**APP\_NAME**);**  **}**  private void btnMinimizeMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  setState**(**javax**.**swing**.**JFrame**.**ICONIFIED**);**  **}**  private void JPnlHeaderMousePressed**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  xAxisMousePos **=** evt**.**getX**();**  yAxisMousePos **=** evt**.**getY**();**  **}**  private void JPnlHeaderMouseDragged**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  int xPosOnScreen **=** evt**.**getXOnScreen**();**  int yPosOnScreen **=** evt**.**getYOnScreen**();**  **this.**setLocation**(**xPosOnScreen **-** xAxisMousePos**,** yPosOnScreen **-** yAxisMousePos**);**  **}**  private void btnCloseMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  closeApp**();**  **}**  private void JPnlGetDataMenuMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)**evt**.**getSource**(),** **true);**  **}**  private void JPnlGetDataMenuMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)**evt**.**getSource**(),** **false);**  **}**  private void JPnlMenuExitMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)**evt**.**getSource**(),** **true);**  **}**  private void JPnlMenuExitMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)**evt**.**getSource**(),** **false);**  **}**  private void JPnlMenuStatsMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)**evt**.**getSource**(),** **true);**  **}**  private void JPnlMenuStatsMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)**evt**.**getSource**(),** **false);**  **}**  private void JPnlMenuReportMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)**evt**.**getSource**(),** **true);**  **}**  private void JPnlMenuReportMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)**evt**.**getSource**(),** **false);**  **}**  private void JPnlGetDataMenuMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  //Prepare DataFrame and present it modally  showFrame**((**JPanel**)**evt**.**getSource**(),** **new** GetDataFrame**());**  **}**  private void JPnlMenuReportMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  //Present Joker Data  showFrame**((**JPanel**)**evt**.**getSource**(),** **new** DiplayDataByMonthFrame**());**  **}**  private void JPnlMenuExitMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  closeApp**();**  **}**  private void JPnlMenuStatsMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  //Present Joker Table and Chart Stats  showFrame**((**JPanel**)**evt**.**getSource**(),** **new** StatsMainFrame**());**  **}**  // Variables declaration - do not modify  private javax**.**swing**.**JLabel JLbBackground**;**  private javax**.**swing**.**JLabel JLbDisplayDataIcon**;**  private javax**.**swing**.**JLabel JLbDisplayDataText**;**  private javax**.**swing**.**JLabel JLbDisplayStatsIcon**;**  private javax**.**swing**.**JLabel JLbDisplayStatsText**;**  private javax**.**swing**.**JLabel JLbExitIcon**;**  private javax**.**swing**.**JLabel JLbExitText**;**  private javax**.**swing**.**JLabel JLbFootnote**;**  private javax**.**swing**.**JLabel JLblGetDataIcon**;**  private javax**.**swing**.**JLabel JLblGetDataTitle**;**  private javax**.**swing**.**JPanel JPnlGetDataMenu**;**  private javax**.**swing**.**JPanel JPnlHeader**;**  private javax**.**swing**.**JPanel JPnlMain**;**  private javax**.**swing**.**JPanel JPnlMainMenu**;**  private javax**.**swing**.**JPanel JPnlMenuExit**;**  private javax**.**swing**.**JPanel JPnlMenuReport**;**  private javax**.**swing**.**JPanel JPnlMenuStats**;**  private javax**.**swing**.**JPanel JPnlWorkingPanel**;**  private javax**.**swing**.**JButton btnClose**;**  private javax**.**swing**.**JButton btnMinimize**;**  private javax**.**swing**.**ButtonGroup buttonGroup1**;**  // End of variables declaration  private void showFrame**(**JPanel selectedPanel**,** javax**.**swing**.**JFrame frame**)** **{**  //Present Joker Data  **if** **(**Objects**.**isNull**(**menuOverlay**.**getSelected**())** **||**  **!**menuOverlay**.**getSelected**().**equals**(**selectedPanel**)){**  //Add the selected panel in the main window  JPnlWorkingPanel**.**removeAll**();**  JPnlWorkingPanel**.**repaint**();**  JPnlWorkingPanel**.**revalidate**();**  JPnlWorkingPanel**.**add**(**frame**.**getContentPane**());**  JPnlWorkingPanel**.**repaint**();**  JPnlWorkingPanel**.**revalidate**();**  //Make current item menu active  menuOverlay**.**setSelected**(**selectedPanel**);**    //deselect other  menuOverlay**.**deselectAll**(**JPnlMainMenu**);**  **}**  **}**    /\*  Gracefully close app  \*/  private void closeApp**(){**  Object**[]** options **=** **{**"Όχι"**,** "Ναι"**};**  int userSelection **=** JOptionPane**.**showOptionDialog**(null,**  "Θέλετε σίγουρα να κλείσετε την εφαρμογή;"**,**  APP\_NAME**,**  JOptionPane**.**YES\_NO\_OPTION**,**  JOptionPane**.**QUESTION\_MESSAGE**,**  **null,**  options**,**  options**[**0**]);**  **if** **(**userSelection **==** 1**)** **{**  **try** **{**  //Stop DerbyDB engine  Process process **=** Runtime**.**getRuntime**().**exec**(**"cmd /c start /b stopNetworkServer.bat"**,** **null,** **new** File**(**"db"**));**  **}** **catch** **(**IOException ex**)** **{**  Logger**.**getLogger**(**MainFrame**.**class**.**getName**()).**log**(**Level**.**SEVERE**,** **null,** ex**);**  **}finally** **{**  System**.**exit**(**0**);**  **}**  **}**  **}**  **}** |

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:** **GetDataFrame.Java** |
| package gui**;**  **import** DataIngestion**.**DBIntegrator**;**  **import** DataIngestion**.**IApiIntegrator**;**  **import** DataIngestion**.**MultiDrawIntegrator**;**  **import** DataIngestion**.**SingleDrawIntegrator**;**  **import** javax**.**swing**.**JOptionPane**;**  **import** java**.**awt**.**Cursor**;**  **import** java**.**awt**.**Rectangle**;**  **import** java**.**text**.**DateFormat**;**  **import** java**.**text**.**DecimalFormat**;**  **import** java**.**text**.**SimpleDateFormat**;**  **import** java**.**util**.**Calendar**;**  **import** java**.**util**.**Date**;**  **import** java**.**util**.**List**;**  **import** java**.**util**.**Objects**;**  **import** java**.**util**.**concurrent**.**TimeUnit**;**  **import** javax**.**swing**.**JLabel**;**  **import** javax**.**swing**.**JPanel**;**  **import** javax**.**swing**.**JSpinner**;**  **import** javax**.**swing**.**SpinnerModel**;**  **import** javax**.**swing**.**SpinnerNumberModel**;**  **import** javax**.**swing**.**SwingWorker**;**  **import** javax**.**swing**.**table**.**DefaultTableCellRenderer**;**  **import** javax**.**swing**.**table**.**DefaultTableModel**;**  **import** javax**.**swing**.**table**.**TableModel**;**  **import** model**.**GameDrawResults**;**  **import** model**.**WinningColumn**;**  **import** tzokerstats**.**FoundEntityException**;**  **import** tzokerstats**.**GUIValidationException**;**  **import** tzokerstats**.**NotFoundEntityException**;**  public class GetDataFrame **extends** javax**.**swing**.**JFrame **{**  //List of API Responce objects for a DRAW  private List**<**GameDrawResults**>** gameDrawResults**;**  //List of API Responce for winning numbers  private List**<**WinningColumn**>** winningColumns**;**  private MenuOverlay menuOverlay **=** **new** MenuOverlay**();**  /\*\*  \* Creates new form GetDataFrame  \*/  public GetDataFrame**()** **{**  initComponents**();**  **}**  private void JPnlGetDataFromAPIMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **if** **(!**JPnlGetDataFromAPI**.**isEnabled**())** **{**  **return;**  **}**  menuOverlay**.**setEnabled**(**JPnlInsertToDB**,** **false);**  menuOverlay**.**setEnabled**(**JPnlDeleteFromDB**,** **false);**  menuOverlay**.**setEnabled**(**JPnlDeleteAllDB**,** **false);**  startWaiting**((**JPanel**)** evt**.**getSource**());**  //Main import data from API task. Executed in background thread  SwingWorker**<**Void**,** Void**>** mySwingWorker **=** **new** SwingWorker**<**Void**,** Void**>()** **{**  @Override  protected Void doInBackground**()** **throws** Exception **{**  **try** **{**  //Get Selected options  String selectedDrawID **=** **(((**JSpinner**.**DefaultEditor**)** JSpnrDrawID**.**getEditor**()).**getTextField**()).**getText**();**  String fromDate **=** ""**,** toDate **=** ""**;**  //Due to api limitations we may perform multiple calls  //for a sing date range requested by user  int howManyAPICalls **=** 1**;**  //Search by dates use case  **if** **(**RadiofetchDataByDateRange**.**isSelected**())** **{**  //User input validation  //From Date  jXDatePickerFrom**.**commitEdit**();**  **if** **(**jXDatePickerFrom**.**getDate**()** **==** **null)** **{**  **throw** **new** GUIValidationException**(**"Παρακαλώ εισάγετε αρχική ημερομηνία."**);**  **}**  DateFormat dateFormat **=** **new** SimpleDateFormat**(**"yyyy-MM-dd"**);**  fromDate **=** dateFormat**.**format**(**jXDatePickerFrom**.**getDate**());**  //ToDate  jXDatePickerTo**.**commitEdit**();**  **if** **(**jXDatePickerTo**.**getDate**()** **==** **null)** **{**  **throw** **new** GUIValidationException**(**"Παρακαλώ εισάγετε τελική ημερομηνία."**);**  **}**  toDate **=** dateFormat**.**format**(**jXDatePickerTo**.**getDate**());**  Date selectedDateFrom **=** jXDatePickerFrom**.**getDate**();**  Date selectedDateTo **=** jXDatePickerTo**.**getDate**();**  howManyAPICalls **=** **(**getDateDiff**(**selectedDateFrom**,** selectedDateTo**)** **/** 15**)** **+** 1**;**  **}**  //Change UI to depict work done in the backround  menuOverlay**.**setEnabled**(**JPnlGetDataFromAPI**,** **false);**  //Format Tables  DefaultTableCellRenderer rightRenderer **=** **new** DefaultTableCellRenderer**();**  DefaultTableCellRenderer centerRenderer **=** **new** DefaultTableCellRenderer**();**  rightRenderer**.**setHorizontalAlignment**(**JLabel**.**RIGHT**);**  centerRenderer**.**setHorizontalAlignment**(**JLabel**.**CENTER**);**  JTblDetailsDrawTable**.**getColumnModel**().**getColumn**(**0**).**setCellRenderer**(**centerRenderer**);**  JTblDetailsDrawTable**.**getColumnModel**().**getColumn**(**1**).**setCellRenderer**(**centerRenderer**);**  JTblDetailsDrawTable**.**getColumnModel**().**getColumn**(**2**).**setCellRenderer**(**rightRenderer**);**  JTblDetailsDrawTable**.**getColumnModel**().**getColumn**(**3**).**setCellRenderer**(**rightRenderer**);**  JTblSummaryDrawTable**.**getColumnModel**().**getColumn**(**0**).**setCellRenderer**(**centerRenderer**);**  //Prepare table models  DefaultTableModel detailsDrawModel **=** **(**DefaultTableModel**)** JTblDetailsDrawTable**.**getModel**();**  DefaultTableModel summaryDrawModel **=** **(**DefaultTableModel**)** JTblSummaryDrawTable**.**getModel**();**  detailsDrawModel**.**setRowCount**(**0**);**  summaryDrawModel**.**setRowCount**(**0**);**  **if** **(!**Objects**.**isNull**(**gameDrawResults**))** **{**  gameDrawResults**.**clear**();**  **}**  **if** **(!**Objects**.**isNull**(**winningColumns**))** **{**  winningColumns**.**clear**();**  **}**  //Split API Calls if this is the case  //of a large date range selection by the user.  **for** **(**int j **=** 0**;** j **<** howManyAPICalls**;** j**++)** **{**  //We use polymorphism to get data from API  IApiIntegrator drawIntegrator**;**  **if** **(**RadiofetchDataByID**.**isSelected**())** **{**  IApiIntegrator singleDrawIntegrator **=** **new** SingleDrawIntegrator**();**  singleDrawIntegrator**.**addAPIargument**(**"gameID"**,** "5104"**);**  singleDrawIntegrator**.**addAPIargument**(**"drawID"**,** selectedDrawID**);**  drawIntegrator **=** singleDrawIntegrator**;**  **}** **else** **{**  Calendar cFrom **=** Calendar**.**getInstance**();**  Calendar cTo **=** Calendar**.**getInstance**();**  cFrom**.**setTime**(**jXDatePickerFrom**.**getDate**());**  DateFormat dateFormat **=** **new** SimpleDateFormat**(**"yyyy-MM-dd"**);**  cFrom**.**add**(**Calendar**.**DATE**,** **(**15 **\*** j**));**  fromDate **=** dateFormat**.**format**(**cFrom**.**getTime**());**  cTo **=** cFrom**;**  cTo**.**add**(**Calendar**.**DATE**,** 14**);**  **if** **(**cTo**.**getTime**().**after**(**jXDatePickerTo**.**getDate**()))** **{**  cTo**.**setTime**(**jXDatePickerTo**.**getDate**());**  **}**  toDate **=** dateFormat**.**format**(**cTo**.**getTime**());**  //Get Multidraw objects  IApiIntegrator multiDrawIntegrator **=** **new** MultiDrawIntegrator**();**  multiDrawIntegrator**.**addAPIargument**(**"gameID"**,** "5104"**);**  multiDrawIntegrator**.**addAPIargument**(**"fromDate"**,** fromDate**);**  multiDrawIntegrator**.**addAPIargument**(**"toDate"**,** toDate**);**  drawIntegrator **=** multiDrawIntegrator**;**  **}**  //Fetch data  drawIntegrator**.**loadDataFromAPI**();**  List**<**GameDrawResults**>** gameDrawResults\_Current **=** **(**List**<**GameDrawResults**>)** drawIntegrator**.**getDataObject**(**GameDrawResults**.**class**);**  List**<**WinningColumn**>** winningColumns\_Current **=** **(**List**<**WinningColumn**>)** drawIntegrator**.**getDataObject**(**WinningColumn**.**class**);**  **if** **(**Objects**.**isNull**(**gameDrawResults**))** **{**  gameDrawResults **=** **(**List**<**GameDrawResults**>)** drawIntegrator**.**getDataObject**(**GameDrawResults**.**class**);**  **}** **else** **{**  //Append at the top to maintain table sorting on draw\_id, Win Category  gameDrawResults**.**addAll**(**0**,** gameDrawResults\_Current**);**  **}**  **if** **(**Objects**.**isNull**(**winningColumns**))** **{**  winningColumns **=** **(**List**<**WinningColumn**>)** drawIntegrator**.**getDataObject**(**WinningColumn**.**class**);**  **}** **else** **{**  //Append at the top to maintain sorting on draw\_id  winningColumns**.**addAll**(**0**,** winningColumns\_Current**);**  **}**  **}**  //Check Results  **if** **(**gameDrawResults**.**size**()** **<** 1**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Δεν βρέθηκαν δεδομένα με τα συγκεκριμένα κριτήρια. "**,**  "Αντληση δεδομένων"**,**  JOptionPane**.**INFORMATION\_MESSAGE**);**  menuOverlay**.**setEnabled**(**JPnlInsertToDB**,** **false);**  menuOverlay**.**setEnabled**(**JPnlDeleteFromDB**,** **false);**  menuOverlay**.**setEnabled**(**JPnlDeleteAllDB**,** **true);**  **}** **else** **{**  //Prepare Formaters  Object**[]** detailsDrawData **=** **new** Object**[**4**];**  Object**[]** summaryDrawData **=** **new** Object**[**3**];**  DecimalFormat decimalFormat **=** **new** DecimalFormat**(**"#,###.00"**);**  DecimalFormat IntFormatter **=** **new** DecimalFormat**(**"#,###"**);**  SimpleDateFormat simpleDateFormat **=** **new** SimpleDateFormat**(**"dd/MM/yyyy"**);**  //add table rows for Details table  **for** **(**int i **=** 0**;** i **<** gameDrawResults**.**size**();** i**++)** **{**  detailsDrawData**[**0**]** **=** gameDrawResults**.**get**(**i**).**getGameDrawResultsPK**().**getDrawId**();**  detailsDrawData**[**1**]** **=** gameDrawResults**.**get**(**i**).**getGameDrawResultsPK**().**getWinningCategoryDescr**();**  detailsDrawData**[**2**]** **=** IntFormatter**.**format**((**Number**)** gameDrawResults**.**get**(**i**).**getSuccessesCnt**());**  detailsDrawData**[**3**]** **=** decimalFormat**.**format**((**Number**)** gameDrawResults**.**get**(**i**).**getDividentAmn**());**  detailsDrawModel**.**addRow**(**detailsDrawData**);**  **}**  //add rows for Summary Draw Table  **for** **(**int i **=** 0**;** i **<** winningColumns**.**size**();** i**++)** **{**  summaryDrawData**[**0**]** **=** winningColumns**.**get**(**i**).**getWinningColumnPK**().**getDrawId**();**  summaryDrawData**[**1**]** **=** simpleDateFormat**.**format**(**winningColumns**.**get**(**i**).**getDrawTime**());**  summaryDrawData**[**2**]** **=** winningColumns**.**get**(**i**).**getWinningColumnDescr**();**  summaryDrawModel**.**addRow**(**summaryDrawData**);**  **}**  //Enable menu option based on results rowcount  menuOverlay**.**setEnabled**(**JPnlInsertToDB**,** detailsDrawModel**.**getRowCount**()** **>** 0**);**  menuOverlay**.**setEnabled**(**JPnlDeleteFromDB**,** detailsDrawModel**.**getRowCount**()** **>** 0**);**  System**.**out**.**println**(**"Data Fetch operation completed Succesfully!"**);**  **}**  **}** **catch** **(**GUIValidationException e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα κατα την άντληση των δεδομένων. " **+** e**.**getMessage**(),**  "Σφάλμα κατα την άντληση δεδομένων"**,**  JOptionPane**.**ERROR\_MESSAGE**);**  **}** **catch** **(**Exception e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα κατα την άντληση των δεδομένων." **+** e**.**getMessage**(),** "Δεν Αντλήθηκαν δεδομένα."**,** JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  **}** **finally** **{**  //Restore initial UI status  stopWaiting**();**  menuOverlay**.**setEnabled**((**JPanel**)** evt**.**getSource**(),** **true);**  menuOverlay**.**setEnabled**(**JPnlDeleteFromDB**,** **true);**  menuOverlay**.**setEnabled**(**JPnlDeleteAllDB**,** **true);**  **}**  **return** **null;**  **}**  **};**  mySwingWorker**.**execute**();**  **}**  private void JPnlGetDataFromAPIMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlGetDataFromAPIMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  private void JPnlInsertToDBMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlInsertToDBMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  private void JPnlDeleteFromDBMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlDeleteFromDBMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  private void JPnlInsertToDBMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **if** **(!**JPnlInsertToDB**.**isEnabled**())** **{**  **return;**  **}**  menuOverlay**.**setEnabled**(**JPnlGetDataFromAPI**,** **false);**  menuOverlay**.**setEnabled**(**JPnlDeleteFromDB**,** **false);**  menuOverlay**.**setEnabled**(**JPnlDeleteAllDB**,** **false);**  startWaiting**((**JPanel**)** evt**.**getSource**());**  //Save data to DB. Executed in background thread  SwingWorker**<**Void**,** Void**>** mySwingWorker **=** **new** SwingWorker**<**Void**,** Void**>()** **{**  @Override  protected Void doInBackground**()** **throws** Exception **{**  **try** **{**  menuOverlay**.**setEnabled**((**JPanel**)** evt**.**getSource**(),** **false);**  **if** **(!**DBIntegrator**.**isDrawResultsReadyForInsert**(**winningColumns**,** gameDrawResults**))** **{**  //there are rows allready in the DB  Object**[]** options **=** **{**"Όχι"**,** "Ναι"**};**  int userSelection **=** JOptionPane**.**showOptionDialog**(null,**  "Κάποιες κληρώσεις βρέθηκαν στην βάση. Θέλετε να εισηχθούν όσες ΔΕΝ βρέθηκαν;"**,**  "Μερική εισαγωγή κληρώσεων στην βάση"**,**  JOptionPane**.**YES\_NO\_OPTION**,**  JOptionPane**.**QUESTION\_MESSAGE**,**  **null,**  options**,**  options**[**0**]);**  **if** **(**userSelection **==** 1**)** **{**  //Insert applicable only Draws  //Attempt to write to DB  DBIntegrator**.**setDrawResults**(**winningColumns**,** gameDrawResults**,** DBIntegrator**.**Operation**.**FORCE\_INSERT**);**  JOptionPane**.**showMessageDialog**(null,** "H μερική εισαγωγή των δεδομένων ολοκληρώθηκε με επιτυχία"**,**  "Μερική εισαγωγή κληρώσεων στην βάση"**,** JOptionPane**.**INFORMATION\_MESSAGE**);**  **}** **else** **{**  JOptionPane**.**showMessageDialog**(null,** "H εισαγωγή των δεδομένων ακυρώθηκε"**,**  "Δεν Εισήχθησαν Δεδομένα"**,** JOptionPane**.**INFORMATION\_MESSAGE**);**  **}**  **}** //All rows can be inserted  **else** **{**  //Attempt to insert data to DB  DBIntegrator**.**setDrawResults**(**winningColumns**,** gameDrawResults**,** DBIntegrator**.**Operation**.**INSERT**);**  JOptionPane**.**showMessageDialog**(null,** "H εισαγωγή των δεδομένων ολοκληρώθηκε με επιτυχία"**,**  "Τα Δεδομένα Εισηχθησαν"**,** JOptionPane**.**INFORMATION\_MESSAGE**);**  **}**  **}** **catch** **(**FoundEntityException e**)** **{**  JOptionPane**.**showMessageDialog**(null,** e**.**getMessage**(),** "Δεν Εισήχθησαν Δεδομένα"**,** JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  **}** **catch** **(**Exception e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα κατα την εισαγωγή των δεδομένων στην βάση:\n"  **+** e**.**getMessage**(),** "Δεν Εισήχθησαν Δεδομένα"**,** JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  **}** **finally** **{**  //Restore initial UI status  stopWaiting**();**  menuOverlay**.**setEnabled**((**JPanel**)** evt**.**getSource**(),** **true);**  menuOverlay**.**setEnabled**(**JPnlDeleteAllDB**,** **true);**  menuOverlay**.**setEnabled**(**JPnlDeleteFromDB**,** **true);**  menuOverlay**.**setEnabled**(**JPnlGetDataFromAPI**,** **true);**  **}**  **return** **null;**  **}**  **};**  mySwingWorker**.**execute**();**  **}**  private void JTblDetailsDrawTableMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  int column **=** 0**;**  int row **=** JTblDetailsDrawTable**.**getSelectedRow**();**  String value **=** JTblDetailsDrawTable**.**getModel**().**getValueAt**(**row**,** column**).**toString**();**  int i **=** getRowByValue**(**JTblSummaryDrawTable**.**getModel**(),** value**);**  JTblSummaryDrawTable**.**getSelectionModel**().**setSelectionInterval**(**i**,** i**);**  JTblSummaryDrawTable**.**scrollRectToVisible**(new** Rectangle**(**JTblSummaryDrawTable**.**getCellRect**(**i**,** 0**,** **true)));**  **}**  private void JPnlDeleteFromDBMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **if** **(!**JPnlDeleteFromDB**.**isEnabled**())** **{**  **return;**  **}**  Object**[]** options **=** **{**"Όχι"**,** "Ναι"**};**  int userSelection **=** JOptionPane**.**showOptionDialog**(null,**  "Θέλετε σίγουρα να διαγράψετε τα δεδομένα των κληρώσεων από τη βάση;"**,**  "Διαγραφή των κληρώσεων απο την βάση"**,**  JOptionPane**.**YES\_NO\_OPTION**,**  JOptionPane**.**QUESTION\_MESSAGE**,**  **null,**  options**,**  options**[**0**]);**  **if** **(**userSelection **==** 1**)** **{**  //Delete for applicable only Draws  menuOverlay**.**setEnabled**(**JPnlGetDataFromAPI**,** **false);**  menuOverlay**.**setEnabled**(**JPnlDeleteAllDB**,** **false);**  menuOverlay**.**setEnabled**(**JPnlInsertToDB**,** **false);**  startWaiting**((**JPanel**)** evt**.**getSource**());**  //Executed in background thread  SwingWorker**<**Void**,** Void**>** mySwingWorker **=** **new** SwingWorker**<**Void**,** Void**>()** **{**  @Override  protected Void doInBackground**()** **throws** Exception **{**  **try** **{**  menuOverlay**.**setEnabled**((**JPanel**)** evt**.**getSource**(),** **false);**  //Get Selected options  String selectedDrawID **=** **(((**JSpinner**.**DefaultEditor**)** JSpnrDrawID**.**getEditor**()).**getTextField**()).**getText**();**  String fromDate **=** ""**;**  String toDate **=** ""**;**  **if** **(**RadiofetchDataByID**.**isSelected**())** **{**  int DrawsDeletedByID **=** DBIntegrator**.**deleteDrawDataWithID**(**"5104"**,** selectedDrawID**);**  JOptionPane**.**showMessageDialog**(null,** "H διαγραφή των δεδομένων με βάση το Draw ID ολοκληρώθηκε με επιτυχία. \nΔιαγράφηκαν τα δεδομένα για την κλήρωση #" **+** selectedDrawID**,**  "Τα δεδομένα Διαγράφηκαν"**,** JOptionPane**.**INFORMATION\_MESSAGE**);**  **}** **else** **if** **(**RadiofetchDataByDateRange**.**isSelected**())** **{**  jXDatePickerFrom**.**commitEdit**();**  **if** **(**jXDatePickerFrom**.**getDate**()** **==** **null)** **{**  **throw** **new** GUIValidationException**(**"Παρακαλώ εισάγετε αρχική ημερομηνία."**);**  **}**  DateFormat dateFormat **=** **new** SimpleDateFormat**(**"yyyy-MM-dd"**);**  fromDate **=** dateFormat**.**format**(**jXDatePickerFrom**.**getDate**());**  //ToDate  jXDatePickerTo**.**commitEdit**();**  **if** **(**jXDatePickerTo**.**getDate**()** **==** **null)** **{**  **throw** **new** GUIValidationException**(**"Παρακαλώ εισάγετε τελική ημερομηνία."**);**  **}**  toDate **=** dateFormat**.**format**(**jXDatePickerTo**.**getDate**());**  int DrawsDeletedByDate **=** DBIntegrator**.**deleteDrawDataWithDateRange**(**"5104"**,** fromDate**,** toDate**);**  JOptionPane**.**showMessageDialog**(null,** "H διαγραφή των δεδομένων με βάση χρονικό διάστημα ολοκληρώθηκε με επιτυχία. \nΔιαγράφηκαν τα δεδομένα για " **+** DrawsDeletedByDate **+** " κληρώσεις"**,**  "Τα δεδομένα Διαγράφηκαν"**,** JOptionPane**.**INFORMATION\_MESSAGE**);**  **}**  **}** **catch** **(**NotFoundEntityException e**)** **{**  JOptionPane**.**showMessageDialog**(null,** e**.**getMessage**(),** "Δεν Διαγράφηκαν Δεδομένα"**,** JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  **}** **catch** **(**Exception e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα κατα την διαγραφή των δεδομένων απο την βάση.\n" **+** e**.**getMessage**(),**  "Δεν Διαγράφηκαν Δεδομένα"**,** JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  **}** **finally** **{**  //Restore initial UI status  menuOverlay**.**setEnabled**((**JPanel**)** evt**.**getSource**(),** **true);**  menuOverlay**.**setEnabled**(**JPnlGetDataFromAPI**,** **true);**  menuOverlay**.**setEnabled**(**JPnlDeleteAllDB**,** **true);**  **if** **(!(**Objects**.**isNull**(**gameDrawResults**)** **||** gameDrawResults**.**size**()** **<** 1**))** **{**  menuOverlay**.**setEnabled**(**JPnlInsertToDB**,** **true);**  **}**  stopWaiting**();**  **}**  **return** **null;**  **}**  **};**  mySwingWorker**.**execute**();**  **}**  **}**  private void JPnlDeleteAllDBMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **if** **(!**JPnlDeleteAllDB**.**isEnabled**())** **{**  **return;**  **}**  Object**[]** options **=** **{**"Όχι δεν θελω να διαγράψω τα πάντα"**,** "Ναι"**};**  int userSelection **=** JOptionPane**.**showOptionDialog**(null,**  "Θέλετε σίγουρα να διαγράψετε ολες τις κληρώσεις από τη βάση δεδομένων;"**,**  "Διαγραφή ΟΛΩΝ των κληρώσεων απο την βάση"**,**  JOptionPane**.**YES\_NO\_OPTION**,**  JOptionPane**.**QUESTION\_MESSAGE**,**  **null,**  options**,**  options**[**0**]);**  **if** **(**userSelection **==** 1**)** **{**  menuOverlay**.**setEnabled**(**JPnlInsertToDB**,** **false);**  menuOverlay**.**setEnabled**(**JPnlGetDataFromAPI**,** **false);**  menuOverlay**.**setEnabled**(**JPnlDeleteFromDB**,** **false);**  startWaiting**((**JPanel**)** evt**.**getSource**());**  SwingWorker**<**Void**,** Void**>** mySwingWorker **=** **new** SwingWorker**<**Void**,** Void**>()** **{**  @Override  protected Void doInBackground**()** **throws** Exception **{**  **try** **{**  menuOverlay**.**setEnabled**((**JPanel**)** evt**.**getSource**(),** **false);**  DBIntegrator**.**deleteALLDrawResults**();**  clearTables**();**  JOptionPane**.**showMessageDialog**(null,** "H Διαγραφή ΟΛΩΝ των δεδομένων ολοκληρώθηκε με επιτυχία"**,**  "Τα δεδομένα Διαγράφηκαν"**,** JOptionPane**.**INFORMATION\_MESSAGE**);**  **}** **catch** **(**Exception e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα κατα την διαγραφή των δεδομένων απο την βάση"**,**  "Δεν Διαγράφηκαν Δεδομένα"**,** JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  **}** **finally** **{**  stopWaiting**();**  menuOverlay**.**setEnabled**(**JPnlInsertToDB**,** **false);**  menuOverlay**.**setEnabled**(**JPnlDeleteFromDB**,** **false);**  menuOverlay**.**setEnabled**((**JPanel**)** evt**.**getSource**(),** **true);**  menuOverlay**.**setEnabled**(**JPnlGetDataFromAPI**,** **true);**  **}**  **return** **null;**  **}**  **};**  mySwingWorker**.**execute**();**  **}**  **}**  private void JPnlDeleteAllDBMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlDeleteAllDBMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  private void RadiofetchDataByDateRangeItemStateChanged**(**java**.**awt**.**event**.**ItemEvent evt**)** **{**  **if** **(**evt**.**getStateChange**()** **==** evt**.**SELECTED**)** **{**  menuOverlay**.**setEnabled**(**JPnlSearchByDate**,** **true);**  menuOverlay**.**setEnabled**(**JPnlSearchlByID**,** **false);**  **}** **else** **if** **(**evt**.**getStateChange**()** **==** evt**.**DESELECTED**)** **{**  menuOverlay**.**setEnabled**(**JPnlSearchByDate**,** **false);**  menuOverlay**.**setEnabled**(**JPnlSearchlByID**,** **true);**  **}**  **}**  private void JTblSummaryDrawTableMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  int column **=** 0**;**  int row **=** JTblSummaryDrawTable**.**getSelectedRow**();**  String value **=** JTblSummaryDrawTable**.**getModel**().**getValueAt**(**row**,** column**).**toString**();**  int i **=** getRowByValue**(**JTblDetailsDrawTable**.**getModel**(),** value**);**  JTblDetailsDrawTable**.**getSelectionModel**().**setSelectionInterval**(**i**,** i**);**  JTblDetailsDrawTable**.**scrollRectToVisible**(new** Rectangle**(**JTblDetailsDrawTable**.**getCellRect**(**i**,** 0**,** **true)));**  **}**  private int getRowByValue**(**TableModel model**,** String value**)** **{**  **for** **(**int i **=** 0**;** i **<** model**.**getRowCount**();** i**++)** **{**  **if** **(**model**.**getValueAt**(**i**,** 0**).**toString**().**equals**(**value**))** **{**  // return the index of the first row for thee value  **return** i**;**  **}**  **}**  **return** **-**1**;**  **}**  private void startWaiting**(**JPanel panel**)** **{**  progressBar**.**setVisible**(true);**  progressBar**.**setIndeterminate**(true);**  progressBar**.**setValue**(**100**);**  JPnlTzokerDataPanel**.**setCursor**(**Cursor**.**getPredefinedCursor**(**Cursor**.**WAIT\_CURSOR**));**  panel**.**grabFocus**();**  **}**  private void stopWaiting**()** **{**  JPnlTzokerDataPanel**.**setCursor**(null);** //turn off the wait cursor  progressBar**.**setIndeterminate**(false);**  progressBar**.**setVisible**(false);**  **}**  private void clearTables**()** **{**  DefaultTableModel detailsDrawModel **=** **(**DefaultTableModel**)** JTblDetailsDrawTable**.**getModel**();**  DefaultTableModel summaryDrawModel **=** **(**DefaultTableModel**)** JTblSummaryDrawTable**.**getModel**();**  detailsDrawModel**.**setRowCount**(**0**);**  summaryDrawModel**.**setRowCount**(**0**);**  **}**  // Variables declaration - do not modify  private javax**.**swing**.**JLabel JLblDeleteAllDBIcon**;**  private javax**.**swing**.**JLabel JLblDeleteAllDBText**;**  private javax**.**swing**.**JLabel JLblDeleteFromDBIcon**;**  private javax**.**swing**.**JLabel JLblDeleteFromDBText**;**  private javax**.**swing**.**JLabel JLblDrawID**;**  private javax**.**swing**.**JLabel JLblGetDataFromApiIcon**;**  private javax**.**swing**.**JLabel JLblGetDataFromApiText**;**  private javax**.**swing**.**JLabel JLblInsertToDBIcon**;**  private javax**.**swing**.**JLabel JLblInsertToDBText**;**  private javax**.**swing**.**JPanel JPnlDeleteAllDB**;**  private javax**.**swing**.**JPanel JPnlDeleteFromDB**;**  private javax**.**swing**.**JPanel JPnlGetDataFromAPI**;**  private javax**.**swing**.**JPanel JPnlInsertToDB**;**  private javax**.**swing**.**JPanel JPnlSearchByDate**;**  private javax**.**swing**.**JPanel JPnlSearchlByID**;**  private javax**.**swing**.**JPanel JPnlSeperator01**;**  private javax**.**swing**.**JPanel JPnlSeperator02**;**  private javax**.**swing**.**JPanel JPnlTables**;**  private javax**.**swing**.**JPanel JPnlTzokerDataPanel**;**  private javax**.**swing**.**JScrollPane JScrollDetailsDrawTable**;**  private javax**.**swing**.**JScrollPane JScrollSummaryDrawTable**;**  private javax**.**swing**.**JSpinner JSpnrDrawID**;**  private javax**.**swing**.**JTable JTblDetailsDrawTable**;**  private javax**.**swing**.**JTable JTblSummaryDrawTable**;**  private javax**.**swing**.**JLabel JlblDateFrom**;**  private javax**.**swing**.**JLabel JlblDateTo**;**  private javax**.**swing**.**JRadioButton RadiofetchDataByDateRange**;**  private javax**.**swing**.**JRadioButton RadiofetchDataByID**;**  private javax**.**swing**.**ButtonGroup btnGrpSelectionOption**;**  private org**.**jdesktop**.**swingx**.**JXDatePicker jXDatePickerFrom**;**  private org**.**jdesktop**.**swingx**.**JXDatePicker jXDatePickerTo**;**  private javax**.**swing**.**JProgressBar progressBar**;**  // End of variables declaration  //Prepare panel UI on post init  private void setDefaults**()** **{**  progressBar**.**setVisible**(false);**  JPnlInsertToDB**.**setEnabled**(false);**  SpinnerModel sm **=** **new** SpinnerNumberModel**(**1**,** 1**,** 999999**,** 1**);**  JSpnrDrawID**.**setModel**(**sm**);**  JSpnrDrawID**.**setEditor**(new** JSpinner**.**NumberEditor**(**JSpnrDrawID**,** "#"**));**  jXDatePickerFrom**.**setFormats**(new** SimpleDateFormat**(**"dd/MM/yyyy"**));**  jXDatePickerTo**.**setFormats**(new** SimpleDateFormat**(**"dd/MM/yyyy"**));**  **}**  private int getDateDiff**(**Date firstDate**,** Date secondDate**)** **{**  long diffInMillies **=** Math**.**abs**(**secondDate**.**getTime**()** **-** firstDate**.**getTime**());**  **return** **(**int**)** TimeUnit**.**DAYS**.**convert**(**diffInMillies**,** TimeUnit**.**MILLISECONDS**);**  **}**  **}** |

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:** **DisplayDatabyMonthFrame.Java** |
| package gui**;**  **import** DataIngestion**.**DBIntegrator**;**  **import** DataIngestion**.**IApiIntegrator**;**  **import** DataIngestion**.**MultiDrawIntegrator**;**  **import** java**.**awt**.**Cursor**;**  **import** java**.**awt**.**Dimension**;**  **import** java**.**text**.**DateFormat**;**  **import** java**.**text**.**SimpleDateFormat**;**  **import** java**.**util**.**Calendar**;**  **import** java**.**util**.**List**;**  **import** java**.**util**.**Objects**;**  **import** javax**.**swing**.**JLabel**;**  **import** javax**.**swing**.**JOptionPane**;**  **import** javax**.**swing**.**JPanel**;**  **import** javax**.**swing**.**SwingWorker**;**  **import** model**.**GameDrawResults**;**  **import** model**.**WinningColumn**;**  public class DiplayDataByMonthFrame **extends** javax**.**swing**.**JFrame **{**  private MenuOverlay menuOverlay **=** **new** MenuOverlay**();**  private List**<**GameDrawResults**>** gameDrawResults**;**  //List of API Responce for winning numbers  private List**<**WinningColumn**>** winningColumns**;**  /\*\*  \* Creates new form NewJFrame  \*/  public DiplayDataByMonthFrame**()** **{**  initComponents**();**  setComboValues**();**  progressBar**.**setVisible**(false);**  JLblPayoutTitle**.**setHorizontalAlignment**(**JLabel**.**CENTER**);**  JLblChartByMonthTitle**.**setHorizontalAlignment**(**JLabel**.**CENTER**);**  **}**    /\*\*  \* This method is called from within the constructor to initialize the form.  \* WARNING: Do NOT modify this code. The content of this method is always  \* regenerated by the Form Editor.  \*/  @SuppressWarnings**(**"unchecked"**)**  // <editor-fold defaultstate="collapsed" desc="Generated Code">  private void initComponents**()** **{**  JPnlTzokerDataPanel **=** **new** javax**.**swing**.**JPanel**();**  JPnlChartByMonth **=** **new** javax**.**swing**.**JPanel**();**  JPnlSelectYear **=** **new** javax**.**swing**.**JPanel**();**  JLblYear **=** **new** javax**.**swing**.**JLabel**();**  JCmbYear **=** **new** javax**.**swing**.**JComboBox**<>();**  JLblFiller **=** **new** javax**.**swing**.**JLabel**();**  JPnlGetAggregatedData **=** **new** javax**.**swing**.**JPanel**();**  JLblExecuteIcon **=** **new** javax**.**swing**.**JLabel**();**  JLblExecuteText **=** **new** javax**.**swing**.**JLabel**();**  progressBar **=** **new** javax**.**swing**.**JProgressBar**();**  JLblTzakpotsTitle **=** **new** javax**.**swing**.**JLabel**();**  JLblDraws **=** **new** javax**.**swing**.**JLabel**();**  JLblDrawsTitle **=** **new** javax**.**swing**.**JLabel**();**  JLblPayout **=** **new** javax**.**swing**.**JLabel**();**  JLblPayoutTitle **=** **new** javax**.**swing**.**JLabel**();**  JLblJackpots **=** **new** javax**.**swing**.**JLabel**();**  JLblChartByMonthTitle **=** **new** javax**.**swing**.**JLabel**();**  setDefaultCloseOperation**(**javax**.**swing**.**WindowConstants**.**EXIT\_ON\_CLOSE**);**  JPnlTzokerDataPanel**.**setBackground**(new** java**.**awt**.**Color**(**190**,** 139**,** 6**));**  JPnlTzokerDataPanel**.**setLayout**(new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteLayout**());**  JPnlTzokerDataPanel**.**add**(**JPnlChartByMonth**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**20**,** 290**,** 670**,** 170**));**  JPnlSelectYear**.**setBorder**(**javax**.**swing**.**BorderFactory**.**createTitledBorder**(**""**));**  JPnlSelectYear**.**setMinimumSize**(new** java**.**awt**.**Dimension**(**70**,** 55**));**  JPnlSelectYear**.**setOpaque**(false);**  JPnlSelectYear**.**setPreferredSize**(new** java**.**awt**.**Dimension**(**370**,** 55**));**  JLblYear**.**setFont**(new** java**.**awt**.**Font**(**"Segoe UI"**,** 1**,** 11**));** // NOI18N  JLblYear**.**setForeground**(new** java**.**awt**.**Color**(**255**,** 255**,** 255**));**  JLblYear**.**setLabelFor**(**JCmbYear**);**  JLblYear**.**setText**(**"Έτος Κλήρωσης:"**);**  JLblYear**.**setHorizontalTextPosition**(**javax**.**swing**.**SwingConstants**.**CENTER**);**  JLblYear**.**setPreferredSize**(new** java**.**awt**.**Dimension**(**90**,** 14**));**  JPnlSelectYear**.**add**(**JLblYear**);**  JCmbYear**.**setModel**(new** javax**.**swing**.**DefaultComboBoxModel**<>(new** String**[]** **{** "Item 1"**,** "Item 2"**,** "Item 3"**,** "Item 4" **}));**  JCmbYear**.**setMaximumSize**(new** java**.**awt**.**Dimension**(**56**,** 22**));**  JCmbYear**.**setMinimumSize**(new** java**.**awt**.**Dimension**(**56**,** 22**));**  JCmbYear**.**setPreferredSize**(new** java**.**awt**.**Dimension**(**65**,** 25**));**  JPnlSelectYear**.**add**(**JCmbYear**);**  JLblFiller**.**setPreferredSize**(new** java**.**awt**.**Dimension**(**50**,** 14**));**  JLblFiller**.**setRequestFocusEnabled**(false);**  JPnlSelectYear**.**add**(**JLblFiller**);**  JPnlTzokerDataPanel**.**add**(**JPnlSelectYear**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**10**,** 60**,** 690**,** 60**));**  JPnlGetAggregatedData**.**setBackground**(new** java**.**awt**.**Color**(**190**,** 139**,** 6**));**  JPnlGetAggregatedData**.**setCursor**(new** java**.**awt**.**Cursor**(**java**.**awt**.**Cursor**.**HAND\_CURSOR**));**  JPnlGetAggregatedData**.**setMinimumSize**(new** java**.**awt**.**Dimension**(**200**,** 55**));**  JPnlGetAggregatedData**.**setPreferredSize**(new** java**.**awt**.**Dimension**(**200**,** 55**));**  JPnlGetAggregatedData**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**  public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  JPnlGetAggregatedDataMouseClicked**(**evt**);**  **}**  public void mouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  JPnlGetAggregatedDataMouseEntered**(**evt**);**  **}**  public void mouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  JPnlGetAggregatedDataMouseExited**(**evt**);**  **}**  **});**  JPnlGetAggregatedData**.**setLayout**(new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteLayout**());**  JLblExecuteIcon**.**setHorizontalAlignment**(**javax**.**swing**.**SwingConstants**.**CENTER**);**  JLblExecuteIcon**.**setIcon**(new** javax**.**swing**.**ImageIcon**(**getClass**().**getResource**(**"/assets/play3.png"**)));** // NOI18N  JPnlGetAggregatedData**.**add**(**JLblExecuteIcon**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**10**,** 0**,** 40**,** 50**));**  JLblExecuteText**.**setFont**(new** java**.**awt**.**Font**(**"Segoe UI"**,** 1**,** 12**));** // NOI18N  JLblExecuteText**.**setForeground**(new** java**.**awt**.**Color**(**255**,** 255**,** 255**));**  JLblExecuteText**.**setText**(**"Εμφάνιση Αποτελεσμάτων"**);**  JPnlGetAggregatedData**.**add**(**JLblExecuteText**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**50**,** 0**,** 170**,** 50**));**  JPnlTzokerDataPanel**.**add**(**JPnlGetAggregatedData**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**0**,** 0**,** 230**,** **-**1**));**  progressBar**.**setPreferredSize**(new** java**.**awt**.**Dimension**(**695**,** 15**));**  JPnlTzokerDataPanel**.**add**(**progressBar**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**5**,** 470**,** 700**,** 20**));**  JLblTzakpotsTitle**.**setFont**(new** java**.**awt**.**Font**(**"Segoe UI"**,** 1**,** 12**));** // NOI18N  JLblTzakpotsTitle**.**setForeground**(new** java**.**awt**.**Color**(**255**,** 255**,** 255**));**  JLblTzakpotsTitle**.**setText**(**"Αριθμός ΤΖΑΚ-ΠΟΤ"**);**  JLblTzakpotsTitle**.**setHorizontalTextPosition**(**javax**.**swing**.**SwingConstants**.**CENTER**);**  JPnlTzokerDataPanel**.**add**(**JLblTzakpotsTitle**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**570**,** 120**,** 120**,** 30**));**  JLblDraws**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 0**,** 48**));** // NOI18N  JLblDraws**.**setForeground**(new** java**.**awt**.**Color**(**51**,** 51**,** 51**));**  JLblDraws**.**setHorizontalAlignment**(**javax**.**swing**.**SwingConstants**.**CENTER**);**  JLblDraws**.**setText**(**"-"**);**  JLblDraws**.**setOpaque**(true);**  JPnlTzokerDataPanel**.**add**(**JLblDraws**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**20**,** 150**,** 120**,** 110**));**  JLblDrawsTitle**.**setFont**(new** java**.**awt**.**Font**(**"Segoe UI"**,** 1**,** 12**));** // NOI18N  JLblDrawsTitle**.**setForeground**(new** java**.**awt**.**Color**(**255**,** 255**,** 255**));**  JLblDrawsTitle**.**setText**(**"Αριθμός Κληρώσεων"**);**  JLblDrawsTitle**.**setHorizontalTextPosition**(**javax**.**swing**.**SwingConstants**.**CENTER**);**  JPnlTzokerDataPanel**.**add**(**JLblDrawsTitle**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**20**,** 120**,** 120**,** 30**));**  JLblPayout**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 0**,** 48**));** // NOI18N  JLblPayout**.**setForeground**(new** java**.**awt**.**Color**(**51**,** 51**,** 51**));**  JLblPayout**.**setHorizontalAlignment**(**javax**.**swing**.**SwingConstants**.**CENTER**);**  JLblPayout**.**setText**(**"-"**);**  JLblPayout**.**setOpaque**(true);**  JPnlTzokerDataPanel**.**add**(**JLblPayout**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**170**,** 150**,** 380**,** 110**));**  JLblPayoutTitle**.**setFont**(new** java**.**awt**.**Font**(**"Segoe UI"**,** 1**,** 12**));** // NOI18N  JLblPayoutTitle**.**setForeground**(new** java**.**awt**.**Color**(**255**,** 255**,** 255**));**  JLblPayoutTitle**.**setText**(**"Διανεμόμενα Κέρδη"**);**  JLblPayoutTitle**.**setHorizontalTextPosition**(**javax**.**swing**.**SwingConstants**.**CENTER**);**  JPnlTzokerDataPanel**.**add**(**JLblPayoutTitle**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**170**,** 120**,** 380**,** 30**));**  JLblJackpots**.**setFont**(new** java**.**awt**.**Font**(**"Tahoma"**,** 0**,** 48**));** // NOI18N  JLblJackpots**.**setForeground**(new** java**.**awt**.**Color**(**51**,** 51**,** 51**));**  JLblJackpots**.**setHorizontalAlignment**(**javax**.**swing**.**SwingConstants**.**CENTER**);**  JLblJackpots**.**setText**(**"-"**);**  JLblJackpots**.**setOpaque**(true);**  JPnlTzokerDataPanel**.**add**(**JLblJackpots**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**570**,** 150**,** 120**,** 110**));**  JLblChartByMonthTitle**.**setFont**(new** java**.**awt**.**Font**(**"Segoe UI"**,** 1**,** 12**));** // NOI18N  JLblChartByMonthTitle**.**setForeground**(new** java**.**awt**.**Color**(**255**,** 255**,** 255**));**  JLblChartByMonthTitle**.**setText**(**"Κέρδη ανα Μήνα (Κλίκ για λεπτομέρειες)"**);**  JLblChartByMonthTitle**.**setFocusCycleRoot**(true);**  JPnlTzokerDataPanel**.**add**(**JLblChartByMonthTitle**,** **new** org**.**netbeans**.**lib**.**awtextra**.**AbsoluteConstraints**(**170**,** 262**,** 380**,** 30**));**  javax**.**swing**.**GroupLayout layout **=** **new** javax**.**swing**.**GroupLayout**(**getContentPane**());**  getContentPane**().**setLayout**(**layout**);**  layout**.**setHorizontalGroup**(**  layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**  **.**addGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**,** layout**.**createSequentialGroup**()**  **.**addComponent**(**JPnlTzokerDataPanel**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 715**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**  **.**addGap**(**0**,** 0**,** Short**.**MAX\_VALUE**))**  **);**  layout**.**setVerticalGroup**(**  layout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**  **.**addComponent**(**JPnlTzokerDataPanel**,** javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** 493**,** Short**.**MAX\_VALUE**)**  **);**  pack**();**  **}**// </editor-fold>  private void JPnlGetAggregatedDataMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **if** **(!**JPnlGetAggregatedData**.**isEnabled**())** **{**  **return;**  **}**  menuOverlay**.**setEnabled**(**JPnlGetAggregatedData**,** **false);**  startWaiting**((**JPanel**)** evt**.**getSource**());**  SwingWorker**<**Void**,** Void**>** mySwingWorker **=** **new** SwingWorker**<**Void**,** Void**>()** **{**  @Override  protected Void doInBackground**()** **throws** Exception **{**  DateFormat dateFormat **=** **new** SimpleDateFormat**(**"yyyy-MM-dd"**);**  int selectedYear **=** Integer**.**parseInt**(**String**.**valueOf**(**JCmbYear**.**getSelectedItem**()));**  int selectedMonth **=** 0**;**  //Original Start Date  String initFromDate **=** "1900-01-01"**;**  //Start and End dates used in loop  String fromDate **=** "1900-01-01"**;**  String toDate **=** "1900-01-01"**;**  //Initialize calendar objects with the user selections  Calendar cFrom **=** Calendar**.**getInstance**();**  Calendar cCurrentTo **=** Calendar**.**getInstance**();**  Calendar cTo **=** Calendar**.**getInstance**();**  cFrom**.**set**(**selectedYear**,** selectedMonth**,** 1**);**  cTo**.**set**(**selectedYear**,** 11**,** 31**);**  cCurrentTo**.**set**(**selectedYear**,** selectedMonth**,** 15**);**  //Keep the original starting date  initFromDate **=** dateFormat**.**format**(**cFrom**.**getTime**());**  **try** **{**  //Split to two calls in order to fetch all data!!  //Bug with the API (omits results while reurns stats code 200!!! when the date range is close to 30 days)  int LastApiCall **=** 0**;**  **do** **{**  **if** **(**cCurrentTo**.**getTime**().**after**(**cTo**.**getTime**()))** **{**  cCurrentTo**.**setTime**(**cTo**.**getTime**());**  LastApiCall **=** 1**;**  **}**    fromDate **=** dateFormat**.**format**(**cFrom**.**getTime**());**  toDate **=** dateFormat**.**format**(**cCurrentTo**.**getTime**());**  IApiIntegrator multiDrawIntegrator **=** **new** MultiDrawIntegrator**();**  multiDrawIntegrator**.**addAPIargument**(**"gameID"**,** "5104"**);**  multiDrawIntegrator**.**addAPIargument**(**"fromDate"**,** fromDate**);**  multiDrawIntegrator**.**addAPIargument**(**"toDate"**,** toDate**);**  //Fetch data  multiDrawIntegrator**.**loadDataFromAPI**();**  List**<**GameDrawResults**>** gameDrawResults\_Current **=** **(**List**<**GameDrawResults**>)** multiDrawIntegrator**.**getDataObject**(**GameDrawResults**.**class**);**  List**<**WinningColumn**>** winningColumns\_Current **=** **(**List**<**WinningColumn**>)** multiDrawIntegrator**.**getDataObject**(**WinningColumn**.**class**);**  **if** **(**Objects**.**isNull**(**gameDrawResults**))** **{**  gameDrawResults **=** **(**List**<**GameDrawResults**>)** multiDrawIntegrator**.**getDataObject**(**GameDrawResults**.**class**);**  **}** **else** **{**  //Append at the top to maintain table sorting on draw\_id, Win Category  gameDrawResults**.**addAll**(**0**,** gameDrawResults\_Current**);**  **}**  **if** **(**Objects**.**isNull**(**winningColumns**))** **{**  winningColumns **=** **(**List**<**WinningColumn**>)** multiDrawIntegrator**.**getDataObject**(**WinningColumn**.**class**);**  **}** **else** **{**  //Append at the top to maintain sorting on draw\_id  winningColumns**.**addAll**(**0**,** winningColumns\_Current**);**  **}**  cFrom**.**add**(**Calendar**.**DATE**,** 15**);**  cCurrentTo**.**setTime**(**cFrom**.**getTime**());**  cCurrentTo**.**add**(**Calendar**.**DATE**,** 14**);**  **}while** **(** LastApiCall **==** 0**);**  //Write to DB ONLY missing draw data  DBIntegrator**.**setDrawResults**(**winningColumns**,** gameDrawResults**,** DBIntegrator**.**Operation**.**FORCE\_INSERT**);**  //fetch aggregated data  DBIntegrator**.**AggregatedDrawData aggregatedDrawData **=** DBIntegrator**.**getDataForMonth**(**5104**,** initFromDate**,** toDate**);**    JLblDraws**.**setText**(**String**.**valueOf**(**aggregatedDrawData**.**getNumberOfDraws**()));**  JLblPayout**.**setText**(**String**.**format**(**"%,.2f"**,** aggregatedDrawData**.**getPayoutAmount**()));**  JLblJackpots**.**setText**(**String**.**valueOf**(**aggregatedDrawData**.**getNumberOfJackpots**()));**  //Show bar Chart for draws  JPnlChartByMonth**.**removeAll**();**  MonthsBarChart monthsBarChart **=** **new** MonthsBarChart**(**5104**,** initFromDate**,** toDate**,** JPnlTzokerDataPanel**);**  monthsBarChart**.**setPreferredSize**(new** Dimension**(**700**,** 170**));**  JPnlChartByMonth**.**add**(**monthsBarChart**);**  JPnlChartByMonth**.**repaint**();**  JPnlChartByMonth**.**revalidate**();**  **}** **catch** **(**Exception e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα κατα την παρουσίαση των δεδομένων." **+** e**.**getMessage**(),**  "Παρουσίαση Δεδομένων"**,** JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  **}** **finally** **{**  //Restore initial UI status  stopWaiting**();**  menuOverlay**.**setEnabled**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  **return** **null;**  **}**  **};**  mySwingWorker**.**execute**();**  **}**  private void JPnlGetAggregatedDataMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlGetAggregatedDataMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  // Variables declaration - do not modify  private javax**.**swing**.**JComboBox**<**String**>** JCmbYear**;**  private javax**.**swing**.**JLabel JLblChartByMonthTitle**;**  private javax**.**swing**.**JLabel JLblDraws**;**  private javax**.**swing**.**JLabel JLblDrawsTitle**;**  private javax**.**swing**.**JLabel JLblExecuteIcon**;**  private javax**.**swing**.**JLabel JLblExecuteText**;**  private javax**.**swing**.**JLabel JLblFiller**;**  private javax**.**swing**.**JLabel JLblJackpots**;**  private javax**.**swing**.**JLabel JLblPayout**;**  private javax**.**swing**.**JLabel JLblPayoutTitle**;**  private javax**.**swing**.**JLabel JLblTzakpotsTitle**;**  private javax**.**swing**.**JLabel JLblYear**;**  private javax**.**swing**.**JPanel JPnlChartByMonth**;**  private javax**.**swing**.**JPanel JPnlGetAggregatedData**;**  private javax**.**swing**.**JPanel JPnlSelectYear**;**  private javax**.**swing**.**JPanel JPnlTzokerDataPanel**;**  private javax**.**swing**.**JProgressBar progressBar**;**  // End of variables declaration  //Populate Year and month Comboboxes  private void setComboValues**()** **{**  **this.**JCmbYear**.**removeAllItems**();**  **for** **(**int i **=** Calendar**.**getInstance**().**get**(**Calendar**.**YEAR**);** i **>=** 2000**;** i**--)** **{**  **this.**JCmbYear**.**addItem**(**String**.**valueOf**(**i**));**  **}**  **}**  private void startWaiting**(**JPanel panel**)** **{**  progressBar**.**setVisible**(true);**  progressBar**.**setIndeterminate**(true);**  progressBar**.**setValue**(**100**);**  JPnlTzokerDataPanel**.**setCursor**(**Cursor**.**getPredefinedCursor**(**Cursor**.**WAIT\_CURSOR**));**  panel**.**grabFocus**();**  **}**  private void stopWaiting**()** **{**  JPnlTzokerDataPanel**.**setCursor**(null);** //turn off the wait cursor  progressBar**.**setIndeterminate**(false);**  progressBar**.**setVisible**(false);**  **}**    **}** |

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:** **DisplayDataByDrawFrame.Java** |
| package gui**;**  **import** DataIngestion**.**DBIntegrator**;**  **import** DataIngestion**.**IApiIntegrator**;**  **import** DataIngestion**.**MultiDrawIntegrator**;**  **import** java**.**awt**.**Cursor**;**  **import** java**.**awt**.**Dimension**;**  **import** java**.**text**.**DateFormat**;**  **import** java**.**text**.**SimpleDateFormat**;**  **import** java**.**util**.**Calendar**;**  **import** java**.**util**.**List**;**  **import** java**.**util**.**Objects**;**  **import** javax**.**swing**.**JLabel**;**  **import** javax**.**swing**.**JOptionPane**;**  **import** javax**.**swing**.**JPanel**;**  **import** javax**.**swing**.**SwingWorker**;**  **import** model**.**GameDrawResults**;**  **import** model**.**WinningColumn**;**  public class DiplayDataByDrawFrame **extends** javax**.**swing**.**JFrame **{**  private MenuOverlay menuOverlay **=** **new** MenuOverlay**();**  private List**<**GameDrawResults**>** gameDrawResults**;**  //List of API Responce for winning numbers  private List**<**WinningColumn**>** winningColumns**;**  /\*\*  \* Creates new form NewJFrame  \*/  public DiplayDataByDrawFrame**()** **{**  initComponents**();**  setComboValues**();**  progressBar**.**setVisible**(false);**  JLblPayoutTitle**.**setHorizontalAlignment**(**JLabel**.**CENTER**);**  JLbChartTitle**.**setHorizontalAlignment**(**JLabel**.**CENTER**);**  **}**  private void JPnlGetDrawDataMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **if** **(!**JPnlGetDrawData**.**isEnabled**())** **{**  **return;**  **}**  execForm**();**  **}**  private void JPnlGetDrawDataMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlGetDrawDataMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  private void JPnlGoBackMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **this.**dispose**();**  **}**  private void JPnlGoBackMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlGoBackMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  // Variables declaration - do not modify  private javax**.**swing**.**JComboBox**<**String**>** JCmbMonth**;**  private javax**.**swing**.**JComboBox**<**String**>** JCmbYear**;**  private javax**.**swing**.**JLabel JLbChartTitle**;**  private javax**.**swing**.**JLabel JLbGoBackIcon**;**  private javax**.**swing**.**JLabel JLbGoBackText**;**  private javax**.**swing**.**JLabel JLblDraws**;**  private javax**.**swing**.**JLabel JLblDrawsTitle**;**  private javax**.**swing**.**JLabel JLblExecuteIcon**;**  private javax**.**swing**.**JLabel JLblExecuteText**;**  private javax**.**swing**.**JLabel JLblFiller**;**  private javax**.**swing**.**JLabel JLblJackpots**;**  private javax**.**swing**.**JLabel JLblJackpotsTitle**;**  private javax**.**swing**.**JLabel JLblMonth**;**  private javax**.**swing**.**JLabel JLblPayout**;**  private javax**.**swing**.**JLabel JLblPayoutTitle**;**  private javax**.**swing**.**JLabel JLblYear**;**  private javax**.**swing**.**JPanel JPnlChartByDraw**;**  private javax**.**swing**.**JPanel JPnlGetDrawData**;**  private javax**.**swing**.**JPanel JPnlGoBack**;**  private javax**.**swing**.**JPanel JPnlShowData**;**  private javax**.**swing**.**JProgressBar progressBar**;**  private javax**.**swing**.**JPanel tzokerDataPanel**;**  // End of variables declaration  //Populate Year and month Comboboxes  private void setComboValues**()** **{**  **this.**JCmbYear**.**removeAllItems**();**  **this.**JCmbMonth**.**removeAllItems**();**  **for** **(**int i **=** Calendar**.**getInstance**().**get**(**Calendar**.**YEAR**);** i **>=** 2000**;** i**--)** **{**  **this.**JCmbYear**.**addItem**(**String**.**valueOf**(**i**));**  **}**  **this.**JCmbMonth**.**addItem**(**"Ιανουάριος"**);**  **this.**JCmbMonth**.**addItem**(**"Φεβρουάριος"**);**  **this.**JCmbMonth**.**addItem**(**"Μάρτιος"**);**  **this.**JCmbMonth**.**addItem**(**"Απρίλιος"**);**  **this.**JCmbMonth**.**addItem**(**"Μάιος"**);**  **this.**JCmbMonth**.**addItem**(**"Ιούνιος"**);**  **this.**JCmbMonth**.**addItem**(**"Ιούλιος"**);**  **this.**JCmbMonth**.**addItem**(**"Αύγουστος"**);**  **this.**JCmbMonth**.**addItem**(**"Σεπτέμβριος"**);**  **this.**JCmbMonth**.**addItem**(**"Οκτώβριος"**);**  **this.**JCmbMonth**.**addItem**(**"Νοέμβριος"**);**  **this.**JCmbMonth**.**addItem**(**"Δεκέμβριος"**);**  **}**  private void startWaiting**(**JPanel panel**)** **{**  progressBar**.**setVisible**(true);**  progressBar**.**setIndeterminate**(true);**  progressBar**.**setValue**(**100**);**  tzokerDataPanel**.**setCursor**(**Cursor**.**getPredefinedCursor**(**Cursor**.**WAIT\_CURSOR**));**  panel**.**grabFocus**();**  **}**  private void stopWaiting**()** **{**  tzokerDataPanel**.**setCursor**(null);** //turn off the wait cursor  progressBar**.**setIndeterminate**(false);**  progressBar**.**setVisible**(false);**  **}**      public void setYear**(**String Year**)** **{**  **this.**JCmbYear**.**setSelectedItem**(**Year**);**  **}**  public void setMonth**(**int Month**)** **{**  **this.**JCmbMonth**.**setSelectedIndex**(**Month**-**1**);**  **}**      public void execForm**(){**    startWaiting**((**JPanel**)** JPnlGetDrawData**);**  menuOverlay**.**setEnabled**(**JPnlGetDrawData**,** **false);**  SwingWorker**<**Void**,** Void**>** mySwingWorker **=** **new** SwingWorker**<**Void**,** Void**>()** **{**  @Override  protected Void doInBackground**()** **throws** Exception **{**  DateFormat dateFormat **=** **new** SimpleDateFormat**(**"yyyy-MM-dd"**);**  int selectedYear **=** Integer**.**parseInt**(**String**.**valueOf**(**JCmbYear**.**getSelectedItem**()));**  int selectedMonth **=** JCmbMonth**.**getSelectedIndex**();**  //Original Start Date  String initFromDate **=** "1900-01-01"**;**  //Start and End dates used in loop  String fromDate **=** "1900-01-01"**;**  String toDate **=** "1900-01-01"**;**  //Initialize calendar objects with the user selections  Calendar cFrom **=** Calendar**.**getInstance**();**  Calendar cTo **=** Calendar**.**getInstance**();**  cFrom**.**set**(**selectedYear**,** selectedMonth**,** 1**);**  int lastDayOfMonth **=** cFrom**.**getActualMaximum**(**Calendar**.**DATE**);**  cFrom**.**set**(**selectedYear**,** selectedMonth**,** 1**);**  cTo**.**set**(**selectedYear**,** selectedMonth**,** 15**);**  //Keep the original starting date  initFromDate **=** dateFormat**.**format**(**cFrom**.**getTime**());**  **try** **{**  //Split to two calls in order to fetch all data!!  //Bug with the API (omits results while reurns stats code 200!!! when the date range is close to 30 days)  **for** **(**int i **=** 0**;** i **<** 2**;** i**++)** **{**  fromDate **=** dateFormat**.**format**(**cFrom**.**getTime**());**  toDate **=** dateFormat**.**format**(**cTo**.**getTime**());**  System**.**out**.**println**(**"From Date " **+** fromDate**);**  System**.**out**.**println**(**"To Date" **+** toDate**);**  IApiIntegrator multiDrawIntegrator **=** **new** MultiDrawIntegrator**();**  multiDrawIntegrator**.**addAPIargument**(**"gameID"**,** "5104"**);**  multiDrawIntegrator**.**addAPIargument**(**"fromDate"**,** fromDate**);**  multiDrawIntegrator**.**addAPIargument**(**"toDate"**,** toDate**);**  //Fetch data  multiDrawIntegrator**.**loadDataFromAPI**();**  List**<**GameDrawResults**>** gameDrawResults\_Current **=** **(**List**<**GameDrawResults**>)** multiDrawIntegrator**.**getDataObject**(**GameDrawResults**.**class**);**  List**<**WinningColumn**>** winningColumns\_Current **=** **(**List**<**WinningColumn**>)** multiDrawIntegrator**.**getDataObject**(**WinningColumn**.**class**);**  **if** **(**Objects**.**isNull**(**gameDrawResults**))** **{**  gameDrawResults **=** **(**List**<**GameDrawResults**>)** multiDrawIntegrator**.**getDataObject**(**GameDrawResults**.**class**);**  **}** **else** **{**  //Append at the top to maintain table sorting on draw\_id, Win Category  gameDrawResults**.**addAll**(**0**,** gameDrawResults\_Current**);**  **}**  **if** **(**Objects**.**isNull**(**winningColumns**))** **{**  winningColumns **=** **(**List**<**WinningColumn**>)** multiDrawIntegrator**.**getDataObject**(**WinningColumn**.**class**);**  **}** **else** **{**  //Append at the top to maintain sorting on draw\_id  winningColumns**.**addAll**(**0**,** winningColumns\_Current**);**  **}**  cFrom**.**set**(**selectedYear**,** selectedMonth**,** 16**);**  cTo**.**set**(**selectedYear**,** selectedMonth**,** lastDayOfMonth**);**  **}**  //Write to DB ONLY missing draw data  DBIntegrator**.**setDrawResults**(**winningColumns**,** gameDrawResults**,** DBIntegrator**.**Operation**.**FORCE\_INSERT**);**  //fetch aggregated data  DBIntegrator**.**AggregatedDrawData aggregatedDrawData **=** DBIntegrator**.**getDataForMonth**(**5104**,** initFromDate**,** toDate**);**  JLblDraws**.**setText**(**String**.**valueOf**(**aggregatedDrawData**.**getNumberOfDraws**()));**  JLblPayout**.**setText**(**String**.**format**(**"%,.2f"**,** aggregatedDrawData**.**getPayoutAmount**()));**  JLblJackpots**.**setText**(**String**.**valueOf**(**aggregatedDrawData**.**getNumberOfJackpots**()));**  //Show bar Chart for draws  JPnlChartByDraw**.**removeAll**();**  DrawsBarChart drawsBarChart **=** **new** DrawsBarChart**(**5104**,** initFromDate**,** toDate**,** String**.**valueOf**(**JCmbMonth**.**getSelectedItem**()));**  drawsBarChart**.**setPreferredSize**(new** Dimension**(**700**,** 170**));**  JPnlChartByDraw**.**add**(**drawsBarChart**);**  JPnlChartByDraw**.**repaint**();**  JPnlChartByDraw**.**revalidate**();**  **}** **catch** **(**Exception e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα κατα την παρουσίαση των δεδομένων." **+** e**.**getMessage**(),**  "Παρουσίαση Δεδομένων"**,** JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  **}** **finally** **{**  //Restore initial UI status  stopWaiting**();**  menuOverlay**.**setEnabled**((**JPanel**)** JPnlGetDrawData**,** **true);**  **}**  **return** **null;**  **}**  **};**  mySwingWorker**.**execute**();**    **}**  **}** |

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:** **StatsMainFrame.Java** |
| package gui**;**  **import** DataIngestion**.**WinningColumnStats**;**  **import** DataIngestion**.**WinningColumnStatsIntegrator**;**  **import** java**.**awt**.**Cursor**;**  **import** java**.**io**.**IOException**;**  **import** javax**.**swing**.**JOptionPane**;**  **import** javax**.**swing**.**JPanel**;**  public class StatsMainFrame **extends** javax**.**swing**.**JFrame **{**  //Object to manipulate the menu behaviour  private MenuOverlay menuOverlay **=** **new** MenuOverlay**();**  /\*\*  \* Creates new form StatsDataFrame  \*/  public StatsMainFrame**()** **{**  initComponents**();**  **}**  // Displays Stats Table Frame, executes api call  private void JPnlShowTableMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **if** **(!**JPnlShowTable**.**isEnabled**())** **{**  **return;**  **}**  menuOverlay**.**setEnabled**(**JPnlShowTable**,** **false);**  startWaiting**((**JPanel**)** evt**.**getSource**());**  StatsTableFrame tableDataFrame **=** **new** StatsTableFrame**();**  JPnlDisplay**.**removeAll**();**  JPnlDisplay**.**repaint**();**  JPnlDisplay**.**revalidate**();**  **try** **{**  WinningColumnStatsIntegrator winningColumnStatsIntegrator  **=** **new** WinningColumnStatsIntegrator**();**  // Setting api call parameters  winningColumnStatsIntegrator**.**addAPIargument**(**"gameID"**,** "5104"**);**  WinningColumnStats winningColumnStats **=** **new** WinningColumnStats**();**  // Execute api call to retrieve winning column statistics  winningColumnStatsIntegrator**.**loadDataFromAPI**();**  winningColumnStats **=** winningColumnStatsIntegrator**.**getTotalNumberStats**();**  // Populate JTables with statistics data from response  tableDataFrame**.**fillTables**(**winningColumnStats**);**  tableDataFrame**.**totaNumberStats **=** winningColumnStats**;**  JPnlDisplay**.**add**(**tableDataFrame**.**getContentPane**());**  JPnlDisplay**.**repaint**();**  JPnlDisplay**.**revalidate**();**  **}** **catch** **(**IOException e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα κατα την"  **+** " ανάγνωση των δεδομένων απο τo: "  **+** e**.**getMessage**()** **+** "\n Ελέγξτε την σύνδεση σας."**,**  "Δεν Διαβάστηκαν Δεδομένα"**,**  JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  menuOverlay**.**setEnabled**(**JPnlShowTable**,** **true);**  **}** **finally** **{**  //Restore initial UI status  stopWaiting**();**  menuOverlay**.**setEnabled**(**JPnlShowStats**,** **true);**  **}**  **}**  private void JPnlShowTableMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlShowTableMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  private void JPnlShowStatsMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlShowStatsMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  // Displays Stats harts Frame  private void JPnlShowStatsMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **if** **(!**JPnlShowStats**.**isEnabled**())** **{**  **return;**  **}**  menuOverlay**.**setEnabled**(**JPnlShowStats**,** **false);**  startWaiting**((**JPanel**)** evt**.**getSource**());**  StatsChartsFrame chartsDataFrame **=** **new** StatsChartsFrame**();**  JPnlDisplay**.**removeAll**();**  JPnlDisplay**.**repaint**();**  JPnlDisplay**.**revalidate**();**  JPnlDisplay**.**add**(**chartsDataFrame**.**getContentPane**());**  JPnlDisplay**.**repaint**();**  JPnlDisplay**.**revalidate**();**  //Restore initial UI status  stopWaiting**();**  menuOverlay**.**setEnabled**(**JPnlShowTable**,** **true);**  **}**  private void startWaiting**(**JPanel panel**)** **{**  JPnlMain**.**setCursor**(**Cursor**.**getPredefinedCursor**(**Cursor**.**WAIT\_CURSOR**));**  panel**.**grabFocus**();**  **}**  private void stopWaiting**()** **{**  JPnlMain**.**setCursor**(null);** //turn off the wait cursor  **}**  // Variables declaration - do not modify  private javax**.**swing**.**JLabel JLblShowStatsIcon**;**  private javax**.**swing**.**JLabel JLblShowStatsText**;**  private javax**.**swing**.**JLabel JLblShowTableIcon**;**  private javax**.**swing**.**JLabel JLblShowTableText**;**  private javax**.**swing**.**JPanel JPnlDisplay**;**  private javax**.**swing**.**JPanel JPnlMain**;**  private javax**.**swing**.**JPanel JPnlShowStats**;**  private javax**.**swing**.**JPanel JPnlShowTable**;**  // End of variables declaration  **}** |

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:** **StatsTableFrame.Java** |
| package gui**;**  **import** javax**.**swing**.**JPanel**;**  **import** DataIngestion**.**WinningColumnStats**;**  **import** javax**.**swing**.**table**.**DefaultTableModel**;**  **import** java**.**io**.**FileOutputStream**;**  **import** com**.**itextpdf**.**text**.**BaseColor**;**  **import** com**.**itextpdf**.**text**.**Document**;**  **import** com**.**itextpdf**.**text**.**DocumentException**;**  **import** com**.**itextpdf**.**text**.**Element**;**  **import** com**.**itextpdf**.**text**.**Font**;**  **import** com**.**itextpdf**.**text**.**Paragraph**;**  **import** com**.**itextpdf**.**text**.**Phrase**;**  **import** com**.**itextpdf**.**text**.**Rectangle**;**  **import** com**.**itextpdf**.**text**.**pdf**.**BaseFont**;**  **import** com**.**itextpdf**.**text**.**pdf**.**ColumnText**;**  **import** com**.**itextpdf**.**text**.**pdf**.**PdfContentByte**;**  **import** com**.**itextpdf**.**text**.**pdf**.**PdfPCell**;**  **import** com**.**itextpdf**.**text**.**pdf**.**PdfPTable**;**  **import** com**.**itextpdf**.**text**.**pdf**.**PdfWriter**;**  **import** java**.**awt**.**Component**;**  **import** java**.**awt**.**Desktop**;**  **import** java**.**awt**.**FlowLayout**;**  **import** java**.**io**.**File**;**  **import** java**.**io**.**FileNotFoundException**;**  **import** java**.**io**.**IOException**;**  **import** java**.**util**.**logging**.**Level**;**  **import** java**.**util**.**logging**.**Logger**;**  **import** javax**.**swing**.**JCheckBox**;**  **import** javax**.**swing**.**JFileChooser**;**  **import** javax**.**swing**.**JLabel**;**  **import** javax**.**swing**.**JOptionPane**;**  **import** javax**.**swing**.**filechooser**.**FileFilter**;**  **import** javax**.**swing**.**table**.**DefaultTableCellRenderer**;**  public class StatsTableFrame **extends** javax**.**swing**.**JFrame **{**  //Object to manipulate the menu behaviour  private MenuOverlay menuOverlay **=** **new** MenuOverlay**();**  public WinningColumnStats totaNumberStats**;**  /\*\*  \* Creates new form TableDataFrame  \*/  public StatsTableFrame**()** **{**  initComponents**();**  //Init Dialog box to save pdf  dirPicker **=** **new** JFileChooser**()** **{**  @Override  public void approveSelection**()** **{**  File f **=** getSelectedFile**();**  **if** **(**f**.**exists**()** **&&** getDialogType**()** **==** SAVE\_DIALOG**)** **{**  int result **=** JOptionPane**.**showConfirmDialog**(this,** "Το αρχείο υπάρχει ήδη. Θέλετε να γίνει αντικατάσταση;"**,** "Υπάρχον αρχείο"**,** JOptionPane**.**YES\_NO\_CANCEL\_OPTION**);**  **switch** **(**result**)** **{**  **case** JOptionPane**.**YES\_OPTION**:**  **super.**approveSelection**();**  **return;**  **case** JOptionPane**.**NO\_OPTION**:**  **return;**  **case** JOptionPane**.**CLOSED\_OPTION**:**  **return;**  **case** JOptionPane**.**CANCEL\_OPTION**:**  cancelSelection**();**  **return;**  **}**  **}**  **super.**approveSelection**();**  **}**  **};**  dirPicker**.**setDialogTitle**(**"Επιλέξτε φάκελο και όνομα PDF αρχείου."**);**  //Dialog box Applicable file types  dirPicker**.**setAcceptAllFileFilterUsed**(false);**  dirPicker**.**setFileFilter**(new** FileFilter**()** **{**  public String getDescription**()** **{**  **return** "PDF Documents (\*.pdf)"**;**  **}**  public boolean accept**(**File f**)** **{**  **if** **(**f**.**isDirectory**())** **{**  **return** **true;**  **}** **else** **{**  String filename **=** f**.**getName**().**toLowerCase**();**  **return** filename**.**endsWith**(**".pdf"**);**  **}**  **}**  **});**  //Prepare "Open after Save" option  JCheckBox ChkOpen **=** **new** JCheckBox**(**"Άνοιγμα του αρχείου"**);**  JPanel accessory **=** **new** JPanel**();**  accessory**.**setLayout**(new** FlowLayout**());**  accessory**.**add**(**ChkOpen**);**  dirPicker**.**setAccessory**(**accessory**);**  **}**    private void JPnlPrintPdfMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  private void JPnlPrintPdfMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  /\* Generates .pdf file in the directory specified  Pdf consists of statistics data in table form \*/  private void generatePdf**(**String fileName**)** **throws** IOException **{**  Document pdf **=** **new** Document**();**  **try** **{**  PdfWriter writer **=** PdfWriter**.**getInstance**(**pdf**,**  **new** FileOutputStream**(**fileName **+** ".pdf"**));**  pdf**.**open**();**  Font normFont **=** **new** Font**(**Font**.**FontFamily**.**HELVETICA**,** 11**,** Font**.**NORMAL**);**  String currentDirectory **=** System**.**getProperty**(**"user.dir"**);**  BaseFont baseFont **=** BaseFont**.**createFont**(**currentDirectory **+**  "\\src\\assets\\Ubuntu-Regular.ttf"**,**  BaseFont**.**IDENTITY\_H**,** BaseFont**.**NOT\_EMBEDDED**);**  Font greek **=** **new** Font**(**baseFont**,** 11**);**  Font greek26 **=** **new** Font**(**baseFont**,** 26**);**  Font greek16 **=** **new** Font**(**baseFont**,** 16**);**  greek**.**setColor**(**BaseColor**.**WHITE**);**  greek16**.**setColor**(**BaseColor**.**WHITE**);**  greek26**.**setColor**(new** BaseColor**(**255**,** 195**,** 20**));**  PdfContentByte canvas **=** writer**.**getDirectContent**();**  Rectangle line **=** **new** Rectangle**(**0**,** 766**,** 600**,** 768**);**  line**.**setBackgroundColor**(new** BaseColor**(**0**,** 53**,** 128**));**  canvas**.**rectangle**(**line**);**  Rectangle rect **=** **new** Rectangle**(**0**,** 770**,** 600**,** 850**);**  rect**.**setBackgroundColor**(new** BaseColor**(**0**,** 53**,** 128**));**  canvas**.**rectangle**(**rect**);**  ColumnText ct **=** **new** ColumnText**(**writer**.**getDirectContent**());**  ct**.**setSimpleColumn**(**rect**);**  Paragraph title **=** **new** Paragraph**();**  title**.**add**(new** Paragraph**(**" "  **+** "ΤΖΟΚΕΡ"**,** greek26**));**  title**.**add**(new** Paragraph**(**" "  **+** "Στατιστικά παιχνιδιού: Εμφανίσεις-Καθυστερήσεις"**,** greek16**));**  ct**.**addElement**(**title**);**  ct**.**go**();**  greek**.**setColor**(**BaseColor**.**BLACK**);**  ColumnText ctDesc **=** **new** ColumnText**(**writer**.**getDirectContent**());**  ctDesc**.**setSimpleColumn**(**335**,** 250**,** 580**,** 345**);**  ctDesc**.**addText**(new** Phrase**(**"\*Οι πίνακες απεικονίζουν πόσο συχνά έχουν "  **+** "εμφανιστεί ή καθυστερήσει να εμφανιστούν οι "  **+** "αριθμοί της νικιτριας στηλης του Τζόκερ από την αρχή του "  **+** "παιχνιδιού μέχρι και σήμερα."**,** greek**));**  ctDesc**.**go**();**  greek**.**setColor**(**BaseColor**.**WHITE**);**  // Table with stats of the five numbers of the winning column  PdfPTable numtable **=** **new** PdfPTable**(**3**);**  PdfPCell c1 **=** **new** PdfPCell**();**  c1**.**setBackgroundColor**(new** BaseColor**(**0**,** 53**,** 128**));**  c1**.**setPadding**(**5**);**  c1**.**setHorizontalAlignment**(**Element**.**ALIGN\_CENTER**);**  // table headers  c1**.**setPhrase**(new** Phrase**(**"Αριθμοί πεντάδας"**,** greek**));**  numtable**.**addCell**(**c1**);**  c1**.**setPhrase**(new** Phrase**(**"Εμφανίσεις"**,** greek**));**  numtable**.**addCell**(**c1**);**  c1**.**setPhrase**(new** Phrase**(**"Καθυστερήση"**,** greek**));**  numtable**.**addCell**(**c1**);**    // Populate five winning numbers table  **for** **(**WinningColumnStats**.**Number number **:** totaNumberStats**.**numbers**)** **{**  PdfPCell cell1 **=** **new** PdfPCell**();**  cell1**.**setFixedHeight**(**10**);**  cell1**.**setHorizontalAlignment**(**Element**.**ALIGN\_CENTER**);**  cell1**.**setPhrase**(new** Phrase**(**String**.**valueOf**(**number**.**number**),** normFont**));**  numtable**.**addCell**(**cell1**);**  PdfPCell cell2 **=** **new** PdfPCell**();**  cell2**.**setHorizontalAlignment**(**Element**.**ALIGN\_CENTER**);**  cell2**.**setPhrase**(new** Phrase**(**String**.**valueOf**(**number**.**occurrences**),** normFont**));**  numtable**.**addCell**(**cell2**);**  PdfPCell cell3 **=** **new** PdfPCell**();**  cell3**.**setHorizontalAlignment**(**Element**.**ALIGN\_CENTER**);**  cell3**.**setPhrase**(new** Phrase**(**String**.**valueOf**(**number**.**delays**),** normFont**));**  numtable**.**addCell**(**cell3**);**  **}**    // Table with winning column stats of joker number  PdfPTable bnumtable **=** **new** PdfPTable**(**3**);**  PdfPCell c2 **=** **new** PdfPCell**();**  c2**.**setBackgroundColor**(new** BaseColor**(**0**,** 53**,** 128**));**  c2**.**setPadding**(**5**);**  c2**.**setHorizontalAlignment**(**Element**.**ALIGN\_CENTER**);**  // table headers  c1**.**setPhrase**(new** Phrase**(**"Αριθμός Τζόκερ"**,** greek**));**  bnumtable**.**addCell**(**c1**);**  c1**.**setPhrase**(new** Phrase**(**"Εμφανίσεις"**,** greek**));**  bnumtable**.**addCell**(**c1**);**  c1**.**setPhrase**(new** Phrase**(**"Καθυστέρηση"**,** greek**));**  bnumtable**.**addCell**(**c1**);**    // Populate joker numbers table  **for** **(**WinningColumnStats**.**BonusNumber bonusNumber **:**  totaNumberStats**.**bonusNumbers**)** **{**  PdfPCell cell1 **=** **new** PdfPCell**();**  cell1**.**setFixedHeight**(**10**);**  cell1**.**setHorizontalAlignment**(**Element**.**ALIGN\_CENTER**);**  cell1**.**setPhrase**(new** Phrase**(**String**.**valueOf**(**  bonusNumber**.**number**),** normFont**));**  bnumtable**.**addCell**(**cell1**);**  PdfPCell cell2 **=** **new** PdfPCell**();**  cell2**.**setHorizontalAlignment**(**Element**.**ALIGN\_CENTER**);**  cell2**.**setPhrase**(new** Phrase**(**  String**.**valueOf**(**bonusNumber**.**occurrences**),** normFont**));**  bnumtable**.**addCell**(**cell2**);**  PdfPCell cell3 **=** **new** PdfPCell**();**  cell3**.**setHorizontalAlignment**(**Element**.**ALIGN\_CENTER**);**  cell3**.**setPhrase**(new** Phrase**(**  String**.**valueOf**(**bonusNumber**.**delays**),** normFont**));**  bnumtable**.**addCell**(**cell3**);**  **}**  ColumnText ct1 **=** **new** ColumnText**(**writer**.**getDirectContent**());**  ct1**.**setSimpleColumn**(**0**,** 0**,** 300**,** 740**);**  ct1**.**addElement**(**numtable**);**  ct1**.**go**();**  ColumnText ct2 **=** **new** ColumnText**(**writer**.**getDirectContent**());**  ct2**.**setSimpleColumn**(**301**,** 0**,** 600**,** 740**);**  ct2**.**addElement**(**bnumtable**);**  ct2**.**go**();**  pdf**.**close**();**  JOptionPane**.**showMessageDialog**(null,** "Το pdf αρχείο δημιουργήθηκε "  **+** "επιτυχώς στο φάκελο:\n" **+** fileName**,**  ""**,** JOptionPane**.**INFORMATION\_MESSAGE**);**  **}** **catch** **(**FileNotFoundException **|** DocumentException ex**)** **{**  Logger**.**getLogger**(**StatsTableFrame**.**class**.**getName**()).**log**(**  Level**.**SEVERE**,** **null,** ex**);**  **}**  **}**  // Variables declaration - do not modify  private javax**.**swing**.**JLabel JLblNumbersATitle**;**  private javax**.**swing**.**JLabel JLblNumbersBTitle**;**  private javax**.**swing**.**JLabel JLblPrintPdfIcon**;**  private javax**.**swing**.**JLabel JLblPrintPdfText**;**  private javax**.**swing**.**JPanel JPnlMain**;**  private javax**.**swing**.**JPanel JPnlNumbersA**;**  private javax**.**swing**.**JPanel JPnlNumbersB**;**  private javax**.**swing**.**JPanel JPnlPrintPdf**;**  private javax**.**swing**.**JTable bonusNumbersTable**;**  private javax**.**swing**.**JLabel jLabel1**;**  private javax**.**swing**.**JScrollPane jScrollPane1**;**  private javax**.**swing**.**JScrollPane jScrollPane2**;**  private javax**.**swing**.**JTable numbersTable**;**  // End of variables declaration  private JFileChooser dirPicker**;**  **}** |

## Ερώτημα Γ – Παρουσίαση Στατιστικών Στοιχείων και Γραφημάτων

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:** **StatsChartsFrame.Java** |
| package gui**;**  **import** DataIngestion**.**DBIntegrator**;**  **import** java**.**awt**.**BasicStroke**;**  **import** java**.**awt**.**Color**;**  **import** java**.**awt**.**Cursor**;**  **import** java**.**awt**.**Rectangle**;**  **import** java**.**text**.**SimpleDateFormat**;**  **import** java**.**util**.**ArrayList**;**  **import** java**.**util**.**Date**;**  **import** javax**.**swing**.**JOptionPane**;**  **import** javax**.**swing**.**JPanel**;**  **import** javax**.**swing**.**SwingWorker**;**  **import** model**.**GameDrawResults**;**  **import** model**.**GameDrawResultsPK**;**  **import** org**.**jfree**.**chart**.**JFreeChart**;**  **import** org**.**jfree**.**chart**.**plot**.**PiePlot3D**;**  **import** org**.**jfree**.**data**.**general**.**DefaultPieDataset**;**  **import** org**.**jfree**.**chart**.**ChartPanel**;**  **import** org**.**jfree**.**chart**.**axis**.**CategoryAxis**;**  **import** org**.**jfree**.**chart**.**axis**.**CategoryLabelPositions**;**  **import** org**.**jfree**.**chart**.**axis**.**NumberAxis**;**  **import** org**.**jfree**.**chart**.**axis**.**ValueAxis**;**  **import** org**.**jfree**.**chart**.**labels**.**StandardCategoryItemLabelGenerator**;**  **import** org**.**jfree**.**chart**.**plot**.**CategoryPlot**;**  **import** org**.**jfree**.**chart**.**plot**.**PlotOrientation**;**  **import** org**.**jfree**.**chart**.**renderer**.**category**.**StackedBarRenderer**;**  **import** org**.**jfree**.**chart**.**renderer**.**category**.**StandardBarPainter**;**  **import** org**.**jfree**.**data**.**category**.**DefaultCategoryDataset**;**  **import** org**.**jfree**.**chart**.**renderer**.**category**.**BarRenderer**;**  **import** org**.**jfree**.**chart**.**ChartFactory**;**  **import** org**.**jfree**.**chart**.**StandardChartTheme**;**  **import** org**.**jfree**.**chart**.**title**.**TextTitle**;**  **import** org**.**jfree**.**chart**.**ui**.**RectangleEdge**;**  **import** org**.**jfree**.**chart**.**ui**.**RectangleInsets**;**  **import** tzokerstats**.**GUIValidationException**;**  **import** model**.**WinningColumn**;**  **import** model**.**WinningColumnPK**;**  public class StatsChartsFrame **extends** javax**.**swing**.**JFrame **{**  /\*\*  \* Creates new form ChartsDataFrame  \*/  public StatsChartsFrame**()** **{**  initComponents**();**  progressBar**.**setVisible**(false);**  jXDatePickerFrom**.**setFormats**(new** SimpleDateFormat**(**"dd/MM/yyyy"**));**  jXDatePickerTo**.**setFormats**(new** SimpleDateFormat**(**"dd/MM/yyyy"**));**  **}**  private MenuOverlay menuOverlay **=** **new** MenuOverlay**();**    private void JPnlDisplayResultMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  **if** **(!**JPnlDisplayResult**.**isEnabled**())** **{**  **return;**  **}**  menuOverlay**.**setEnabled**(**JPnlDisplayResult**,** **false);**  startWaiting**((**JPanel**)** evt**.**getSource**());**    //Start Waiting  SwingWorker**<**Void**,** Void**>** mySwingWorker **=** **new** SwingWorker**<**Void**,** Void**>()** **{**  @Override  protected Void doInBackground**()** **throws** Exception **{**    **try** **{**  // Check if dates have been selected  jXDatePickerFrom**.**commitEdit**();**  **if** **(**jXDatePickerFrom**.**getDate**()** **==** **null)** **{**  **throw** **new** GUIValidationException**(**  "Παρακαλώ εισάγετε αρχική ημερομηνία."**);**  **}**  jXDatePickerTo**.**commitEdit**();**  **if** **(**jXDatePickerTo**.**getDate**()** **==** **null)** **{**  **throw** **new** GUIValidationException**(**  "Παρακαλώ εισάγετε τελική ημερομηνία."**);**  **}**  // Get selected dates  Date selectedDateFrom **=** jXDatePickerFrom**.**getDate**();**  Date selectedDateTo **=** jXDatePickerTo**.**getDate**();**  // Retrieve records from database depending on selected date range  ArrayList**<**WinningColumn**>** dateRangeResults **=**  DBIntegrator**.**getWinningColumnByDateRange**(**  selectedDateFrom**,** selectedDateTo**);**  **if** **(**dateRangeResults**.**size**()** **<** 1**)** **{**  **throw** **new** GUIValidationException**(**"Δεν βρέθηκαν αποτελέσματα "  **+** "γι΄αυτό το εύρος ημερομηνιών στη βάση δεδομένων."**);**  **}** **else** **{**  displayChartPanel**.**removeAll**();**  **if** **(**"Μέσος όρος κερδών ανά κατηγορία(πίτα)"**.**equals**(**  String**.**valueOf**(**jComboBox1**.**getSelectedItem**())))** **{**  generatePieChart**(**dateRangeResults**);**  **}** **else** **{**  generateBarChart**(**dateRangeResults**);**  **}**  **}**  **}** **catch** **(**GUIValidationException e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε σφάλμα. " **+**  e**.**getMessage**(),**  "Δεν Αντλήθηκαν δεδομένα."**,**  JOptionPane**.**ERROR\_MESSAGE**);**  **}** **catch** **(**Exception e**)** **{**  JOptionPane**.**showMessageDialog**(null,** "Παρουσιάστηκε "  **+** "σφάλμα κατα την άντληση από την βάση δεδομένων." **+**  e**.**getMessage**(),** "Δεν Αντλήθηκαν δεδομένα."**,**  JOptionPane**.**ERROR\_MESSAGE**);**  System**.**out**.**println**(**e**);**  **}** **finally** **{**  stopWaiting**();**  //menuOverlay.setEnabled(JPnlInsertToDB, false);  menuOverlay**.**setEnabled**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  **return** **null;**  **}**  **};**  mySwingWorker**.**execute**();**  **}**  private void JPnlDisplayResultMouseEntered**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **true);**  **}**  private void JPnlDisplayResultMouseExited**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**  menuOverlay**.**setHilighted**((**JPanel**)** evt**.**getSource**(),** **false);**  **}**  // Generates suitable bar charts with given WinningColumn data.  private void generateBarChart**(**ArrayList**<**WinningColumn**>** dateRangeResults**)** **{**  // Create jfreechart dataset instance  DefaultCategoryDataset dataset **=** **new** DefaultCategoryDataset**();**  String categoryAxisString **=** ""**;**  String valueAxisString **=** ""**;**  double maximumBarWidth **=** 0**;**  /\* Populate dataset and specify bar chart parameters bar width and  category description depending on selected combo box option \*/  **if** **(**"Συχνότητα εμφάνισης αριθμών"**.**equals**(**String**.**valueOf**(**  jComboBox1**.**getSelectedItem**())))** **{**  categoryAxisString **=** "Νούμερο"**;**  valueAxisString **=** "Συχνότητα Εμφάνισης"**;**  ArrayList**<**Integer**>** numbers **=** **new** ArrayList**();**  int**[]** occurrences **=** **new** int**[**46**];;**  **for** **(**WinningColumn winningColumn **:** dateRangeResults**)** **{**  numbers**.**add**(**winningColumn**.**getN1**());**  numbers**.**add**(**winningColumn**.**getN2**());**  numbers**.**add**(**winningColumn**.**getN3**());**  numbers**.**add**(**winningColumn**.**getN4**());**  numbers**.**add**(**winningColumn**.**getN5**());**  **}**  **for** **(**int a **:** numbers**)** **{**  occurrences**[**a**]++;**  **}**  **for** **(**int i **=** 1**;** i **<** occurrences**.**length**;** i**++)** **{**  Integer numberInt **=** i**;**  String number **=** numberInt**.**toString**();**  dataset**.**setValue**(**occurrences**[**i**],** ""**,** number**);**  **}**  maximumBarWidth **=** 0.018**;**  **}** **else** **if** **(**"Συχνότητα εμφάνισης αριθμών Joker"**.**equals**(**  String**.**valueOf**(**jComboBox1**.**getSelectedItem**())))** **{**  categoryAxisString **=** "Νούμερο"**;**  valueAxisString **=** "Συχνότητα Εμφάνισης"**;**  ArrayList**<**Integer**>** numbers **=** **new** ArrayList**();**  int**[]** occurrences **=** **new** int**[**21**];**  **for** **(**WinningColumn winningColumn **:** dateRangeResults**)** **{**  numbers**.**add**(**winningColumn**.**getJ1**());**  **}**  **for** **(**int a **:** numbers**)** **{**  occurrences**[**a**]++;**  **}**  **for** **(**int i **=** 1**;** i **<** occurrences**.**length**;** i**++)** **{**  Integer numberInt **=** i**;**  String number **=** numberInt**.**toString**();**  dataset**.**setValue**(**occurrences**[**i**],** ""**,** number**);**  **}**  maximumBarWidth **=** 0.04**;**  **}** **else** **{** // "Μέσος όρος κερδών ανά κατηγορία" selected  categoryAxisString **=** "Κατηγορία"**;**  valueAxisString **=** "Συνολικά Κέρδη"**;**  ArrayList**<**Integer**>** drawIds **=** **new** ArrayList**();**  double**[]** payoutAmnt **=** **new** double**[**9**];**  int gameCounter **=** 0**;**  // Extract drawIds from aquired WiiningColumn records  **for** **(**WinningColumn winningColumn **:** dateRangeResults**)** **{**  WinningColumnPK pK **=** winningColumn**.**getWinningColumnPK**();**  drawIds**.**add**(**pK**.**getDrawId**());**  **}**  // Retrieve GameDrawResults records of drawIds specified above  ArrayList**<**ArrayList**<**GameDrawResults**>>** totalGameDrawResults **=**  DBIntegrator**.**getDrawResultsByDrawId**(**drawIds**,** 5104**);**  **for** **(**ArrayList**<**GameDrawResults**>**singleGameDrawResults**:**totalGameDrawResults**)** **{**  gameCounter **+=** 1**;**  **for** **(**GameDrawResults drawResults **:** singleGameDrawResults**)** **{**  double totalPayout **=** drawResults**.**getTotalPayout**();**  GameDrawResultsPK gameDrawResultsPK **=**  drawResults**.**getGameDrawResultsPK**();**  Integer categoryDescr **=** Integer**.**valueOf**(**  gameDrawResultsPK**.**getWinningCategoryDescr**());**  payoutAmnt**[**categoryDescr**]** **+=** totalPayout**;**  **}**  **}**  **for** **(**int i **=** 1**;** i **<** payoutAmnt**.**length**;** i**++)** **{**  String categoryDesc **=** **new** String**();**  // Define category description  **switch** **(**i**)** **{**  **case** 1**:**  categoryDesc **=** "5+1"**;**  **break;**  **case** 2**:**  categoryDesc **=** "5"**;**  **break;**  **case** 3**:**  categoryDesc **=** "4+1"**;**  **break;**  **case** 4**:**  categoryDesc **=** "4"**;**  **break;**  **case** 5**:**  categoryDesc **=** "3+1"**;**  **break;**  **case** 6**:**  categoryDesc **=** "3"**;**  **break;**  **case** 7**:**  categoryDesc **=** "2+1"**;**  **break;**  **case** 8**:**  categoryDesc **=** "1+1"**;**  **break;**  **}**  dataset**.**setValue**(**removeDecimal**((**  payoutAmnt**[**i**]** **/** gameCounter**),** 1**),** ""**,** categoryDesc**);**  **}**  maximumBarWidth **=** 0.1**;**  **}**    // Chart axis instantiation and settings  CategoryAxis categoryAxis **=** **new** CategoryAxis**(**categoryAxisString**);**  categoryAxis**.**setLowerMargin**(**.01**);**  categoryAxis**.**setCategoryMargin**(**.01**);**  categoryAxis**.**setUpperMargin**(**.01**);**  categoryAxis**.**setCategoryLabelPositions**(**CategoryLabelPositions**.**STANDARD**);**    ValueAxis valueAxis **=** **new** NumberAxis**(**valueAxisString**);**  valueAxis**.**setAxisLineVisible**(** **false** **);**  valueAxis**.**setTickMarksVisible**(** **false** **);**  // Create chart bar rednerer(settings must be defined after theme is applied)  StackedBarRenderer renderer **=** **new** StackedBarRenderer**();**    // Create CategoryPlot instance with axis and renderer parameters specified above  CategoryPlot plot **=** **new** CategoryPlot**(**dataset**,**  categoryAxis**,**  valueAxis**,**  renderer**);**    // CategoryPlot settings  plot**.**setOrientation**(**PlotOrientation**.**VERTICAL**);**  plot**.**getRangeAxis**().**setStandardTickUnits**(**  NumberAxis**.**createIntegerTickUnits**()** **);**  plot**.**setOutlineVisible**(** **false** **);**  plot**.**setRangeGridlineStroke**(** **new** BasicStroke**()** **);**  // Create chart instace  JFreeChart chart **=** **new** JFreeChart**(**""**,**  JFreeChart**.**DEFAULT\_TITLE\_FONT**,**  plot**,**  **true);**  // Create chart theme and specify colors  StandardChartTheme theme **=** **(**StandardChartTheme**)**org**.**jfree**.**chart**.**  StandardChartTheme**.**createJFreeTheme**();**  theme**.**setRangeGridlinePaint**(** **new** java**.**awt**.**Color**(**192**,** 192**,** 192**));**  theme**.**setPlotBackgroundPaint**(** Color**.**white **);**  theme**.**setChartBackgroundPaint**(** Color**.**white **);**  theme**.**setGridBandPaint**(** Color**.**red **);**  theme**.**setAxisOffset**(** **new** RectangleInsets**(**0**,**0**,**0**,**0**)** **);**  theme**.**setBarPainter**(new** StandardBarPainter**());**  theme**.**apply**(** chart **);**  // Chart specific settings  chart**.**removeLegend**();**  chart**.**setTextAntiAlias**(** **true** **);**  chart**.**setAntiAlias**(** **true** **);**    // Bar renderer settings(bar width, bar & label color, shadow e.t.c)  renderer**.**setSeriesPaint**(**0**,** **new** java**.**awt**.**Color**(**50**,** 122**,** 194**));**  renderer**.**setDefaultItemLabelPaint**(new** java**.**awt**.**Color**(**245**,** 245**,** 245**));**  renderer**.**setMaximumBarWidth**(**maximumBarWidth**);**  renderer**.**setSeriesItemLabelsVisible**(**0**,** **true);**  renderer**.**setSeriesItemLabelGenerator**(**0**,**  **new** StandardCategoryItemLabelGenerator**());**  renderer**.**setShadowVisible**(** **true** **);**  renderer**.**setShadowXOffset**(** 2 **);**  renderer**.**setShadowYOffset**(** 0 **);**  renderer**.**setShadowPaint**(** **new** java**.**awt**.**Color**(**192**,** 192**,** 192**)** **);**  ChartPanel chartPanel **=** **new** ChartPanel**(**chart**);**  displayChartPanel**.**setLayout**(new** java**.**awt**.**BorderLayout**());**  displayChartPanel**.**add**(**chartPanel**);** // Display Chart  displayChartPanel**.**validate**();**  **}**    // Generates pie chart that shows total payout of each joker game category  private void generatePieChart**(**ArrayList**<**WinningColumn**>** dateRangeResults**)** **{**  DefaultPieDataset dataset **=** **new** DefaultPieDataset**();**  ArrayList**<**Integer**>** drawIds **=** **new** ArrayList**();**  double**[]** payoutAmnt **=** **new** double**[**9**];**  int gameCounter **=** 0**;**  // Extract drawIds from aquired WiiningColumn records  **for** **(**WinningColumn winningColumn **:** dateRangeResults**)** **{**  WinningColumnPK pK **=** winningColumn**.**getWinningColumnPK**();**  drawIds**.**add**(**pK**.**getDrawId**());**  **}**  // Retrieve GameDrawResults records of drawIds specified above  ArrayList**<**ArrayList**<**GameDrawResults**>>** totalGameDrawResults **=**  DBIntegrator**.**getDrawResultsByDrawId**(**drawIds**,** 5104**);**  **for** **(**ArrayList**<**GameDrawResults**>** singleGameDrawResults **:** totalGameDrawResults**)** **{**  gameCounter **+=** 1**;**  **for** **(**GameDrawResults drawResults **:** singleGameDrawResults**)** **{**  double totalPayout **=** drawResults**.**getTotalPayout**();**  GameDrawResultsPK gameDrawResultsPK **=**  drawResults**.**getGameDrawResultsPK**();**  Integer categoryDescr **=** Integer**.**valueOf**(**  gameDrawResultsPK**.**getWinningCategoryDescr**());**  payoutAmnt**[**categoryDescr**]** **+=** totalPayout**;**  **}**  **}**  **for** **(**int i **=** 1**;** i **<** payoutAmnt**.**length**;** i**++)** **{**  String categoryDesc **=** **new** String**();**  **switch** **(**i**)** **{**  // Define category description  **case** 1**:**  categoryDesc **=** "5+1"**;**  **break;**  **case** 2**:**  categoryDesc **=** "5"**;**  **break;**  **case** 3**:**  categoryDesc **=** "4+1"**;**  **break;**  **case** 4**:**  categoryDesc **=** "4"**;**  **break;**  **case** 5**:**  categoryDesc **=** "3+1"**;**  **break;**  **case** 6**:**  categoryDesc **=** "3"**;**  **break;**  **case** 7**:**  categoryDesc **=** "2+1"**;**  **break;**  **case** 8**:**  categoryDesc **=** "1+1"**;**  **break;**  **}**  dataset**.**setValue**(**categoryDesc**,** removeDecimal**(**  payoutAmnt**[**i**]** **/** gameCounter**,** 1**));**  **}**  // Create pie chart instace  JFreeChart chart **=** ChartFactory**.**createPieChart3D**(**  ""**,** // chart title  dataset**,** // data  **true,** // include legend  **true,**  **false);**  chart**.**setBackgroundPaint**(new** java**.**awt**.**Color**(**255**,** 255**,** 255**));**  chart**.**setBorderPaint**(new** java**.**awt**.**Color**(**190**,** 139**,** 6**));**    // Create PiePlot3D instance and specify settings and colors  final PiePlot3D plot **=** **(**PiePlot3D**)** chart**.**getPlot**();**  plot**.**setStartAngle**(**270**);**  plot**.**setForegroundAlpha**(**0.60f**);**  plot**.**setInteriorGap**(**0.02**);**  plot**.**setSectionPaint**(**"5+1"**,** **new** java**.**awt**.**Color**(**32**,** 133**,** 236**));**  plot**.**setSectionPaint**(**"5"**,** **new** java**.**awt**.**Color**(**114**,** 180**,** 235**));**  plot**.**setSectionPaint**(**"4+1"**,** **new** java**.**awt**.**Color**(**235**,** 170**,** 0**));**  plot**.**setSectionPaint**(**"4"**,** **new** java**.**awt**.**Color**(**255**,** 212**,** 20**));**  plot**.**setSectionPaint**(**"3+1"**,** **new** java**.**awt**.**Color**(**219**,** 219**,** 219**));**  plot**.**setSectionPaint**(**"3"**,** **new** java**.**awt**.**Color**(**110**,** 110**,** 110**));**  plot**.**setSectionPaint**(**"2+1"**,** **new** java**.**awt**.**Color**(**50**,** 50**,** 50**));**  plot**.**setSectionPaint**(**"1+1"**,** **new** java**.**awt**.**Color**(**10**,** 65**,** 122**));**  plot**.**setLabelBackgroundPaint**(new** java**.**awt**.**Color**(**255**,** 237**,** 161**));**  plot**.**setBackgroundPaint**(new** java**.**awt**.**Color**(**255**,** 255**,** 255**));**  plot**.**setOutlinePaint**(new** java**.**awt**.**Color**(**255**,** 255**,** 255**));**  plot**.**setLegendItemShape**(new** Rectangle**(**12**,**9**));**    TextTitle legendText **=** **new** TextTitle**(**"Κατηγορίες επιτυχιών: "**);**  legendText**.**setPosition**(**RectangleEdge**.**BOTTOM**);**  chart**.**addSubtitle**(**legendText**);**    ChartPanel chartPanel **=** **new** ChartPanel**(**chart**);**  displayChartPanel**.**setLayout**(new** java**.**awt**.**BorderLayout**());**  displayChartPanel**.**add**(**chartPanel**);**  displayChartPanel**.**validate**();**  **}**    //Removes decimal points of paremeter value beyond given paremeter decimalpoint  private static double removeDecimal**(**double value**,** int decimalpoint**)** **{**  value **=** value **\*** Math**.**pow**(**10**,** decimalpoint**);**  value **=** Math**.**floor**(**value**);**  value **=** value **/** Math**.**pow**(**10**,** decimalpoint**);**  **return** value**;**  **}**    private void startWaiting**(**JPanel panel**)** **{**  progressBar**.**setVisible**(true);**  progressBar**.**setIndeterminate**(true);**  progressBar**.**setValue**(**100**);**  JPnlMain**.**setCursor**(**Cursor**.**getPredefinedCursor**(**Cursor**.**WAIT\_CURSOR**));**  panel**.**grabFocus**();**  **}**  private void stopWaiting**()** **{**  JPnlMain**.**setCursor**(null);** //turn off the wait cursor  progressBar**.**setIndeterminate**(false);**  progressBar**.**setVisible**(false);**  **}**  // Variables declaration - do not modify  private javax**.**swing**.**JLabel JLbDateRangeTitle**;**  private javax**.**swing**.**JLabel JLbFromDate**;**  private javax**.**swing**.**JLabel JLbShowResultsIcon**;**  private javax**.**swing**.**JLabel JLbShowResultsText**;**  private javax**.**swing**.**JLabel JLbStatsTitle**;**  private javax**.**swing**.**JLabel JLbToDate**;**  private javax**.**swing**.**JPanel JPnlChartContainer**;**  private javax**.**swing**.**JPanel JPnlDateRange**;**  private javax**.**swing**.**JPanel JPnlDisplayResult**;**  private javax**.**swing**.**JPanel JPnlFiller**;**  private javax**.**swing**.**JPanel JPnlMain**;**  private javax**.**swing**.**JPanel JPnlStatsSelection**;**  private javax**.**swing**.**JPanel displayChartPanel**;**  private javax**.**swing**.**JComboBox**<**String**>** jComboBox1**;**  private javax**.**swing**.**JLabel jLabel6**;**  private org**.**jdesktop**.**swingx**.**JXDatePicker jXDatePickerFrom**;**  private org**.**jdesktop**.**swingx**.**JXDatePicker jXDatePickerTo**;**  private javax**.**swing**.**JProgressBar progressBar**;**  // End of variables declaration  **}** |

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:** **DrawsBarChart.Java** |
| package gui**;**  **import** DataIngestion**.**DBIntegrator**;**  **import** DataIngestion**.**DBIntegrator**.**AggregatedDrawDataByDraw**;**  **import** java**.**awt**.**BasicStroke**;**  **import** java**.**awt**.**Color**;**  **import** org**.**jfree**.**chart**.**ChartFactory**;**  **import** org**.**jfree**.**chart**.**ChartPanel**;**  **import** org**.**jfree**.**chart**.**JFreeChart**;**  **import** org**.**jfree**.**chart**.**plot**.**PlotOrientation**;**  **import** org**.**jfree**.**data**.**category**.**CategoryDataset**;**  **import** org**.**jfree**.**data**.**category**.**DefaultCategoryDataset**;**  **import** java**.**awt**.**Dimension**;**  **import** java**.**sql**.**SQLException**;**  **import** java**.**util**.**ArrayList**;**  **import** java**.**util**.**List**;**  **import** org**.**jfree**.**chart**.**ChartMouseEvent**;**  **import** org**.**jfree**.**chart**.**ChartMouseListener**;**  **import** org**.**jfree**.**chart**.**StandardChartTheme**;**  **import** org**.**jfree**.**chart**.**renderer**.**category**.**BarRenderer**;**  **import** org**.**jfree**.**chart**.**renderer**.**category**.**StandardBarPainter**;**  **import** org**.**jfree**.**chart**.**ui**.**RectangleInsets**;**  public class DrawsBarChart **extends** javax**.**swing**.**JPanel **{**  private String fromDate**;**  private String toDate**;**  private String monthName**;**  private int gameID**;**  public DrawsBarChart**(**int gameID**,** String fromDate**,** String toDate**,** String monthName**)** **throws** SQLException **{**  **this.**gameID **=** gameID**;**  **this.**fromDate **=** fromDate**;**  **this.**toDate **=** toDate**;**  **this.**monthName **=** monthName**;**  initUI**();**  **}**  private void initUI**()** **throws** SQLException **{**  DefaultCategoryDataset dataset **=** **new** DefaultCategoryDataset**();**  List**<**AggregatedDrawDataByDraw**>** aggregatedDrawDataByDrawList **=** **new** ArrayList**<>();**  //Prepare data set  aggregatedDrawDataByDrawList **=** DBIntegrator**.**getAggregatedDrawDataByDraw**(**gameID**,** fromDate**,** toDate**);**  **for** **(**AggregatedDrawDataByDraw aggregatedDrawDataByDraw **:** aggregatedDrawDataByDrawList**)** **{**  dataset**.**addValue**(**aggregatedDrawDataByDraw**.**getPayoutAmount**(),**  monthName**,**  String**.**valueOf**(**aggregatedDrawDataByDraw**.**getDrawID**()));**  **}**  //Create chart  JFreeChart chart **=** ChartFactory**.**createBarChart**(**  "Κέρδη ανα Κλήρωση"**,** //Chart Title  ""**,**//Κληρώσεις", // Category axis  ""**,**//Κέρδη", // Value axis  dataset**,**  PlotOrientation**.**VERTICAL**,**  **true,** **true,** **false**  **);**  StandardChartTheme theme **=** **(**StandardChartTheme**)**org**.**jfree**.**chart**.**StandardChartTheme**.**createJFreeTheme**();**  theme**.**setRangeGridlinePaint**(** **new** java**.**awt**.**Color**(**192**,** 192**,** 192**));**  theme**.**setPlotBackgroundPaint**(** Color**.**white **);**  theme**.**setChartBackgroundPaint**(** Color**.**white **);**  theme**.**setGridBandPaint**(** Color**.**red **);**  theme**.**setAxisOffset**(** **new** RectangleInsets**(**0**,**0**,**0**,**0**)** **);**  theme**.**setBarPainter**(new** StandardBarPainter**());**  theme**.**apply**(** chart **);**    chart**.**getCategoryPlot**().**setOutlineVisible**(** **false** **);**  chart**.**getCategoryPlot**().**getRangeAxis**().**setAxisLineVisible**(** **false** **);**  chart**.**getCategoryPlot**().**getRangeAxis**().**setTickMarksVisible**(** **false** **);**  chart**.**getCategoryPlot**().**setRangeGridlineStroke**(** **new** BasicStroke**()** **);**    chart**.**setTextAntiAlias**(** **true** **);**  chart**.**setAntiAlias**(** **true** **);**  chart**.**getCategoryPlot**().**getRenderer**().**setSeriesPaint**(** 0**,** **new** java**.**awt**.**Color**(**50**,** 122**,** 194**));**  BarRenderer renderer **=** **(**BarRenderer**)** chart**.**getCategoryPlot**().**getRenderer**();**  renderer**.**setShadowVisible**(** **true** **);**  renderer**.**setShadowXOffset**(** 2 **);**  renderer**.**setShadowYOffset**(** 0 **);**  renderer**.**setShadowPaint**(** **new** java**.**awt**.**Color**(**192**,** 192**,** 192**)** **);**  renderer**.**setMaximumBarWidth**(**0.06**);**    chart**.**removeLegend**();**  chart**.**setTitle**(**""**);**  ChartPanel cp **=** **new** ChartPanel**(**chart**)** **{**  @Override  public Dimension getPreferredSize**()** **{**  **return** **new** Dimension**(**650**,** 150**);**  **}**  **};**  cp**.**setMouseWheelEnabled**(true);**  cp**.**addChartMouseListener**(new** ChartMouseListener**()** **{**  public void chartMouseClicked**(**ChartMouseEvent e**)** **{**  System**.**out**.**println**(**e**.**getEntity**().**getToolTipText**());**  **}**  public void chartMouseMoved**(**ChartMouseEvent e**)** **{**  **}**  **});**  add**(**cp**);**  **}**  **}** |

|  |
| --- |
| **Πακέτο:** **gui**  **Κλάση:MonthsBarChart.Java** |
| package gui**;**  **import** DataIngestion**.**DBIntegrator**;**  **import** DataIngestion**.**DBIntegrator**.**AggregatedDrawDataByMonth**;**  **import** java**.**awt**.**BasicStroke**;**  **import** java**.**awt**.**Color**;**  **import** org**.**jfree**.**chart**.**ChartFactory**;**  **import** org**.**jfree**.**chart**.**ChartPanel**;**  **import** org**.**jfree**.**chart**.**JFreeChart**;**  **import** org**.**jfree**.**chart**.**plot**.**PlotOrientation**;**  **import** org**.**jfree**.**data**.**category**.**DefaultCategoryDataset**;**  **import** java**.**awt**.**Dimension**;**  **import** java**.**awt**.**Point**;**  **import** java**.**sql**.**SQLException**;**  **import** java**.**util**.**ArrayList**;**  **import** java**.**util**.**HashMap**;**  **import** java**.**util**.**List**;**  **import** javax**.**swing**.**JDialog**;**  **import** javax**.**swing**.**JPanel**;**  **import** org**.**jfree**.**chart**.**ChartMouseEvent**;**  **import** org**.**jfree**.**chart**.**ChartMouseListener**;**  **import** org**.**jfree**.**chart**.**StandardChartTheme**;**  **import** org**.**jfree**.**chart**.**renderer**.**category**.**BarRenderer**;**  **import** org**.**jfree**.**chart**.**renderer**.**category**.**StandardBarPainter**;**  **import** org**.**jfree**.**chart**.**ui**.**RectangleInsets**;**  public class MonthsBarChart **extends** javax**.**swing**.**JPanel **{**  private String fromDate**;**  private String toDate**;**  private int gameID**;**  private String year**;**  private HashMap**<**Integer**,** String**>** monthTable**;**  JPanel sourcePanel**;**  public MonthsBarChart**(**int gameID**,** String fromDate**,** String toDate**,** JPanel sourcePanel**)** **throws** SQLException **{**  **this.**gameID **=** gameID**;**  **this.**fromDate **=** fromDate**;**  **this.**toDate **=** toDate**;**  **this.**year **=** fromDate**.**substring**(**0**,** 4**);**  monthTable **=** **new** HashMap**<>();**  **this.**sourcePanel **=** sourcePanel**;**  initMonthMap**();**  initUI**();**  **}**  private void initUI**()** **throws** SQLException **{**  DefaultCategoryDataset dataset **=** **new** DefaultCategoryDataset**();**  List**<**AggregatedDrawDataByMonth**>** aggregatedDrawDataByMonthList **=** **new** ArrayList**<>();**  //Prepare data set  aggregatedDrawDataByMonthList **=** DBIntegrator**.**getAggregatedDrawDataByMonth**(**gameID**,** fromDate**,** toDate**);**  **for** **(**AggregatedDrawDataByMonth aggregatedDrawDataByMonth **:** aggregatedDrawDataByMonthList**)** **{**  dataset**.**addValue**(**aggregatedDrawDataByMonth**.**getPayoutAmount**(),**  "Μήνας"**,**  String**.**valueOf**(**aggregatedDrawDataByMonth**.**getMonthID**()));**  **}**  //Create chart  JFreeChart chart **=** ChartFactory**.**createBarChart**(**  "Μήνας"**,** //Chart Title  ""**,**//Κληρώσεις", // Category axis  ""**,**//Κέρδη", // Value axis  dataset**,**  PlotOrientation**.**VERTICAL**,**  **true,** **true,** **false**  **);**  //Trim chart  chart**.**removeLegend**();**  chart**.**setTitle**(**""**);**    StandardChartTheme theme **=** **(**StandardChartTheme**)**org**.**jfree**.**chart**.**StandardChartTheme**.**createJFreeTheme**();**  theme**.**setRangeGridlinePaint**(** **new** java**.**awt**.**Color**(**192**,** 192**,** 192**));**  theme**.**setPlotBackgroundPaint**(** Color**.**white **);**  theme**.**setChartBackgroundPaint**(** Color**.**white **);**  theme**.**setGridBandPaint**(** Color**.**red **);**  theme**.**setAxisOffset**(** **new** RectangleInsets**(**0**,**0**,**0**,**0**)** **);**  theme**.**setBarPainter**(new** StandardBarPainter**());**  theme**.**apply**(** chart **);**    chart**.**getCategoryPlot**().**setOutlineVisible**(** **false** **);**  chart**.**getCategoryPlot**().**getRangeAxis**().**setAxisLineVisible**(** **false** **);**  chart**.**getCategoryPlot**().**getRangeAxis**().**setTickMarksVisible**(** **false** **);**  chart**.**getCategoryPlot**().**setRangeGridlineStroke**(** **new** BasicStroke**()** **);**  chart**.**setTextAntiAlias**(** **true** **);**  chart**.**setAntiAlias**(** **true** **);**  chart**.**getCategoryPlot**().**getRenderer**().**setSeriesPaint**(** 0**,** **new** java**.**awt**.**Color**(**50**,** 122**,** 194**));**  BarRenderer renderer **=** **(**BarRenderer**)** chart**.**getCategoryPlot**().**getRenderer**();**  renderer**.**setShadowVisible**(** **true** **);**  renderer**.**setShadowXOffset**(** 2 **);**  renderer**.**setShadowYOffset**(** 0 **);**  renderer**.**setShadowPaint**(** **new** java**.**awt**.**Color**(**192**,** 192**,** 192**)** **);**  System**.**out**.**println**(**aggregatedDrawDataByMonthList**.**size**());**  **if** **(**aggregatedDrawDataByMonthList**.**size**()<**4**){**  renderer**.**setMaximumBarWidth**(**0.2**);**  **}**  **else** **if(**aggregatedDrawDataByMonthList**.**size**()<**7**){**  renderer**.**setMaximumBarWidth**(**0.1**);**  **}**  **else{**  renderer**.**setMaximumBarWidth**(**0.06**);**  **}**  ChartPanel cp **=** **new** ChartPanel**(**chart**)** **{**  @Override  public Dimension getPreferredSize**()** **{**  **return** **new** Dimension**(**650**,** 150**);**  **}**  **};**  cp**.**setMouseWheelEnabled**(true);**  //Add on click listener to perform drill down  cp**.**addChartMouseListener**(new** ChartMouseListener**()** **{**  public void chartMouseClicked**(**ChartMouseEvent e**)** **{**  //Drill Down to Month  DiplayDataByDrawFrame diplayDataFrame **=** **new** DiplayDataByDrawFrame**();**  diplayDataFrame**.**setYear**(**year**);**  diplayDataFrame**.**setMonth**(**getSelectedMonth**(**e**.**getEntity**().**getToolTipText**()));**  diplayDataFrame**.**execForm**();**  final JDialog frame **=** **new** JDialog**(**diplayDataFrame**,** "Προβολή Δεδομένων Τζόκερ ανα μήνα"**,** **true);**  frame**.**setUndecorated**(true);** // <-- the title bar is removed here  frame**.**getContentPane**().**add**(**diplayDataFrame**.**getContentPane**());**  frame**.**setResizable**(false);**  frame**.**pack**();**  Point point **=** sourcePanel**.**getLocationOnScreen**();**  frame**.**setLocation**(new** Point**(**point**.**x**,** point**.**y**));**  frame**.**setVisible**(true);**  **}**  public void chartMouseMoved**(**ChartMouseEvent e**)** **{**  **}**  **});**  add**(**cp**);**  **}**  //Inditify Selected month and return it to client  private int getSelectedMonth**(**String tooltipText**)** **{**  //Get Month  String**[]** month **=** tooltipText**.**split**(**","**);**  int index **=** month**[**1**].**indexOf**(**')'**);**  String res **=** month**[**1**].**substring**(**1**,** index**);**  int ss **=** Integer**.**parseInt**(**res**.**trim**());**  **return** ss**;**  **}**  private void initMonthMap**()** **{**  monthTable**.**put**(**1**,** "Ιανουάριος"**);**  monthTable**.**put**(**2**,** "Φεβρουάριος"**);**  monthTable**.**put**(**3**,** "Μάρτιος"**);**  monthTable**.**put**(**4**,** "Απρίλιος"**);**  monthTable**.**put**(**5**,** "Μάιος"**);**  monthTable**.**put**(**6**,** "Ιούνιος"**);**  monthTable**.**put**(**7**,** "Ιούλιος"**);**  monthTable**.**put**(**8**,** "Αύγουστος"**);**  monthTable**.**put**(**9**,** "Σεπτέμβριος"**);**  monthTable**.**put**(**10**,** "Οκτώβριος"**);**  monthTable**.**put**(**11**,** "Νοέμβριος"**);**  monthTable**.**put**(**12**,** "Δεκέμβριος"**);**  **}**  **}** |

# ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ

|  |  |
| --- | --- |
| **Απαίτηση** | **Εικόνα** |
| **R1**  Κεντρικό Μενού | Εικόνα 3.1 - Κεντρικό Μενού  Με την εκτέλεση της εφαρμογής, εμφανίζεται το κεντρικό μενού στο οποίο παρουσιάζονται οι εξής επιλογές:   1. Διαχείριση Δεδομένων Τζόκερ 2. Προβολή Δεδομένων Τζόκερ 3. Προβολή Στατιστικών Τζόκερ 4. Έξοδος   Επιλέγοντας με τον κέρσορα ένα από τα πλήκτρα (1-3), εμφανίζεται στα δεξιά του παραθύρου η αντίστοιχη οθόνη. Επιλέγοντας το πλήκτρο **«Έξοδος»** εμφανίζεται μήνυμα επιβεβαίωσης εξόδου από την εφαρμογή (εικόνα 3.2).    Εικόνα 3.2 - Μήνυμα προτροπής για επιβεβαίωση εξόδου από την εφαρμογή |
| **R2**  Διαχείριση Δεδομένων Τζόκερ | Εικόνα 3.3 - Διαχείριση Δεδομένων Τζόκερ – Άντληση Δεδομένων με Κωδικό Κλήρωσης  Στην οθόνη διαχείρισης δεδομένων Τζόκερ, δίνεται η δυνατότητα άντλησης δεδομένων για οποιαδήποτε κλήρωση Τζόκερ έχει πραγματοποιηθεί από την αρχή του παιχνιδιού έως και την τρέχουσα ημερομηνία. H άντληση των δεδομένων των κληρώσεων μπορεί να γίνει με δυο διαφορετικούς τρόπους, είτε βάσει του κωδικού κλήρωσης του παιχνιδιού, όπου προβάλλονται λεπτομέρειες για το εν λόγο παιχνίδι, είτε βάσει ενός επιλεγμένου εύρους ημερομηνιών, όπου προβάλλονται λεπτομέρειες για όσα παιχνίδια έχουν πραγματοποιηθεί εντός του ορισμένου εύρους.  Για την άντληση των δεδομένων βάσει του κωδικού κλήρωσης, ο χρήστης επιλέγει την ένδειξη **«Επιλογή Δεδομένων με Κωδικό Κλήρωσης»** οπότε και ενεργοποιείται το πεδίο **«Κωδικός Κλήρωσης»** όπου μπορεί να συμπληρώσει τον κωδικό (εικόνα 3.3). Πατώντας το πλήκτρο **«Άντληση Δεδομένων»** ξεκινά η διαδικασία της άντλησης των δεδομένων της κλήρωσης.    Εικόνα 3.4 - Διαχείριση Δεδομένων Τζόκερ – Άντληση δεδομένων βάσει εύρους ημερομηνιών  Για την άντληση των δεδομένων βάσει του εύρους ημερομηνιών, ο χρήστης επιλέγει την ένδειξη **«Επιλογή Δεδομένων με εύρος ημερομηνιών»** οπότε ενεργοποιούνται τα πεδία επιλογής ημερομηνίας. Επιλέγοντας με τον κέρσορα την κάθετη ένδειξη  , αναδύεται ημερολόγιο από το οποίο ο χρήστης μπορεί να διαλέξει τις ημερομηνίες που επιθυμεί (εικόνα 3.4). Έπειτα επιλέγοντας **«Άντληση Δεδομένων»** ξεκινάει η διαδικασία της άντλησης των δεδομένων των κληρώσεων που έχουν πραγματοποιηθεί στο ορισμένο εύρος ημερομηνιών. Στην περίπτωση που δεν έχει πραγματοποιηθεί κλήρωση στο επιλεγμένο εύρος ημερομηνιών ή δεν έχει συμπληρωθεί αρχική και τελική ημ/νια, εμφανίζεται σχετικό μήνυμα σφάλματος:  *Εικόνα 3.5 - Μήνυμα σε περίπτωση που δεν έχει συμπληρωθεί ημερομηνία*    Εικόνα 3.6 - Μήνυμα σε περίπτωση που δεν έχει πραγματοποιηθεί κλήρωση στο ορισμένο εύρος ημερομηνιών.        Εικόνα 3.7 - Διαχείριση Δεδομένων Τζόκερ – Παρουσίαση δεδομένων κληρώσεων  Με την ολοκλήρωση της άντλησης των δεδομένων, παρουσιάζονται στους πίνακες οι λεπτομέρειες της συγκεκριμένης κλήρωσης (ή κληρώσεων). Στην περίπτωση που έχουν αντληθεί δεδομένα για περισσότερες από μία κληρώσεις, επιλέγοντας από τον πρώτο πίνακα την γραμμή που αντιστοιχεί σε μία κλήρωση, εστιάζονται στον δεύτερο πίνακα οι πληροφορίες που αφορούν την συγκεκριμένη κλήρωση (εικόνα 3.7).  Mε την επιλογή του πλήκτρου **«Εισαγωγή στη Β.Δ.»** δίνεται η δυνατότητα στον χρήστη να αποθηκεύσει τα δεδομένα που παρουσιάζονται στους πίνακες. Εάν κάποια από τα δεδομένα των κληρώσεων που παρουσιάζονται υπάρχουν ήδη στη Β.Δ., τότε δίνεται η δυνατότητα να αποθηκευτούν, από αυτά που αντλήθηκαν, εκείνα το οποία δεν υπάρχουν ήδη στη Β.Δ (εικόνα 3.8). Ο χρήστης ενημερώνεται για το αποτέλεσμα του αιτήματος αποθήκευσης με κατάλληλο μήνυμα (εικόνα 3.9).  Με το πλήκτρο **«Διαγραφή από τη Β.Δ.»** ο χρήστης μπορεί να διαγράψει από την Β.Δ. τα στοιχεία που παρουσιάζονται στους πίνακες, εφόσον αυτά είναι ήδη αποθηκευμένα στη Β.Δ. Επιλέγοντας το πλήκτρο **«Διαγραφή από τη Β.Δ.»** εμφανίζεται μήνυμα επιβεβαίωσης του αιτήματος διαγραφής (εικόνα 3.10). Αφού ολοκληρωθεί η διαγραφή των δεδομένων εμφανίζεται κατάλληλο μήνυμα ενημέρωσης για το αποτέλεσμα του αιτήματος διαγραφής (εικόνα 3.11, 3.12).  Επιπλέον με το πλήκτρο **«Διαγραφή Όλων»** δίνεται η δυνατότητα διαγραφής του συνόλου των δεδομένων που βρίσκονται αποθηκευμένα στη Β.Δ., αφου προηγηθεί προτροπή για επιβεβαίωση του αιτήματος διαγραφής (εικόνα 3.13). Αφού ολοκληρωθεί η διαγραφή των δεδομένων εμφανίζεται μήνυμα ενημέρωσης για το αποτέλεσμα του αιτήματος διαγραφής (εικόνα 3.14).      Εικόνα 3.8-Μήνυμα προτροπής για την αποθήκευση των δεδομένων όσων κληρώσεων δεν βρέθηκαν στη Β.Δ.      Εικόνα 3.9 -Μηνύματα ενημέρωσης για το αποτέλεσμα του αιτήματος αποθήκευσης των δεδομένων στην Β.Δ.∙- μήνυμα επιτυχίας (αριστερά), μήνυμα αποτυχίας (δεξιά)    Εικόνα 3.10 - Μήνυμα προτροπής για επιβεβαίωση διαγραφής των δεδομένων από την Β.Δ    Εικόνα 3.11 - Μήνυμα ενημέρωσης για το αποτέλεσμα του αιτήματος διαγραφής των δεδομένων που αφορούν μια συγκεκριμένη κλήρωση    Εικόνα 3.12 - Μήνυμα ενημέρωσης για το αποτέλεσμα του αιτήματος διαγραφής των δεδομένων που αφορούν πολλαπλές κληρώσεις    Εικόνα 3.13- Μήνυμα προτροπής για επιβεβαίωση διαγραφής όλων των δεδομένων από τη Β.Δ    Εικόνα 3.14- Μήνυμα ενημέρωσης για το αποτέλεσμα του αιτήματος διαγραφής όλων των δεδομένων από τη Β.Δ |
| **R3**  Προβολή Δεδομένων Τζόκερ | Εικόνα 3.15 - Προβολή δεδομένων Τζόκερ – Επιλογή ημερολογιακού έτους  Στην οθόνη προβολής δεδομένων Τζόκερ, επιλέγοντας με τον κέρσορα την κάθετη ένδειξη  , εμφανίζονται τα χρονολογικά έτη από τα οποία μπορεί να επιλέξει ο χρήστης (εικόνα 3.15). Μετά την επιλογή του επιθυμητού έτους, ο χρήστης επιλέγει το πλήκτρο **«Εμφάνιση Αποτελεσμάτων»** και ξεκινά η διαδικασία συγκέντρωσης των δεδομένων.    Εικόνα 3.16 - Προβολή δεδομένων Τζόκερ – Συγκεντρωτικά δεδομένα έτους  Μόλις ολοκληρωθεί η διαδικασία συγκέντρωσης των δεδομένων, εμφανίζεται στην οθόνη το συνολικό πλήθος των κληρώσεων, τα διανεμόμενα κέρδη και το πλήθος των ΤΖΑΚ-ΠΟΤ για το συγκεκριμένο έτος. Ταυτόχρονα εμφανίζεται ένα γράφημα στο οποίο απεικονίζονται τα διανεμόμενα κέρδη, ανά μήνα, για το επιλεγμένο έτος. Τοποθετώντας τον κέρσορα πάνω σε μία από τις μπάρες του γραφήματος, εμφανίζεται το ακριβές ποσό των κερδών για τον συγκεκριμένο μήνα (εικόνα 3.16).  Στο σημείο αυτό, δίνεται η δυνατότητα προβολής των συγκεντρωτικών δεδομένων για έναν συγκεκριμένο μήνα, επιλέγοντας με τον κέρσορα την μπάρα του γραφήματος που αντιστοιχεί στον επιθυμητό μήνα.    Εικόνα 3.17 - Προβολή δεδομένων Τζόκερ – Συγκεντρωτικά δεδομένα μήνα  Αφού επιλεχθεί η μπάρα του επιθυμητού μήνα, προβάλλονται τα συγκεντρωτικά δεδομένα που αφορούν τον μήνα (πλήθος κληρώσεων, διανεμόμενα κέρδη, πλήθος ΤΖΑΚ- ΠΟΤ) και ένα γράφημα το οποίο απεικονίζει τα διανεμόμενα κέρδη ανά κλήρωση που πραγματοποιήθηκε κατά τον μήνα αυτό. Τοποθετώντας τον κέρσορα πάνω σε μια από τις μπάρες του γραφήματος, εμφανίζεται το ακριβές ποσό των κερδών για τη συγκεκριμένη κλήρωση. Υπάρχει επίσης η δυνατότητα προβολής δεδομένων για διαφορετικό μήνα. Επιλέγοντας την κάθετη ένδειξη  που αντιστοιχεί είτε στο έτος κλήρωσης είτε στο μήνα κλήρωσης, αναδύεται η λίστα με τα διαθέσιμα έτη και τους διαθέσιμους μήνες από τους οποίους μπορεί να επιλέξει ο χρήστης. Σε περίπτωση αλλαγής του έτους ή του μήνα, για να εμφανιστούν τα αντίστοιχα δεδομένα, ο χρήστης πρέπει να επιλέξει το πλήκτρο **«Εμφάνιση Αποτελεσμάτων»**. Τέλος επιλέγοντας το πλήκτρο **«Επιστροφή»,** επιστρέφουμε στην οθόνη προβολής συγκεντρωτικών δεδομένων του έτους. |
| **R4**  Προβολή στατιστικών δεδομένων Τζόκερ και εκτύπωση σε αρχείο pdf | Εικόνα 3.18 - Προβολή στατιστικών Τζόκερ – Προβολή σε πίνακα  Στην οθόνη προβολής στατιστικών Τζόκερ, επιλέγοντας το πλήκτρο **«Προβολή σε Πίνακα»,** εμφανίζονται δυο πίνακες με τα στατιστικά που αφορούν τους αριθμούς της νικήτριας στήλης του Τζόκερ. Στον πρώτο πίνακα απεικονίζονται, η συχνότητα εμφάνισης και η καθυστέρηση για κάθε αριθμό πεντάδας της νικητήριας στήλης, ενώ στον δεύτερο τα αντίστοιχα δεδομένα για τον αριθμό Τζόκερ της νικητήριας στήλης (εικόνα 3.18).  Ο χρήστης έχει την δυνατότητα να εκτυπώσει σε αρχείο pdf τους πίνακες που εμφανίζονται στην οθόνη. Επιλέγοντας το πλήκτρο **«Εκτύπωση σε PDF»,** εμφανίζεται ένα νέο παράθυρο (εικόνα 3.19) όπου επιλέγει τον φάκελο στον οποίο θα αποθηκευτεί το pdf αρχείο, συμπληρώνει το όνομα του αρχείου και πατάει το πλήκτρο **«Save»** για να το αποθηκεύσει. Επίσης δίνεται η δυνατότητα άμεσης προβολής του pdf αρχείου επιλέγοντας την ένδειξη.  Με την ολοκλήρωση της διαδικασίας αποθήκευσης του αρχείου, εμφανίζεται μήνυμα το οποίο ενημερώνει τον χρήστη για την τοποθεσία στην οποία αποθηκεύτηκε το pdf αρχείο (εικόνα 3.20).    Εικόνα 3.19 - Παράθυρο επιλογής ονόματος και τοποθεσίας για την αποθήκευση του pdf    Εικόνα 3.20 - Μήνυμα ενημέρωσης για την τοποθεσία στην οποία αποθηκεύτηκε το pdf αρχείο |
| **R5**  Προβολή στατιστικών στοιχείων κληρώσεων σε γραφική μορφή | Εικόνα 3.21 - Προβολή στατιστικών Τζόκερ – Προβολή σε γραφική μορφή  Στην οθόνη προβολής στατιστικών Τζόκερ, επιλέγοντας το πλήκτρο **«Προβολή σε Γραφική Μορφή»** εμφανίζονται τα πεδία επιλογής των δεδομένων που θα χρησιμοποιηθούν για την δημιουργία του γραφήματος (εικόνα 3.21). Ο χρήστης έχει την δυνατότητα να επιλέξει το εύρος των ημερομηνιών που επιθυμεί και μια από τις παρακάτω επιλογές όσον αφορά το στατιστικά δεδομένα που θα χρησιμοποιηθούν για να παραχθεί το γράφημα:   1. Συχνότητα εμφάνισης αριθμών 2. Συχνότητα εμφάνισης αριθμών Τζόκερ 3. Μέσος όρος κερδών ανά κατηγορία 4. Μέσος όρος κερδών ανά κατηγορία(πίτα)   Στη συνέχεια, επιλέγοντας το πλήκτρο **«Εμφάνιση Αποτελεσμάτων»** εμφανίζεται το αντίστοιχο γράφημα. Ενδεικτικά, για τις ημερομηνίες 01/02/21 έως και 04/03/21 τα είδη γραφημάτων που μπορούν να παραχθούν παρουσιάζονται στις εικόνες 3.22, 3.23, 3.24 και 3.25. Εάν έχει παραχθεί γράφημα πίτας, τοποθετώντας τον κέρσορα σε ένα από τα τμήματα του γραφήματος, εμφανίζεται το ακριβές ποσό των κερδών για τη συγκεκριμένη κατηγορία (εικόνα 3.22).  Στην περίπτωση που δεν υπάρχουν δεδομένα για κληρώσεις που πραγματοποιήθηκαν στο επιλεγμένο εύρος ημερομηνιών ή δεν έχει συμπληρωθεί αρχική και τελική ημερομηνία, εμφανίζεται κατάλληλο μήνυμα σφάλματος (εικόνες 3.26 και 3.27).      Εικόνα 3.22 - Προβολή στατιστικών Τζόκερ – Μέσος όρος κερδών ανά κατηγορία (γράφημα πίτας)    Εικόνα 3.23 - Προβολή στατιστικών Τζόκερ – Συχνότητα εμφάνισης αριθμών    Εικόνα 3.24 - Προβολή στατιστικών Τζόκερ – Συχνότητα εμφάνισης αριθμών Τζόκερ    Εικόνα 3.25 - Προβολή στατιστικών Τζόκερ – Μέσος όρος κερδών ανά κατηγορία    Εικόνα 3.26 - Μήνυμα σε περίπτωση που δεν βρέθηκαν δεδομένα για το επιλεγμένο εύρος ημερομηνιών    Εικόνα 3.27 - Μήνυμα σε περίπτωση που δεν έχει συμπληρωθεί ημερομηνία |

# ΑΝΑΦΟΡΕΣ

Συγγράμματα:

* Β. Βεσκούκης*, Τεχνολογία Λογισμικού ΙΙ*, τόμος Β’, 2η έκδ., ΕΑΠ, Πάτρα, 2008
* Κ. Θραμπουλίδης*, Γλώσσες Προγραμματισμού ΙΙ (Αντικειμενοστρεφής Προγραμματισμός)*, τόμος Γ’, ΕΑΠ, Πάτρα, 2001
* Μ. Ξένος κ.α., *Προγραμματισμός Έργων Πληροφορικής - Αντικειμενοστραφείς Μεθοδολογίες*, τόμος Δ’, ΕΑΠ, Πάτρα, 2008
* Β. Βεσκούκης*, Στοιχεία Τεχνολογίας Λογισμικού*, ΣΕΑΒ, Θεσσαλονίκη, 2015
* P. Deitel and H. Deitel, *Java Προγραμματισμός*, 10η εκδ., Μ. Γκιούρδας, Αθήνα, 2015
* A. Denis κ.α., *Ανάλυση και Σχεδιασμός συστημάτων με την UML 2.0*, 3η έκδ., Κλειδάριθμος, Αθήνα, 2010

Ειδικό Διδακτικό Υλικό ΠΛΗ24:

* Π. Φιτσιλής, *Ανάπτυξη συστήματος λογισμικού με UML και Java*, ΕΑΠ, Πάτρα, 2007
* Α. Θάνος, *Πρότυπα σχεδίασης GUI*, ΕΑΠ, Πάτρα, 2013
* Α. Θάνος και Ν. Δρόσος*, Γραφικά Περιβάλλοντα με Swing*, ΕΑΠ, Πάτρα, 2012
* Π. Αλεφραγκής, *Γραφικά Ενδιάμεσα Χρήστη στην Java*, ΕΑΠ, Πάτρα, 2015
* N. Δρόσος και Α. Θάνος, *Εφαρμογές ΒΔ με χρήση του JPA: Java Persistence API*, ΕΑΠ, Πάτρα, 2012

Πηγές στο Internet:

* JFreeChart Library Tutorial, <https://zetcode.com/java/jfreechart/>
* JFreeChart 1.5.0 API Documentation, <https://www.jfree.org/jfreechart/api/javadoc/index.html>
* iText 5.5.9 API Documentation, <https://api.itextpdf.com/iText5/java/5.5.9/>
* iTextpdf GitHub, <https://github.com/itext/itextpdf>
* Java Swing API Documentation (Java™ Platform Standard Ed.8) <https://docs.oracle.com/javase/8/docs/api/index.html?javax/swing/package-summary.html>
* Java Swing Tutorial Video, <https://www.youtube.com/watch?v=Kmgo00avvEw>
* Using Transactions in JAVA, <https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html>
* Fibonacci Scale for Agile Estimation, <https://www.parabol.co/templates/agile-estimation/fibonacci-scale/>
* Json to POJO Converter, <https://json2csharp.com/json-to-pojo>
* GSON – Json Parser Tutorial, <http://tutorials.jenkov.com/java-json/gson-jsonparser.html>
* A Visual Guide to Layout Managers, <https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>
* Using Git in Apache NetBeans, <https://netbeans.apache.org/kb/docs/ide/git.html>