ΣΧΟΛΗ ΑΝΘΡΩΠΙΣΤΙΚΩΝ ΚΑΙ ΚΟΙΝΩΝΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΙΣΤΟΡΙΑΣ-ΑΡΧΑΙΟΛΟΓΙΑΣ

ΠΠΣ ΔΙΑΧΕΙΡΙΣΗΣ ΠΟΛΙΤΙΣΜΙΚΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΚΑΙ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ

Πτυχιακή Εργασία

Σχεδιασμός και υλοποίηση σε JAVA εφαρμογών για ανάδειξη διοικητικών σωμάτων μέσω ηλεκτρονικής διεξαγωγής μη χειραγωγήσιμων κληρώσεων

Νικολάος – Αλεξάνδρος Χαρίτος (Α.Μ.: 1001)

Επιβλέπουσα: Εύη Παπαϊωάννου, Επίκ. Καθηγήτρια

Συνεπιβλέπων: Δημήτρης Τσώλης, Επίκ. Καθηγητής

Μάρτιος 2021

Περίληψη

Η ανάγκη συγκρότησης διοικητικών σωμάτων και ομάδων εργασίας μέσω «τυχαίας» επιλογής ατόμων από ένα σύνολο υποψηφίων μελών είναι ένα ζήτημα που ανακύπτει συχνά τόσο σε φορείς δημόσιας διοίκησης όσο και σε εταιρείες και οργανισμούς. Ειδικότερα, δε, σε ιδρύματα τριτοβάθμιας εκπαίδευσης προκύπτει συχνά η ανάγκη σύστασης επιτροπών διοικητικού ή εκπαιδευτικού χαρακτήρα για σύσταση διοικητικών σωμάτων (π.χ., εκλεκτορικά σώματα, επιτροπές, κτλ) αλλά και ομάδων εργασίας μέσω «κλήρωσης». Η παραδοσιακή μέθοδος διεξαγωγής κληρώσεων με εκτυπωμένους «λαχνούς» (δηλαδή με εκτυπωμένα χαρτάκια σε κουτί) είναι χρονοβόρα και συχνά εγείρει ζητήματα ακεραιότητας και διαφάνειας. Η έλλειψη εμπιστοσύνης στις διαδικασίες επιλογής μπορεί να οδηγήσει σε σώματα και ομάδες χαμηλής ποιότητας και σε κακή συνεργασία, τα οποία με τη σειρά τους ενδέχεται να προκαλέσουν διάφορες μορφές αστάθειας στην εκάστοτε κοινότητα.

Στο πλαίσιο της παρούσας πτυχιακής εργασίας παρουσιάζουμε μια εφαρμογή για τη διεξαγωγή ηλεκτρονικών κληρώσεων μικρής κλίμακας για τη σύσταση διοικητικών σωμάτων, επιτροπών ή ομάδων εργασίας. Η αποδοτικότητα της διαδικασίας και η πραγματικά τυχαία και μη χειραγωγίσιμη φύση των αποτελεσμάτων προκύπτει από τη θεωρητικά τεκμηριωμένη αρτιότητα των αξιοποιούμενων μεθόδων υπολογισμού μεταθέσεων, δηλαδή του αλγορίθμου υπολογισμού μεταθέσεων των Durstenfeld-Knuth (Durstenfeld-Knuth shuffle) καθώς και του αλγορίθμου από τον οποίο προέκυψε και είναι γνωστός στη βιβλιογραφία ως μέθοδος υπολογισμού μεταθέσεων Fisher-Yates (Fisher-Yates shuffle).

Η εφαρμογή αυτή, που δεν απαιτεί προηγμένες δεξιότητες στη χρήση ηλεκτρονικών υπολογιστών ούτε εξειδικευμένο εξοπλισμό, θα μπορούσε να διευκολύνει σημαντικά, να απλουστεύσει και να βελτιώσει διοικητικές διαδικασίες παρέχοντας ταχύτητα και διαφάνεια κατά τη σύσταση διοικητικών ομάδων ή ομάδων εργασίας.

Abstract

The constitution of administrative bodies by drawing candidates out of a predefined pool is a frequent issue arising during several administrative procedures which require the formation of committees or working groups. Usually, pools of candidates as well as required administrative bodies are composed of a relatively small number of members. A basic requirement, common in all these cases, is the need for transparency in the selection process, which implies draws generating true random and unbiased results. A traditional implementation approach, often adopted in practice, consists in picking "tickets out of a hat" in public for preserving transparency. However, the transparency and the true randomness of this process are often questioned. In addition, manually conducted draws are time-consuming, thus making inefficient solutions.

Motivated by experiences in our professional environment (a higher education institute) as well as by similar experiences of colleagues in other sectors ranging from public administration agencies to small-scale companies, we designed and implemented an application for conducting small-scale draws, guaranteed to generate true random and unbiased results in the blink of an eye. Our application can be very easily installed on any machine (desktop computer or laptop featuring average specifications), requiring elementary computer skills. It offers a user-friendly graphical user interface (GUI), which is simple and light-weighted, and quickly computes results which are provably true random and unbiased due to the underlying efficient shuffling algorithms, i.e., Fisher-Yates shuffle and Durstenfeld-Knuth shuffle.

Πίνακας περιεχομένων

1 Εισαγωγή

Αυτή η εργασία έχει ως αφορμή από μια κατάσταση που συνήθως εμφανίζεται στο πλαίσιο αρκετών διοικητικών υποθέσεων σε ένα ίδρυμα τριτοβάθμιας εκπαίδευσης: μια μικρή ομάδα ατόμων πρέπει να κληρωθεί τυχαία από μια υπάρχουσα λίστα. Ορισμένα ενδεικτικά παραδείγματα τέτοιων ομάδων είναι εκλεκτορικά σώματα, επιτροπές για οικονομικά και εκπαιδευτικά θέματα, επιτροπές αξιολόγησης καθώς και ομάδες για την ανάθεση εργασιών. Σύμφωνα με το ισχύον νομικό πλαίσιο, τέτοιες επιτροπές πρέπει να συγκροτούνται με έναν πραγματικά τυχαίο τρόπο, ώστε να διασφαλίζεται η διαφάνεια και τα αμερόληπτα αποτελέσματα.

Παρά την ύπαρξη πολλών εξελιγμένων συστημάτων, τα οποία, ωστόσο, συνήθως απαιτούν συνδρομή επί πληρωμή ή αγορά που δεν είναι πάντα προσιτή, αυτό που γίνεται συνήθως είναι χειροκίνητες κληρώσεις, ειδικά όταν οι καθηγητές και το διοικητικό προσωπικό δεν είναι πραγματικά εξοικειωμένοι με τη χρήση σύνθετων πληροφοριακών συστημάτων. Αυτό σημαίνει πρακτικά ότι διατηρείται μια αριθμημένη λίστα και αυτή η λίστα περιέχει ονόματα υποψηφίων μελών, τα οποία χρησιμοποιούνται ως λαχνοί σε κάποια κλήρωση. Στη συνέχεια, η κλήρωση διεξάγεται χειροκίνητα, ζητώντας από άτομα να επιλέξουν τυχαία μέλη έως ότου επιτευχθεί ο απαιτούμενος αριθμός μελών. Μια τέτοια διαδικασία θα μπορούσε εύκολα να θεωρηθεί «συνηθισμένη» κάποτε. Ωστόσο, οι αλγόριθμοι και η σύγχρονη τεχνολογία μπορούν σίγουρα να προσφέρουν πιο αποτελεσματικές λύσεις.

Επιπλέον, η συνηθισμένη διαδικασία κλήρωσης αμφισβητείται συνήθως ως προς τη διαφάνεια και την ακεραιότητα: είναι πλήρης η λίστα υποψηφίων μελών; Περιέχει η λίστα όλους τους υποψηφίους; Κάθε υποψήφιος συμπεριλαμβάνεται στην κλήρωση μόνο μία φορά; Η επιστήμη των υπολογιστών μπορεί να προτείνει απλές και αποτελεσματικές λύσεις, χωρίς κόστος, χωρίς να απαιτείται ειδικός εξοπλισμός ούτε προηγμένες δεξιότητες και γνώσεις στη χρήση υπολογιστών, υπολογίζοντας πραγματικά τυχαία αποτελέσματα.

Η λύση που προτείνεται βασίζεται στον "Algorithm P (Shuffling)", επίσης γνωστό ως Durstenfeld-Knuth shuffle [1]. Ο αλγόριθμος προτάθηκε για πρώτη φορά τη δεκαετία του '60 βελτιώνοντας προηγούμενο αλγόριθμο γνωστό ως Fisher-Yates shuffle. Η εφαρμογή που προτείνεται αναπτύχθηκε σε java και μπορεί εύκολα να χρησιμοποιηθεί από μη ειδικούς χωρίς να απαιτεί προηγμένες δεξιότητες και γνώσεις χρήσης υπολογιστών. Η εφαρμογή χρειάζεται μια λίστα υποψηφίων και τον αριθμό των απαιτούμενων μελών που πρέπει να κληρωθούν. Η λίστα παρέχεται είτε χειροκίνητα είτε μέσω της μεταφόρτωσης ενός αρχείου. Όταν υπάρχει ανάγκη να γίνει κλήρωση με περιορισμένο αριθμό ατόμων (το πολύ 50), διατηρώντας τη διαφάνεια, όπως για παράδειγμα για τον σχηματισμό μιας μικρής επιτροπής κατά τη διάρκεια μιας συνέλευσης, χρησιμοποιείται η χειροκίνητη μέθοδος για τη δημιουργία της λίστας των υποψηφίων μελών. Η λίστα που δημιουργείται μετά τη κλήρωση περιέχει το πλήθος των ατόμων που ζητήθηκαν, η οποία δημιουργήθηκε από την τυχαία μετάθεση της αρχικής λίστας και υπολογίζεται από τον "Algorithm P (Shuffling)". Η τελική λίστα που δημιουργείται μπορεί είτε να εμφανιστεί στην οθόνη είτε να αποθηκευτεί σε ένα αρχείο..

Η διπλωματική εργασία είναι δομημένη ως εξής. Στην Ενότητα 2, παρουσιάζουμε το θεωρητικό πλαίσιο, περιγράφοντας λεπτομερώς τον αλγόριθμο Fisher – Yates shuffle και τον Durstenfeld-Knuth shuffle ή αλλιώς τον "Algorithm P (Shuffling)" [1, 3]. Στην Ενότητα 3, παρουσιάζεται η εφαρμογή και στην Ενότητα 4 είναι ο επίλογος.

2 Θεωρητικό πλαίσιο

Η ιδέα που εφαρμόσαμε είναι η ακόλουθη. Μόλις λάβουμε τον πλήρη κατάλογο των υποψηφίων, τον μεταθέτουμε τυχαία και στη συνέχεια εξάγουμε ένα πρόθεμα μεγέθους ίσο με τον απαιτούμενο αριθμό μελών της επιτροπής. Η είσοδος, δηλαδή το μητρώο και ο αριθμός των απαιτούμενων μελών της επιτροπής, μπορούν να παρέχονται είτε χειροκίνητα - πληκτρολογώντας τα ονόματα των υποψηφίων - είτε μέσω ενός απλού αρχείου κειμένου. Το αποτέλεσμα, δηλαδή η λίστα των τυχαία επιλεγμένων μελών εμφανίζεται στην οθόνη καθώς και αν ζητηθεί, γίνεται η εμφάνισή του και σε ένα αρχείο κειμένου. Το κρίσιμο σημείο της παραπάνω διαδικασίας είναι να διασφαλιστεί ότι η παραλλαγή της λίστας εισόδου είναι πραγματικά τυχαία και αμερόληπτη. Για το σκοπό αυτό, έχουμε εκμεταλλευτεί δύο απλούς αλλά αποτελεσματικούς αλγόριθμους από τη βιβλιογραφία: "Fisher-Yates shuffle" [2, 3] και "Durstenfeld-Knuth shuffle" [1, 3]. Όπως υποδηλώνουν τα ονόματά τους, και οι δύο αλγόριθμοι εκτελούν την διαδικασία της τυχαίας Μέθοδος υπολογισμού μεταθέσεων Fisher-Yates

Ο αλγόριθμος "Fisher-Yates shuffle" επιλέγει τυχαία στοιχεία που υπάρχουν στη λίστα εισόδου και τα τοποθετεί σε μία νέα λίστα, τη λίστα εξόδου. Ο αλγόριθμος παρουσιάστηκε για πρώτη φορά το 1938 από τους Ronald Fisher και Frank Yates στο βιβλίο τους " Statistical tables for biological, agricultural and medical research " [2]. Αυτό που κάνει ο αλγόριθμος είναι να μετατρέπει τυχαία N στοιχεία μιας λίστας εισόδου [s1, s2,… sq,…, sN], αλλάζοντας κάθε ένα από αυτά με ένα τυχαίο στοιχείο από q σε N. Ο αλγόριθμος υπολογίζει τις μεταθέσεις αμερόληπτα με την έννοια ότι όλα οι μεταθέσεις είναι εξίσου πιθανές. Ο "Fisher-Yates shuffle", αρχικά προτάθηκε ως μέθοδος για τους ερευνητές να αναδιατάξουν μη αυτόματα (8) οντότητες - στοιχεία σε τυχαία σειρά, έχει μια πολυπλοκότητα O (N^2).

Σύμφωνα με την αρχική του περιγραφή, ο αλγόριθμος Fisher-Yates ανακατεύει τυχαία τα στοιχεία μιας δεδομένης λίστας εισόδου, αφαιρώντας τα από τη λίστα εισόδου και εισάγοντάς τα σε μια δεύτερη λίστα εξόδου. Πιο συγκεκριμένα,

ξεκινάμε με μια λίστα εισόδου N στοιχείων. Δηλώνουμε με q τον αριθμό των στοιχείων που δεν έχουν αφαιρεθεί ακόμη. Κατά τη διάρκεια κάθε βήματος του αλγορίθμου, επιλέγουμε τυχαία έναν αριθμό από [1, q] και μετακινούμε το εκάστοτε στοιχείο q της λίστας εισόδου στο τέλος της λίστας εξόδου. Ο αλγόριθμος τερματίζεται μόλις όλα τα στοιχεία από τη λίστα εισόδου μετακινηθούν στη λίστα εξόδου. Σε αυτό το σημείο, η λίστα εξόδου περιέχει μια τυχαία παραλλαγή των στοιχείων της δεδομένης λίστας εισόδου. Η διαδικασία που αναλύθηκε παραπάνω έχει πολυωνυμική πολυπλοκότητα O (n^2), λόγω της επανάληψης που απαιτείται σε κάθε βήμα για τον προσδιορισμό του εκάστοτε q στοιχείου που δεν έχει αφαιρεθεί από την λίστα εισόδου.

Εφαρμόζοντας αυτόν τον αλγόριθμο στο αρχικό παράδειγμα των 8 στοιχείων του Fisher-Yates [2], υπολογίζεται μια τυχαία μετάθεση των 8 στοιχείων όπως απεικονίζεται στην Εικόνα 2.1.1.

Αρχικά

Λίστα εισόδου: 1 2 3 4 5 6 7 8

q σε [1,8] επιλέχθηκε τυχαία η θέση 1

Λίστα εξόδου: 1

Μετά το βήμα 1

Λίστα εισόδου: 1 2 3 4 5 6 7 8

q σε [1,7] επιλέχθηκε τυχαία η θέση 5

Λίστα εξόδου: 1 6

Μετά το βήμα 4

Λίστα εισόδου: 1 2 3 4 5 6 7 8

q σε [1,4] επιλέχθηκε τυχαία η θέση 2

Λίστα εξόδου: 1 6 3 4 5

Μετά το βήμα 5

Λίστα εισόδου: 1 2 3 4 5 6 7 8

q σε [1,3] επιλέχθηκε τυχαία η θέση 3

Λίστα εξόδου: 1 6 3 4 5 8

Μετά το βήμα 2

Λίστα εισόδου: 1 2 3 4 5 6 7 8

q σε [1,6] επιλέχθηκε τυχαία η θέση 2

Λίστα εξόδου: 1 6 3

Μετά το βήμα 6

Λίστα εισόδου: 1 2 3 4 5 6 7 8

q σε [1,2] επιλέχθηκε τυχαία η θέση 2

Λίστα εξόδου: 1 6 3 4 5 8 7

Μετά το βήμα 3

Λίστα εισόδου: 1 2 3 4 5 6 7 8

q σε [1,5] επιλέχθηκε τυχαία η θέση 2

Λίστα εξόδου: 1 6 3 4

Τέλος

Λίστα εισόδου: 1 2 3 4 5 6 7 8

Λίστα εξόδου: 1 6 3 4 5 8 7 2

Εικόνα 2.1.1: Βήμα προς βήμα η υλοποίηση της τυχαίας μετάθεσης που προτάθηκε αρχικά από τους Fisher και Yates.

Η μετάθεση που προκύπτει είναι πραγματικά τυχαία εφόσον η επιλογή του q στοιχείου σε κάθε βήμα πραγματοποιείται με έναν πραγματικά τυχαίο τρόπο.

Μαζί με την αρχική περιγραφή της τεχνικής τους, οι συγγραφείς ανέφεραν το ζήτημα που προκύπτει σχετικά με το πώς μπορούν να ληφθούν οι απαιτούμενοι τυχαίοι αριθμοί αποφεύγοντας τη μεροληψία· παρείχαν επίσης εναλλακτικές προσεγγίσεις για να καταστήσουν τη διαδικασία επιλογής πιο αποτελεσματική, διατηρώντας παράλληλα την πραγματική τυχαιότητα [2].

2.1 Αλγόριθμος υπολογισμού μεταθέσεων Durstenfeld-Knuth

Ο αλγόριθμος "Durstenfeld-Knuth shuffle" είναι μια βελτιωμένη παραλλαγή του αλγορίθμου Fisher-Yates που προτάθηκε από τον Richard Durstenfeld το 1964 [1] και αργότερα διαδόθηκε ως "Algorithm P (Shuffling)" ή "Knuth shuffle" από τον Donald E. Knuth στο βιβλίο του "The Art of Computer Programming" [3]. Ο αλγόριθμος που προτάθηκε από τον Durstenfeld προχωρά με τρόπο παρόμοιο με αυτόν που πρότειναν αρχικά οι Fisher και Yates, ωστόσο, με μια μικρή αλλά σημαντική διαφορά στην εφαρμογή του. Πιο συγκεκριμένα, δεδομένου ότι έχει αρχικοποιηθεί μία λίστα, ο αλγόριθμος υπολογισμού τυχαίων μεταθέσεων που προτάθηκε από τον Richard Durstenfeld το 1964, μεταθέτει τα στοιχεία της λίστας στην ίδια λίστα, αντί να παράγει νέα λίστα. Αυτό το χαρακτηριστικό του αλγορίθμου μπορεί να έχει σημαντικό αντίκτυπο όταν υπάρχουν μεγάλες λίστες εισόδου.

Ξεκινώντας από το τελευταίο στοιχείο της αρχικής λίστας N στοιχείων, γίνεται μετάθεση αυτού του στοιχείου με ένα τυχαία επιλεγμένο στοιχείο (συμπεριλαμβανομένου και του τελευταίου στοιχείου) από τη λίστα. Στη συνέχεια, επαναλαμβάνεται αυτή η διαδικασία για N-1 στοιχεία της λίστας, δηλαδή, εξαιρουμένου του στοιχείου που έχει ήδη μετατεθεί. Η διαδικασία συνεχίζεται με τον ίδιο τρόπο μέχρι να φτάσει στο πρώτο στοιχείο της λίστας.

Algorithm P για τυχαία μετάθεση N στοιχείων μίας λίστας s [s_0,..., s_{N-1}]

για κάθε q από N - 1 έως 1

r = τυχαίος ακέραιος σε [0,q]

μετάθεση a[q] με a[r]

Με αυτόν τον τρόπο, υποθέτοντας ότι η επιλογή τυχαίων αριθμών απαιτεί σταθερό χρόνο, ο αλγόριθμος απαιτεί χρόνο ανάλογο με τον αριθμό των στοιχείων που

μετατίθενται και χωρίς επιπλέον χώρο αποθήκευσης, αποκτώντας έτσι βελτιωμένη πολυπλοκότητα της τάξης του O(N) σε σύγκριση με το $O(N^2)$ που απαιτείται από τον αλγόριθμο Fisher-Yates.

Εφαρμόζοντας αυτόν τον βελτιωμένο αλγόριθμο στο αρχικό παράδειγμα 8 στοιχείων του Fisher-Yates [2], θα υπολογιζόταν μια μετάθεση των 8 στοιχείων όπως απεικονίζεται στην Εικόνα 2.1.2.

Φαίνεται ότι κάθε στοιχείο μετατίθεται εξίσου πιθανά σε μία θέση της λίστας (με πιθανότητα 1 / N). Έτσι, οι μεταθέσεις παραμένουν πραγματικά τυχαίες και αμερόληπτες.

Πιο συγκεκριμένα, η απόδειξη της ορθότητας του αλγορίθμου μεταθέσεων βασίζεται στο γεγονός ότι όλες οι μεταθέσεις των N! στοιχείων της λίστας είναι εξίσου πιθανές. Ας υποθέσουμε μια λίστα στοιχείων L [1,…, N]. Στην ιδανική περίπτωση, ένας αμερόληπτος αλγόριθμος θα πρέπει να μεταθέτει τυχαία όλα τα στοιχεία του L έτσι ώστε μετά την διαδικασία η πιθανότητα επιστροφής κάθε μετάθεσης του L να παραμείνει η ίδια (δηλ., 1 / N!) για όλες τις πιθανές μεταθέσεις (που είναι συνολικά N!). Επιπλέον, για οποιοδήποτε αυθαίρετο στοιχείο και οποιαδήποτε θέση q στο L, όπου $1 \leq q \leq N$, η πιθανότητα το στοιχείο να μετακινηθεί στη θέση L [q] ισούται με 1 / N. Για να δούμε γιατί ισχύει αυτό, παρατηρούμε πρώτα ότι η πιθανότητα οποιοδήποτε αυθαίρετο στοιχείο να μετακινηθεί στην τελευταία θέση είναι ουσιαστικά ίση με την πιθανότητα να επιλεχθεί το στοιχείο αυτό να μετακινηθεί κατά το πρώτο βήμα, δηλ. 1 / N. Για οποιοδήποτε αυθαίρετο στοιχείο, η πιθανότητα μετακίνησης στη δεύτερη τελευταία θέση παραμένει 1 / N, καθώς αυτό το αυθαίρετο στοιχείο δεν πρέπει να επιλέγεται για την τελευταία θέση (που συμβαίνει με πιθανότητα N-1 / N) και πρέπει να επιλεγεί για μετακίνηση κατά το δεύτερο βήμα (το οποίο συμβαίνει με πιθανότητα 1 / N-1). Προχωρώντας με αυτόν τον τρόπο, καταλήγουμε στο συμπέρασμα ότι η πιθανότητα μετακίνησης οποιουδήποτε αυθαίρετου στοιχείου σε οποιαδήποτε αυθαίρετη θέση ισούται με 1 / N, αποδεικνύοντας έτσι την ορθότητα του αλγορίθμου αυτού.

Αρχικά

Λίστα εισόδου: 1 2 3 4 5 6 7 8

q σε [1,8] επιλέχθηκε τυχαία το 1

Μετά το βήμα 1

Λίστα εισόδου: 8 2 3 4 5 6 7 1

q σε [1,7] επιλέχθηκε τυχαία το 5

Μετά το βήμα 2

Λίστα εισόδου: 8 2 3 4 7 6 5 1

q σε [1,6] επιλέχθηκε τυχαία το 2

Μετά το βήμα 3

Λίστα εισόδου: 8 6 3 4 7 2 5 1

q σε [1,5] επιλέχθηκε τυχαία το 2

Μετά το βήμα 4

Λίστα εισόδου: 8 7 3 4 6 2 5 1

q σε [1,4] επιλέχθηκε τυχαία το 2

Μετά το βήμα 5

Λίστα εισόδου: 8 4 3 7 6 2 5 1

q σε [1,3] επιλέχθηκε τυχαία το 3

Μετά το βήμα 6

Λίστα εισόδου: 8 4 3 7 6 2 5 1

q σε [1,2] επιλέχθηκε τυχαία το 2

Τέλος

Παραλλαγμένη λίστα εισαγωγής: 8 4 3 7 6 2 5 1

Εικόνα *2.1.2: Βήμα προς βήμα η εκτέλεση του βελτιωμένου αλγορίθμου Durstenfeld-Knuth.*

3 Η εφαρμογή

Για την εφαρμογή χρησιμοποιήθηκε ο αρχικός αλγόριθμος Fisher-Yates [2, 3], καθώς και η βελτιωμένη έκδοση του, ο αλγόριθμος Durstenfeld-Knuth shuffle ή Algorithm P (Shuffling) [1, 3].

Η εφαρμογή λαμβάνει ως εισαγωγή μια λίστα υποψηφίων και τον αριθμό των απαιτούμενων επιλογών. Η λίστα παρέχεται είτε χειροκίνητα είτε με τη μεταφόρτωση ενός αρχείου. Η έξοδος αποτελείται από μια νέα λίστα που περιέχει τον αριθμό και τα στοιχεία των ατόμων που επιλέχθηκαν τυχαία και αποτελεσματικά μέσω των αλγορίθμων Fisher-Yates ή Durstenfeld-Knuth shuffle.

Η εφαρμογή αποτελείται από έξι εκτελέσιμα αρχεία, δηλαδή τα FYS-gr.exe, DKS-gr.exe, FYS-mi-gr.exe, DKS-mi-gr.exe, fysmi-gr.exe και dksmi-gr.exe. Για την περίπτωση που η είσοδος παρέχεται με μη αυτόματο τρόπο, δηλαδή απλά πληκτρολογώντας τα στοιχεία της λίστας, υπάρχουν τα αρχεία FYS-mi-gr.exe και DKS-mi-gr.exe αντίστοιχα. Στην περίπτωση της εισόδου με χειροκίνητο τρόπο υπάρχουν τα αρχεία που εκτελούνται μέσω της γραμμής εντολών fysmi-gr.exe και dksmi-gr.exe αντίστοιχα. Όλα τα αρχεία της εφαρμογής είναι διαθέσιμα στο σύνδεσμο https://bit.ly/35bbrw6. Προκειμένου να χρησιμοποιηθεί η εφαρμογή, το συμπιεσμένο αρχείο που υπάρχει στον παραπάνω σύνδεσμο θα πρέπει να αποσυμπιεσθεί τοπικά σε κάποιον υπολογιστή και να εκτελεστούν τα αρχεία .exe. Δεν απαιτείται εγκατάσταση της Java ή οποιουδήποτε άλλου συμπληρωματικού λογισμικού.

Μετά την εγκατάσταση των εκτελέσιμων αρχείων σε υπολογιστή με Windows, εμφανίζονται αντίστοιχες καταχωρήσεις στην ενότητα "Όλα τα προγράμματα". Οι συντομεύσεις προστίθενται στη Επιφάνεια εργασίας αν ο χρήστης επέλεξε να δημιουργήσει τις συντομεύσεις όταν του ζητήθηκε κατά τη διαδικασία εγκατάστασης.

3.1 Πλαίσιο υλοποίησης

Προτείνεται μια απλή, εύχρηστη εφαρμογή Java που αναπτύχθηκε χρησιμοποιώντας το Apache Net Beans IDE 11.3 [4] και το Java Swing [5], μια εργαλειοθήκη γραφικών GUI (Γραφικό περιβάλλον διεπαφής χρήστη) για Java.

Το NetBeans εκτελείται σε Windows, macOS, Linux και Solaris. Εκτός από την ανάπτυξη σε γλώσσα Java, έχει επεκτάσεις και για άλλες γλώσσες όπως PHP, C, C ++, HTML5 και JavaScript.

Το Swing είναι ένα GUI framework για Java. Είναι μέρος του Java Foundation Classes (JFC) της Oracle - ένα API (Διεπαφή Προγραμματισμού Εφαρμογών) για την παροχή γραφικής διεπαφής χρήστη (GUI) για προγράμματα που δημιουργήθηκαν μέσω της γλώσσας προγραμματισμού Java.

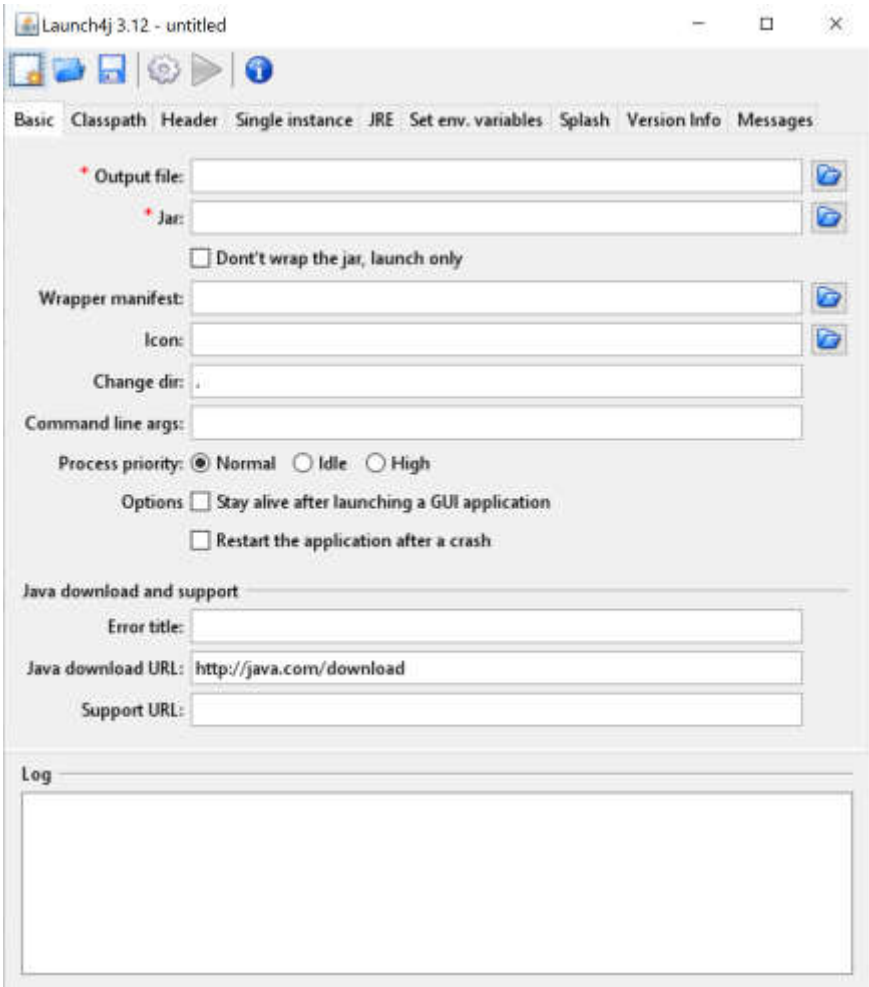
Το Swing αναπτύχθηκε για να παρέχει ένα πιο εξελιγμένο σύνολο στοιχείων GUI από το προηγούμενο Abstract Window Toolkit (AWT). Εκτός από τα γνωστά στοιχεία, όπως κουμπιά, πλαίσια ελέγχου και ετικέτες, το Swing παρέχει πολλά προηγμένα στοιχεία όπως πάνελ με καρτέλες, παράθυρα κύλισης, πίνακες και λίστες.

Χρησιμοποιήθηκε το Launch4j 3.12 [6] για τη δημιουργία εκτελέσιμων αρχείων από τα αρχικά αρχεία .jar. Χρησιμοποιήθηκε επίσης η έκδοση 6.0.4 του Inno Setup [7] για τη δημιουργία προγραμμάτων εγκατάστασης. Για την εκτέλεσή τους χρησιμοποιήθηκε ένας φορητός υπολογιστής DELL Inspiron 7559 με Windows 10 Pro 64-bit σε Intel (R) Core (TM) i7-6700HQ CPU @ 2.60GHz (8CPUs) με 8GB RAM. Για τη δοκιμή της εφαρμογής χρησιμοποιήθηκε ένας φορητός υπολογιστής Lenovo 80MJ με Windows 10 Pro 64-bit σε Intel (R) Celeron (R) CPU N2840 @ 2,16 GHz (2CPUs) με 4GB RAM. Επιπλέον, χρησιμοποιήθηκε το https://repl.it/, ένα διαδικτυακό IDE που επιτρέπει την δημιουργία εφαρμογών στο διαδίκτυο

χρησιμοποιώντας μόνο ένα πρόγραμμα περιήγησης χωρίς να απαιτείται τοπική εγκατάσταση λογισμικού. Η αξιοποίηση προηγμένων δυνατοτήτων της πλατφόρμας είναι εμπορική και απαιτεί συνδρομή επί πληρωμή. Ωστόσο, η δωρεάν βασική έκδοση της πλατφόρμας διευκόλυνε τη μακρινή συνεργασία (που προκλήθηκε λόγω της πανδημίας).

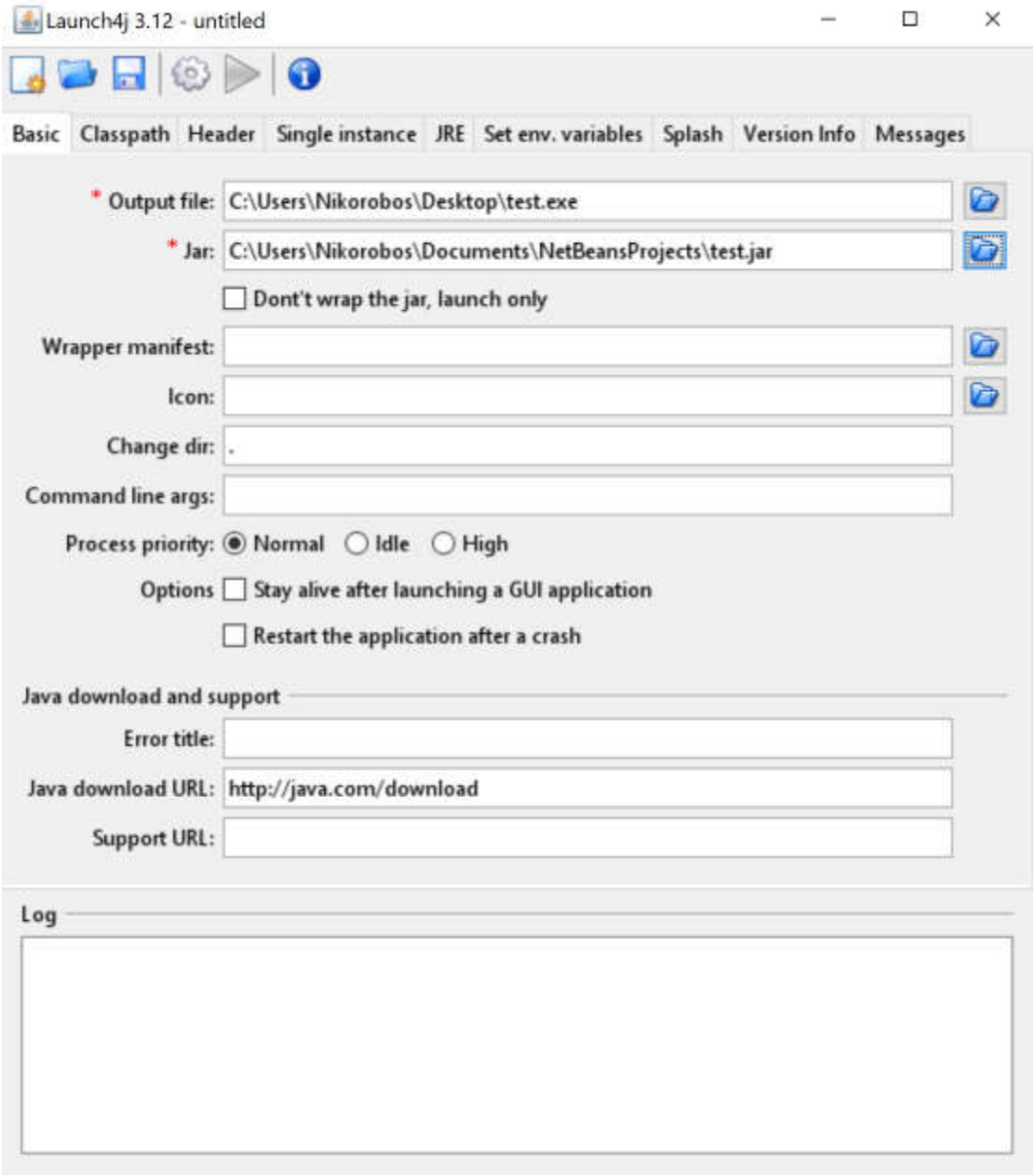
3.1.1 Δημιουργία εκτελέσιμων αρχείων (.exe)

Για τη δημιουργία εκτελέσιμων αρχείων χρησιμοποιήθηκε η εφαρμογή Launch4j 3.12. Στο πεδίο “Output file” βάζουμε το όνομα που θέλουμε να ονομάσουμε το εκτελέσιμο αρχείο. Στο “Jar” επιλέγουμε το αρχείο με κατάληξη “.jar” που έχει δημιουργηθεί αυτόματα από το Apache Netbeans που χρησιμοποιήσαμε για τη δημιουργία της εφαρμογής. Στο γραφικό περιβάλλον του Launch4j υπάρχουν κι άλλοι παράμετροι που μπορούμε να ορίσουμε αλλά γι’ αυτή την εφαρμογή δεν χρησιμοποιήθηκαν (βλ. Εικόνα 3.1.1.1).



Εικόνα 3.1.1.1: Γραφικό περιβάλλον της εφαρμογής Launch4j.

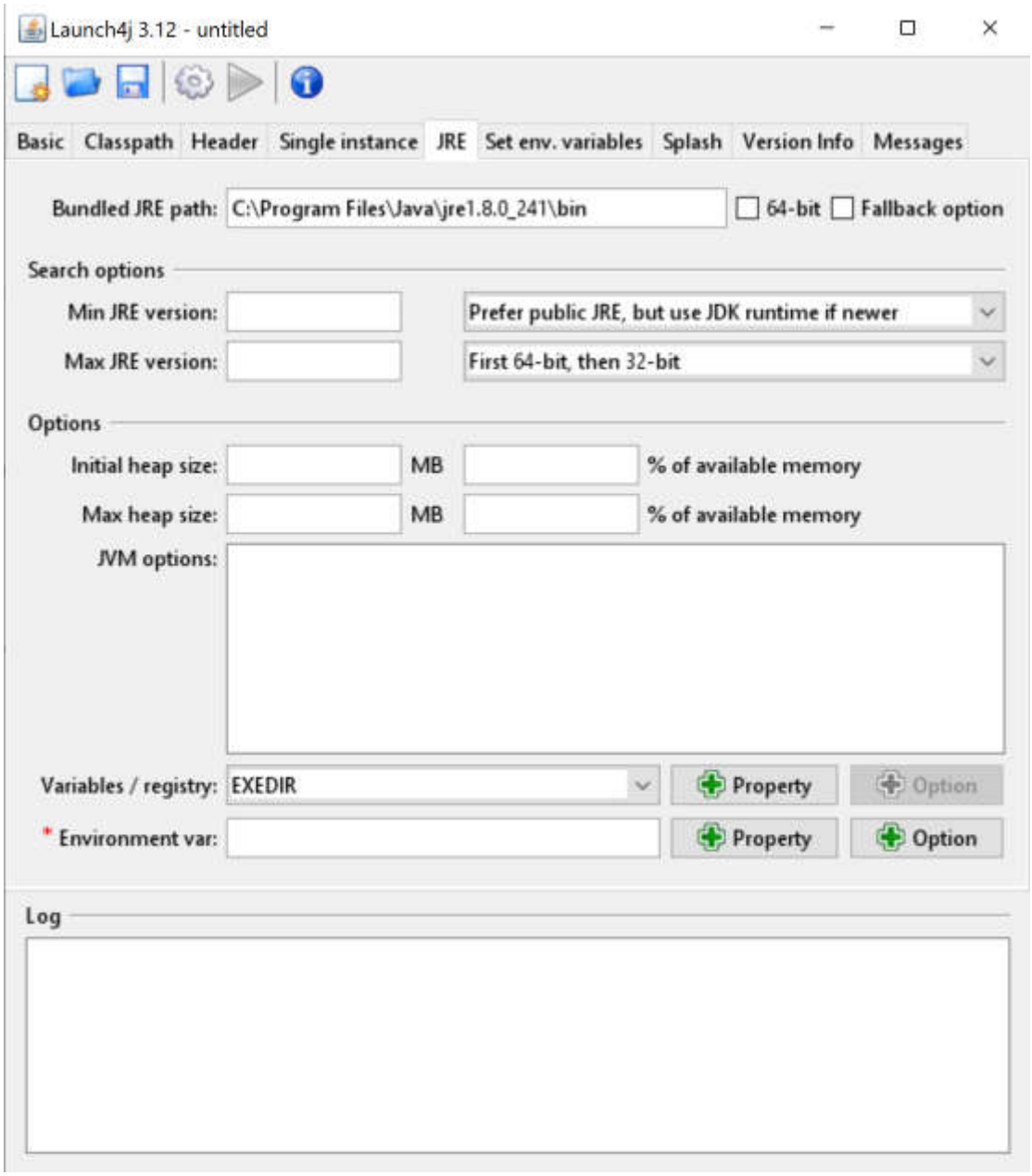
Με το πάτημα του φακέλου δεξιά από το ”Output file” επιλέγουμε τον προορισμό που θέλουμε να αποθηκευτεί το αρχείο. Με το αντίστοιχο για την επιλογή “Jar” επιλέγουμε τον προορισμό που βρίσκεται το “.jar” αρχείο που δημιουργήθηκε (βλ. Εικόνα 3.1.1.2).



Εικόνα 3.1.1.2: Εύρεση αρχείου ".jar" και επιλογή φακέλου για την αποθήκευση του εκτελέσιμου αρχείου που δημιουργήσαμε.

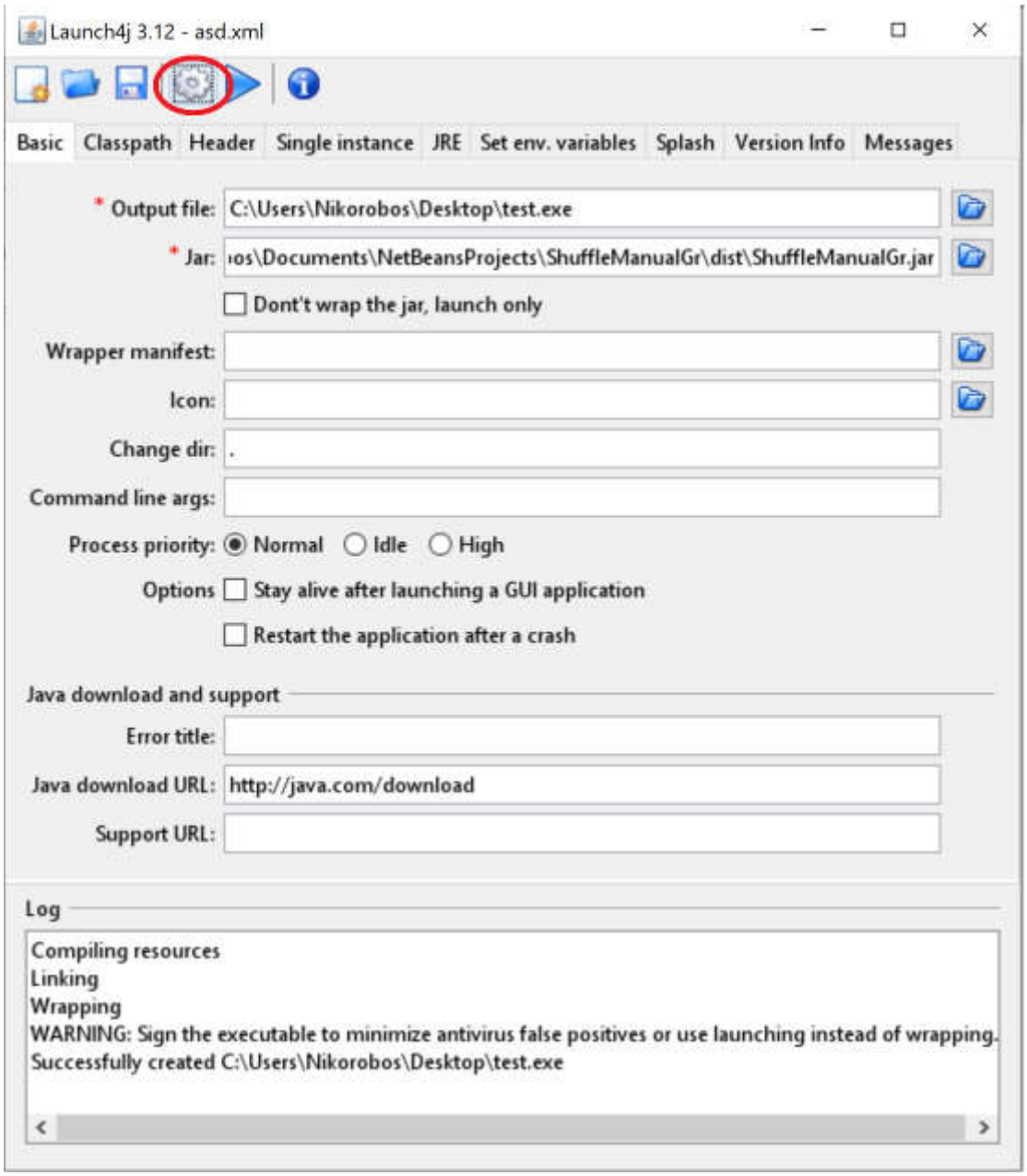
Στη συνέχεια στο navigation bar επιλέγουμε το "JRE" το οποίο μας εμφανίζει περισσότερες επιλογές. Εδώ έχουμε τη δυνατότητα να θέσουμε κάποιον περιορισμό για την έκδοση του JRE (Java Runtime Environment) που χρειάζεται να υπάρχει εγκατεστημένο για να εκτελεστεί η εφαρμογή. Στην δική μας περίπτωση επιλέγουμε να εισάγουμε το JRE που υπάρχει εγκατεστημένο ήδη έτσι ώστε ο

χρήστης να μη χρειάζεται να κατεβάσει και να εγκαταστήσει από την αρχή το JRE. Στο “Bundled JRE path” επιλέγουμε τον προορισμό του JRE (βλ. Εικόνα 3.1.1.3).



Εικόνα 3.1.1.3: Επιλογή προορισμού JRE (Java Runtime Environment).

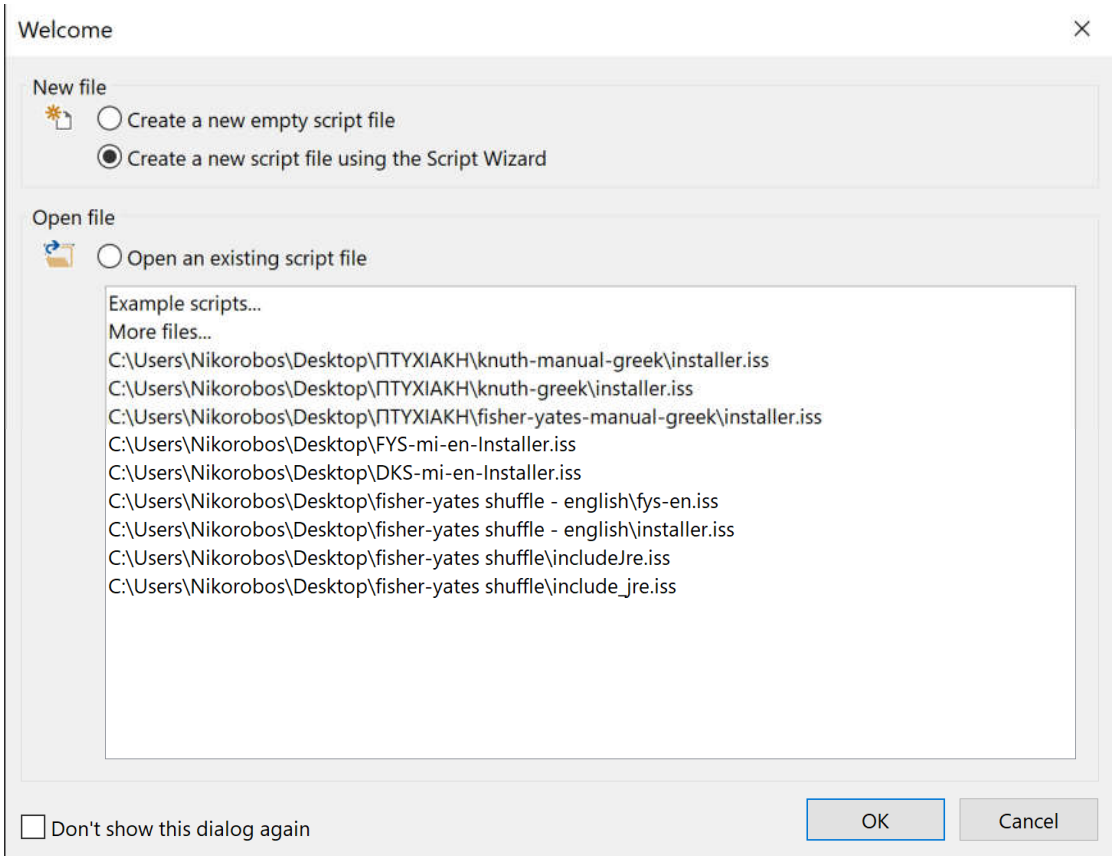
Τέλος για να ολοκληρωθεί η διαδικασία για τη δημιουργία του εκτελέσιμου αρχείου, χρειάζεται να πατήσουμε το κουμπί που μοιάζει με γρανάζι (βλ. Εικόνα 3.1.1.4).



Εικόνα 3.1.1.4: Δημιουργία εκτελέσιμου αρχείου.

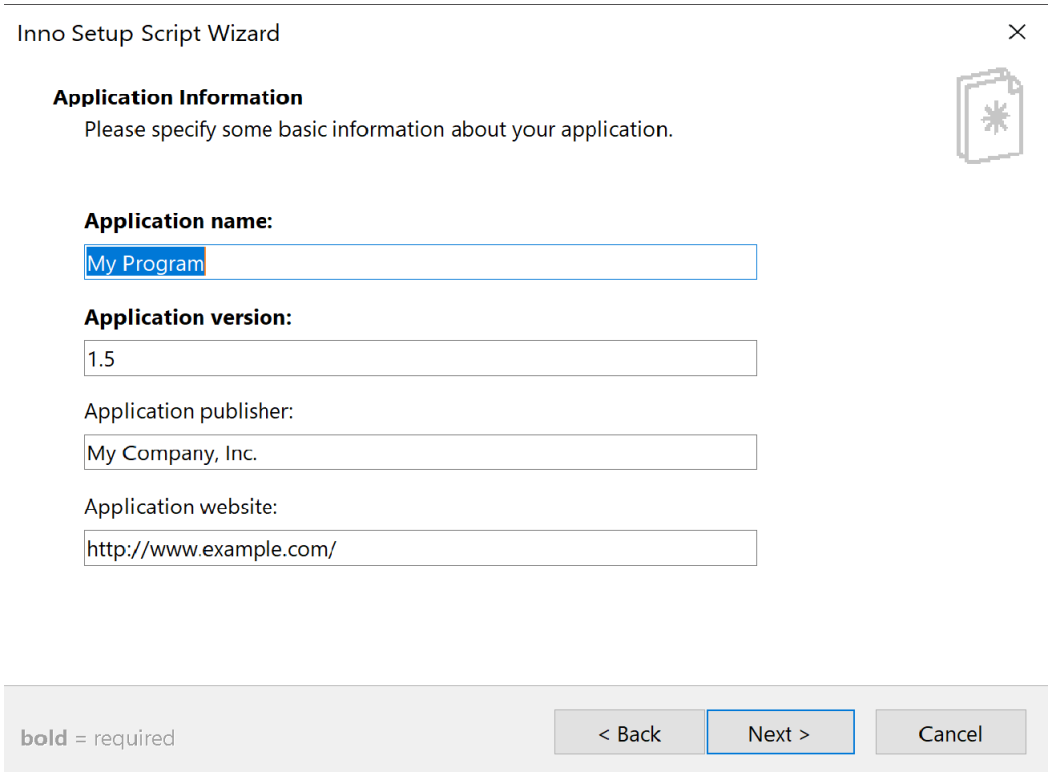
3.1.2 Δημιουργία αρχείων εγκατάστασης

Για να δημιουργήσουμε αρχεία εγκατάστασης, έτσι ώστε ο χρήστης να μπορεί ο χρήστης να τα εγκαταστήσει τοπικά στον υπολογιστή του, χρησιμοποιήθηκε το πρόγραμμα Inno Setup Compiler 6.0.4. Η εφαρμογή αυτή επιτρέπει τη δημιουργία αυτών των αρχείων με τη προϋπόθεση ότι υπάρχουν ήδη τα εκτελέσιμα αρχεία (.exe). Αρχικά επιλέγουμε από το “New file” την επιλογή “Create a new script file using the Script Wizard” (βλ. Εικόνα 3.1.2.1)



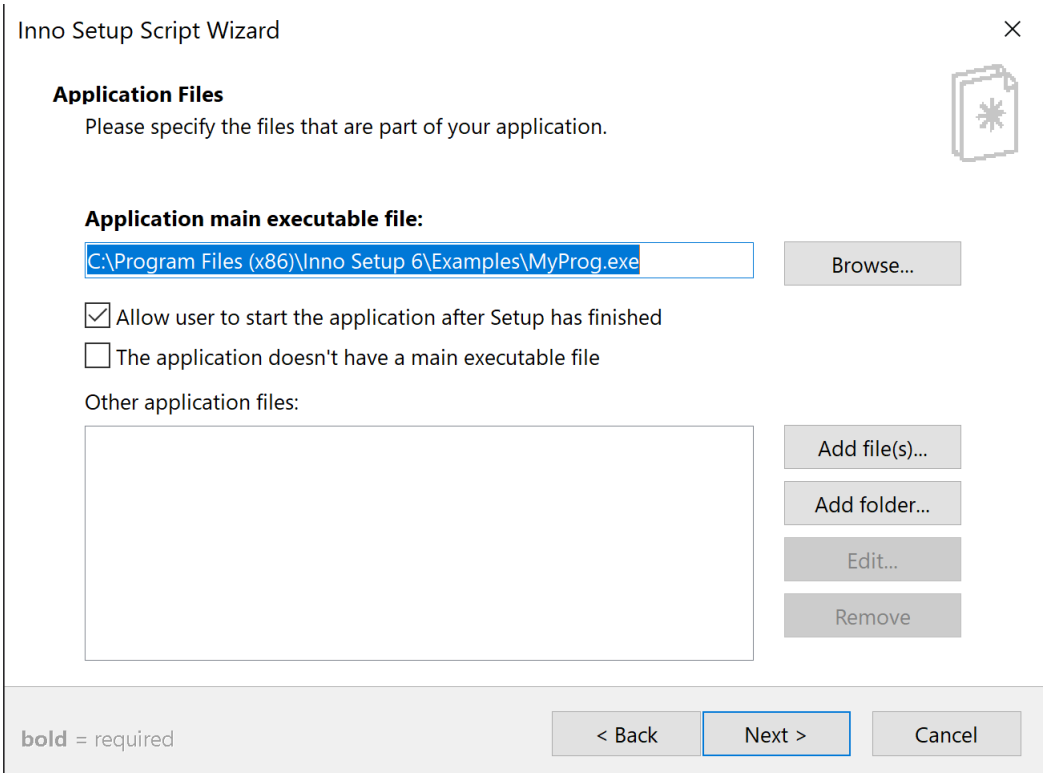
Εικόνα 3.1.2.1: Δημιουργία νέου αρχείου εγκατάστασης.

Στη συνέχεια πληκτρολογούμε το όνομα της εφαρμογής και τον αριθμό της έκδοσης τα οποία είναι υποχρεωτικά πεδία καθώς και το όνομα του δημιουργού της εφαρμογής και την ιστοσελίδα του που δεν είναι υποχρεωτικά πεδία (βλ. Εικόνα 3.1.2.2)



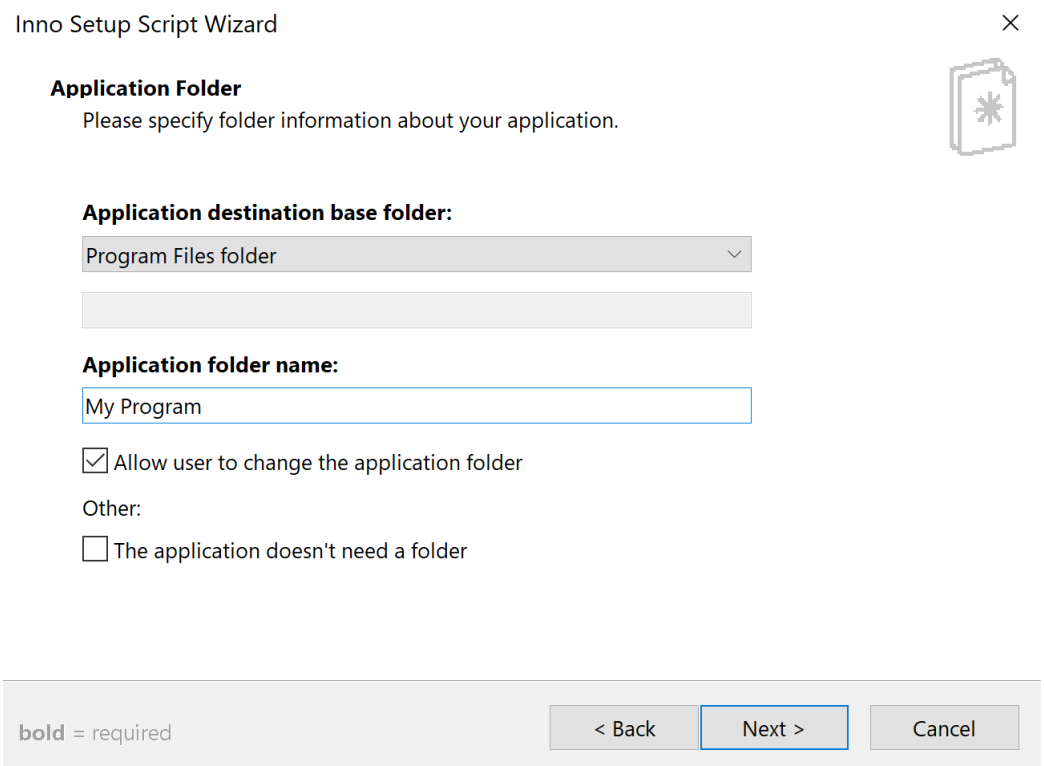
Εικόνα 3.1.2.2: Εισαγωγή στοιχείων για την εφαρμογή.

Επιλέγουμε τον προορισμό που είναι αποθηκευμένο το εκτελέσιμο αρχείο το οποίο δημιουργήσαμε (Εικόνα 3.1.2.3).



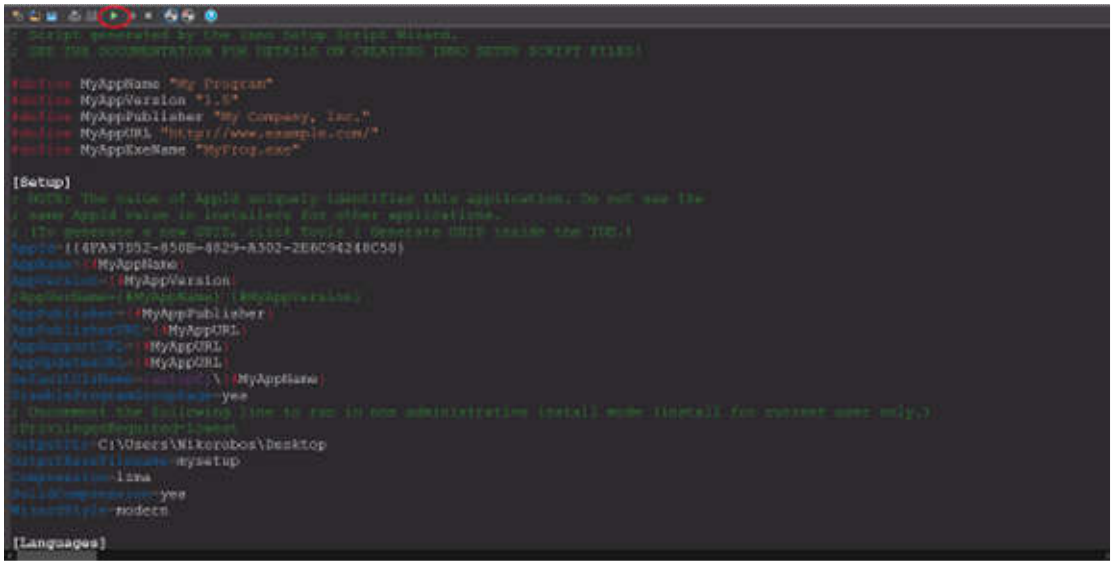
Εικόνα 3.1.2.3: Εύρεση εκτελέσιμου αρχείου.

Επιλέγουμε το όνομα του αρχείου εγκατάστασης, τον προορισμό που θα αποθηκευτεί αυτό το αρχείο καθώς και αν θα έχει κάποιο εικονίδιο και κάποιον κωδικό και πατάμε “Next” για να μας μεταφέρει στο παράθυρο της εφαρμογής όπου υπάρχει ο κώδικας που φτιάχτηκε βάσει των επιλογών μας (βλ. Εικόνα 3.1.2.4).



Εικόνα 3.1.2.4: Επιλογή προορισμού και ονόματος αρχείου εγκατάστασης.

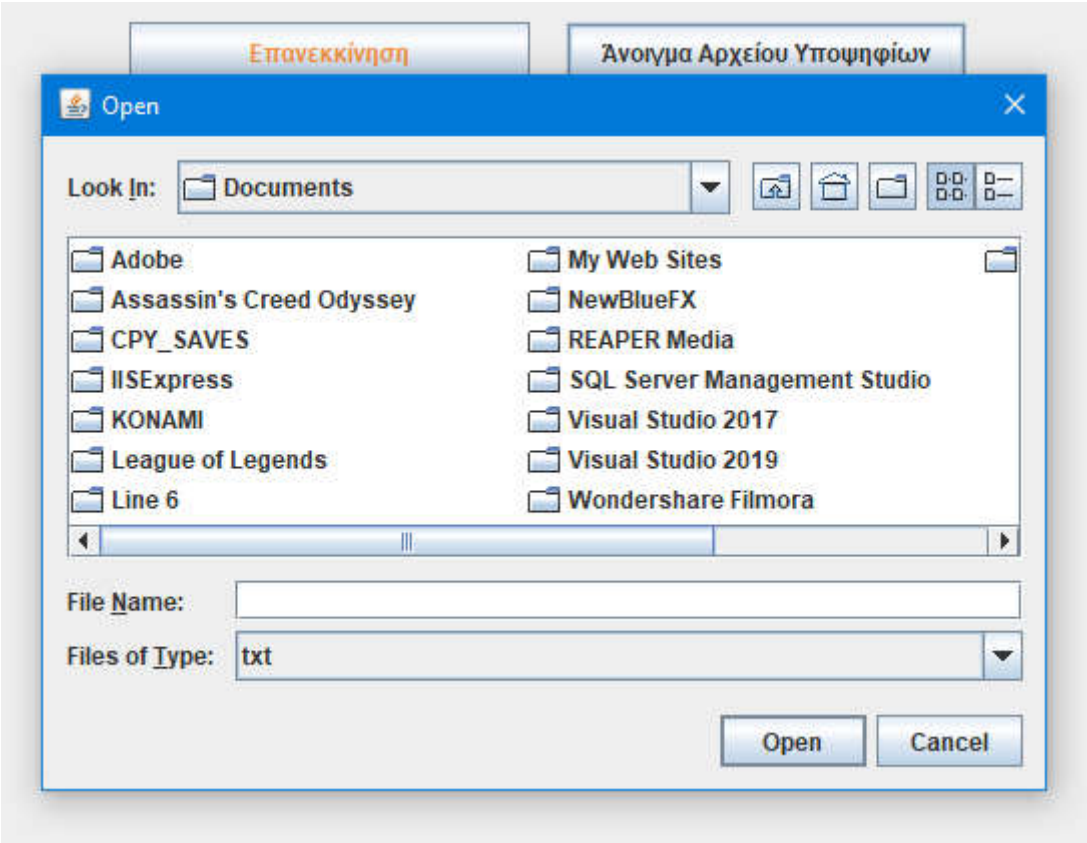
Μπορούμε να αλλάξουμε ορισμένες εντολές μέσα από τον κώδικα που δημιουργήθηκε αυτόματα. Υπάρχει αναλυτικά η λειτουργία της κάθε εντολής στο εγχειρίδιο χρήσης της εφαρμογής Inno Setup που υπάρχει διαθέσιμο στην ιστοσελίδα της εφαρμογής. Για να ολοκληρωθεί η διαδικασία δημιουργίας του αρχείου εγκατάστασης αρκεί να πατήσουμε το πράσινο κουμπί στο navigation bar στο πάνω μέρος της εφαρμογής (βλ. Εικόνα 3.1.2.5).



Εικόνα 3.1.2.5: Εκτέλεση του κώδικα για τη δημιουργία του αρχείου εγκατάστασης.

3.2 Λειτουργία

Για τη λειτουργία της εφαρμογής απαιτείται από το χρήστη να έχει έναν ηλεκτρονικό υπολογιστή, τη λίστα εισόδου (λίστα με τα στοιχεία των υποψηφίων), τον αριθμό των υποψηφίων που θέλει να κληρωθούν. Η εκτέλεση και χρήση της εφαρμογής μπορεί να πραγματοποιηθεί σε οποιοδήποτε υπολογιστικό σύστημα χωρίς να απαιτείται από τον ηλεκτρονικό υπολογιστή εξοπλισμός υψηλών προδιαγραφών (κάρτα γραφικών, επεξεργαστής τελευταίας τεχνολογίας). Όσον αφορά τη λίστα εισόδου, ο χρήστης έχει δύο τρόπους για να την εισάγει. Ο πρώτος τρόπος είναι μεταφορτώνοντας ένα αρχείο κειμένου (.txt), στο οποίο αναγράφονται τα στοιχεία των υποψηφίων για τη κλήρωση, ένας υποψήφιος ανά γραμμή. Για τη διαδικασία αυτή, ο χρήστης πρέπει να επιλέξει το κουμπί «Άνοιγμα Αρχείου Υποψηφίων» στο οποίο θα του ζητηθεί να αναζητήσει το αρχείο μέσα από τον Η/Υ του και να το επιλέξει (βλ. Εικόνα 3.2.1).



Εικόνα 3.2.1: Παράθυρο αναζήτησης και φόρτωσης λίστας εισόδου.

Ο δεύτερος τρόπος εισαγωγής της λίστας υποψηφίων στο σύστημα είναι η χειροκίνητη – μη αυτόματη εισαγωγή. Σε αυτή τη περίπτωση ο χρήστης ανοίγοντας την εφαρμογή πρέπει να εισάγει τους υποψηφίους και τα στοιχεία τους πληκτρολογώντας τα ένα προς ένα.

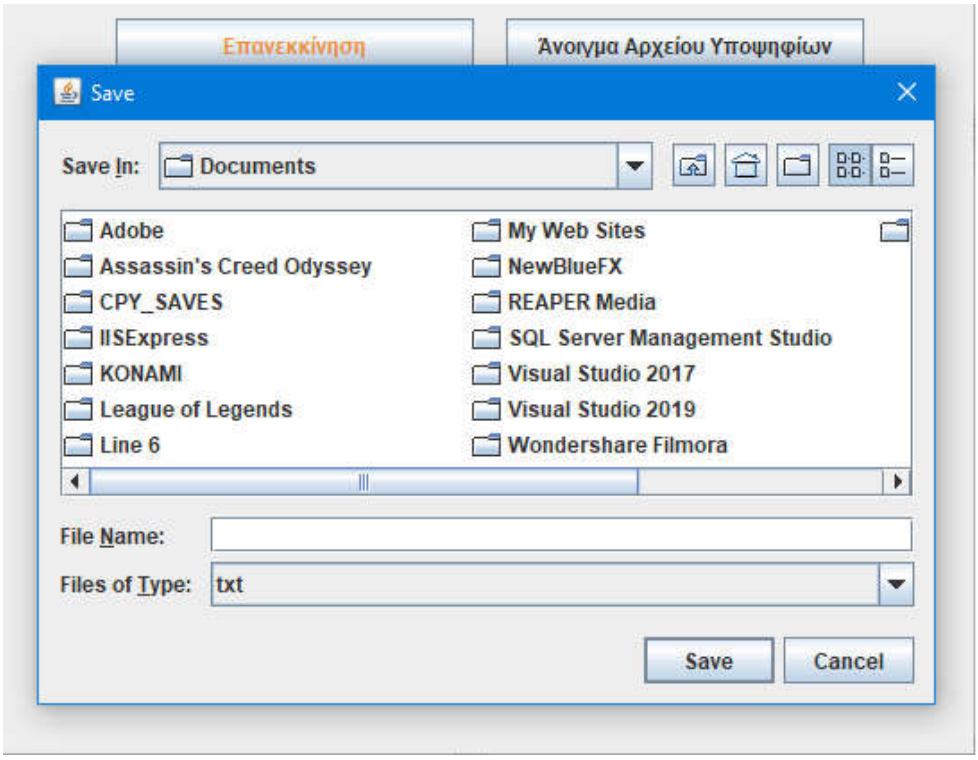
Αφού ο χρήστης έχει ολοκληρώσει τη διαδικασία εισαγωγής των υποψηφίων (λίστα εισόδου) το επόμενο στάδιο, η εκτέλεση των αλγορίθμων Fisher-Yates shuffle και Durstenfeld-Knuth shuffle και το τελικό αποτέλεσμα (λίστα εξόδου), γίνεται από την εφαρμογή.

Στην περίπτωση απόδοσης της τελικής λίστας εξόδου μέσω της χρήσης του αλγορίθμου Fisher-Yates shuffle, ο αλγόριθμος αφότου διαβάσει τη λίστα εισαγωγής επιλέγει τυχαία έναν υποψήφιο από την ήδη υπάρχουσα λίστα και τον μεταφέρει σε μία νέα λίστα, τη λίστα εξόδου, αφαιρώντας τον υποψήφιο από την αρχική λίστα εισαγωγής. Η διαδικασία αυτή επαναλαμβάνεται έως ότου ο αριθμός

των υποψηφίων που βρίσκονται στη λίστα εξόδου αντιστοιχεί με τον αριθμό που όρισε ο χρήστης.

Στη περίπτωση που ο χρήστης επιλέξει την εφαρμογή για την απόδοση της τελικής λίστας εξόδου μέσω του αλγορίθμου Durstenfeld-Knuth shuffle, ο αλγόριθμος όταν ολοκληρώσει τη διαδικασία ανάγνωσης της λίστας εισόδου, επιλέγει έναν τυχαίο υποψήφιο και μεταφέρει τον υποψήφιο με τα στοιχεία του στη τελευταία θέση της ίδιας λίστας χωρίς να χρειαστεί να δημιουργήσει νέα λίστα. Η διαδικασία αυτή επαναλαμβάνεται μέχρι ο τελικός αριθμός των υποψηφίων που μετατέθηκαν ισούται με τον αριθμό που όρισε ο χρήστης.

Το τελικό στάδιο, μετά την ολοκλήρωση της διαδικασίας εκτέλεσης των αλγορίθμων από την εφαρμογή, είναι το στάδιο ορισμού και αποθήκευσης της τελικής λίστας, δηλαδή η λίστα αποτελεσμάτων των αλγορίθμων (λίστα εξόδου υποψηφίων). Ο χρήστης καλείται να ονομάσει το αρχείο της τελικής λίστας (αρχείο κειμένου .txt) και να επιλέξει τη τοποθεσία αποθήκευσής της τοπικά στον ηλεκτρονικό του υπολογιστή (βλέπε Εικόνα 3.2.2).



Εικόνα 3.2.2: Τοποθεσία φακέλου αποθήκευσης και ονομασία λίστας εξόδου.

3.3 Ο κώδικας

3.3.1 Γραφικό περιβάλλον για αυτόματη είσοδος λίστας υποψηφίων

Σε πρώτη φάση, γίνεται αρχικοποίηση των μεταβλητών που χρειαζόμαστε και που εμφανίζονται στο παράθυρο της εφαρμογής όπως φαίνεται στην Εικόνα 3.3.1.1.

```
public class ShuffleGr extends JFrame implements ActionListener {

    JLabel listContains, itemsToDraw, successDraw, fileCreated;
    JFrame f;
    JButton open, save, shuffle, restart;
    JSpinner area;
    ArrayList<String> list = new ArrayList<>();
    File file;
```

Εικόνα 3.3.1.1: Αρχικοποίηση μεταβλητών (JLabels, JFrame, JButtons, JSpinner, ArrayList, File).

Το JFrame λειτουργεί ως πλαίσιο για να «φιλοξενήσει» τα στοιχεία που χρειαζόμαστε στο παράθυρο. Τα JLables είναι «ετικέτες» δηλαδή κείμενο που δείχνει π.χ. τα στοιχεία που περιέχει η λίστα κλπ. Τα JButtons είναι κουμπιά που το καθένα έχει διαφορετική λειτουργία ένα για να ανοίξει το παράθυρο μέσω του οποίου θα κάνουμε αναζήτηση για το αρχείο κειμένου που είναι η λίστα μας, άλλο για την αποθήκευση της νέας λίστας που δημιουργήθηκε, άλλο για να γίνει η υλοποίηση του αλγορίθμου που χρησιμοποιήθηκε και τέλος για την επανεκκίνηση της εφαρμογής . Μέσω του JSpinner επιλέγουμε πάσα στοιχεία θέλουμε να κληρωθούν ξεκινώντας από το 1 έως τον αριθμό των στοιχείων της λίστας που δόθηκε. Με τη χρήση του ArrayList αποθηκεύουμε τα στοιχεία της λίστας μας ένα προς ένα σε έναν πίνακα με θέσεις που αντιστοιχούν σε κάθε στοιχείο της λίστας. Τέλος το File είναι το αρχείο που θα δημιουργηθεί και θα αποθηκευτεί στη συνέχεια, το οποίο θα περιέχει τη νέα λίστα σε μορφή αρχείου κειμένου.

Στη συνέχεια ορίζουμε τη θέση που θα έχουν τα στοιχεία μας μέσα στο παράθυρο (JFrame) καθώς και το κείμενο που θα εμφανίζεται μέσα στο κάθε κουμπί που

δημιουργήσαμε και το όνομα που θα έχει το παράθυρο της εφαρμογής μας (βλ. Εικόνα 3.3.1.2).

```
public ShuffleGr() {
    f = new JFrame();
    f.setTitle("Fisher-Yates Shuffle");
    listContains = new JLabel();
    listContains.setBounds(167, 47, 300, 70);

    itemsToDraw = new JLabel();
    itemsToDraw.setBounds(127, 104, 300, 30);

    successDraw = new JLabel();
    successDraw.setBounds(187, 207, 250, 30);

    fileCreated = new JLabel();
    fileCreated.setBounds(57, 327, 450, 30);

    open = new JButton("Άνοιγμα Αρχείου Υποψηφίων");
    open.setBounds(283, 10, 200, 30);
    open.addActionListener(this);

    save = new JButton("Αποθήκευση Αρχείου");
    save.setBounds(177, 267, 180, 30);
    save.addActionListener(this);

    shuffle = new JButton();
    shuffle.setBounds(177, 157, 180, 30);
    shuffle.addActionListener(this);

    restart = new JButton("Επανεκίνηση");
    restart.setBounds(65, 10, 200, 30);
    Color color = Color.decode("#f59042");
    restart.setForeground(color);
    restart.addActionListener(this);
```

Εικόνα 3.3.1.2: Ορισμός στοιχείων που θα εμφανίζονται στο παράθυρο της εφαρμογής.

Στη συνέχεια προσθέτουμε τα στοιχεία μας στο JFrame και ορίζουμε το μέγεθος του παραθύρου της εφαρμογής (βλ. Εικόνα 3.3.1.3).

```
        f.add(listContains);
        f.add(itemsToDraw);
        f.add(successDraw);
        f.add(fileCreated);
        f.add(open);
        f.add(save);
        f.add(shuffle);
        f.add(restart);

        listContains.setVisible(true);
        itemsToDraw.setVisible(false);
        successDraw.setVisible(false);
        fileCreated.setVisible(false);
        shuffle.setVisible(false);
        save.setVisible(false);

        f.setSize(550, 450);
        f.setLayout(null);
        f.setLocationRelativeTo(null);
        f.setBackground(Color.GRAY);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setResizable(false);
    }
```

Εικόνα 3.3.1.3: Μέγεθος παραθύρου και εισαγωγή στοιχείων στο JFrame.

Όταν ο χρήστης επιλέξει «Άνοιγμα Αρχείου Υποψηφίων» εμφανίζεται ένα παράθυρο που του δίνει τη δυνατότητα να επιλέξει μέσα από τους φακέλους του υπολογιστή του το αρχείο με τη λίστα εισόδου και να μεταφορτώσει. Στη συνέχεια η εφαρμογή διαβάζει ένα προς ένα τα στοιχεία απο τη λίστα και τα αποθηκεύει σε μία λίστα πίνακα (ArrayList). Έπειτα εμφανίζεται το πλήθος των στοιχείων που περιέχει η λίστα. Εμφανίζει και το «JSpinner» που μέσω αυτού ο χρήστης έχει τη δυνατότητα να επιλέξει το πλήθος των υποψηφίων που θέλει να κληρωθούν. Στην περίπτωση που η λίστα εισόδου δεν έχει κανένα στοιχείο τότε ζητάει από το χρήστη να επιλέξει ένα αρχείο κειμένου που να περιέχει στοιχεία. (βλ. Εικόνα 3.3.1.4)

```
if (e.getSource() == open) {

    JFileChooser chooser = new JFileChooser();
    chooser.setFileFilter(new FileNameExtensionFilter("txt", "txt"));
    int answer = chooser.showOpenDialog(this);

    if (answer == JFileChooser.APPROVE_OPTION) {

        try {
            BufferedReader reader = new BufferedReader(new FileReader(chooser.getSelectedFile()));
            while (reader.ready()) {
                list.add(reader.readLine());
            }
        } catch (IOException ee) {
            System.out.println(ee.getMessage());
        }

        if (list.isEmpty()) {
            listContains.setText("<html>      "
                    + "            "
                    + "      Η λίστα είναι άδεια. "
                    + "<br>Παρακαλώ επιλέξτε ένα txt αρχείο που περιέχει ονοματίες/λαμπά");
        } else {
            listContains.setText("Το αρχείο περιέχει " + list.size() + " ονοματίες");
            itemToDraw.setText("Πρώτος αριθ.:πολύφυλλο κλειστόφυλλο");
            itemToDraw.setVisible(true);
            shuffle.setText("Ανακατέψτε");
            shuffle.setVisible(true);
            int arrayListSize = list.size();
            area = new JSpinner(new SpinnerNumberModel(1, 1, arrayListSize, 1));
            JFormattedTextField txt = ((JSpinner.NumberEditor) area.getEditor()).getTextField();
            ((NumberFormatter) txt.getFormatter()).setAllowsInvalid(false);
            area.setBounds(317, 100, 50, 30);
            area.setVisible(true);
            f.add(area);
```

Εικόνα 3.3.1.4: Λειτουργία κουμπιού «Άνοιγμα Αρχείου Υποψηφίων».

Με το πάτημα του κουμπιού «Ανακατέψτε» εκτελούνται οι αλγόριθμοι Fisher-Yates shuffle και Durstenfeld-Knuth shuffle αντίστοιχα (βλ. Εικόνα 3.3.1.5 και βλ. Εικόνα 3.3.1.6). Επίσης εμφανίζεται το μήνυμα ότι η κλήρωση έγινε με επιτυχία και το κουμπί «Αποθήκευση».

```
if (e.getSource() == shuffle) {
    Collections.shuffle(list);

    successDraw.setText("Η κλήρωση έγινε με επιτυχία");
    successDraw.setVisible(true);
    save.setVisible(true);
}
```

```
if (e.getSource() == shuffle) {
    Random r = new Random();
    int value = (Integer) area.getValue();
    int length = list.size();
    for (int i = length - 1; i >= length - value; --i) {
        Collections.swap(list, i, r.nextInt(i + 1));
    }

    successDraw.setText("Η κλήρωση έγινε με επιτυχία");
    successDraw.setVisible(true);
    save.setVisible(true);

}
```

Εικόνα 3.3.1.5 & Εικόνα 3.3.1.6: Fisher Yates & Durstenfeld-Knuth shuffle.

Όταν γίνει το πάτημα του κουμπιού «Αποθήκευση» τότε εμφανίζεται ένα παράθυρο μέσω του οποίου ο χρήστης έχει τη δυνατότητα να αποθηκεύσει το αρχείο τοπικά σε κάποιον φάκελο της επιλογής του και προστίθεται η ημερομηνία και η ώρα στο όνομα του αρχείου αυτόματα. Η κατάληξη ".txt" (αρχείο κειμένου) προστίθεται και αυτή αυτόματα χωρίς να είναι απαραίτητο ο χρήστης να την πληκτρολογήσει. Στη συνέχεια δημιουργείται η λίστα εξόδου που περιέχει το πλήθος των υποψηφίων που επιλέχθηκε προηγουμένως και ένας αύξων αριθμός πριν από το όνομα του κάθε υποψηφίου. Τέλος, εμφανίζεται το μήνυμα ότι το αρχείο δημιουργήθηκε με επιτυχία καθώς και το όνομα του αρχείου. (βλ. Εικόνα 3.3.1.7 και βλ. Εικόνα 3.3.1.8)

```
if (e.getSource() == save) {
    SimpleDateFormat formatter2 = new SimpleDateFormat("dd.MM.yyyy 'στις' HH.mm.ss");
    Date date2 = new Date();
    String dateToStr2 = formatter2.format(date2);
    JFileChooser chooser = new JFileChooser();

    chooser.setFileFilter(new FileNameExtensionFilter("txt", "txt"));

    int answer = chooser.showSaveDialog(this);
    file = chooser.getSelectedFile();
    if (FilenameUtils.getExtension(file.getName()).equalsIgnoreCase("txt")) {
        file = new File(file.toString() + "-" + dateToStr +   ".txt");
    } else {
        file = new File(file.toString() + "-" + dateToStr +   ".txt");
        file = new File(file.getParentFile(), FilenameUtils.getBaseName(file.getName()) + ".txt");
    }
    PrintWriter pw = null;
    if (answer == JFileChooser.APPROVE_OPTION) {
        try {
            BufferedWriter br = new BufferedWriter(
                    new OutputStreamWriter(new FileOutputStream(file, true), Charset.defaultCharset()

                    ));
            pw = new PrintWriter(br);

            int index = 0;
            int value = (Integer) area.getValue();
            for (String items : list.subList(0, value)) {
                ++index;
                pw.println(index + ". " + items);

            }
            pw.println("\n");
            pw.println("Δημιουργήθηκε "+dateToStr2);
```

Εικόνα 3.3.1.7: Αποθήκευση τοπικά σε φάκελο και δημιουργία λίστα εξόδου.

```
        } catch (FileNotFoundException ex) {
            Logger.getLogger(ShuffleGr.class.getName()).log(Level.SEVERE, null, ex);
        } finally {
            if (pw != null) {
                pw.close();
            }
        }
        fileCreated.setText("Το αρχείο "+ file.getName() + " δημιουργήθηκε με επιτυχία");
        fileCreated.setVisible(true);
    }
}
```

Εικόνα 3.3.1.8: Εμφάνιση μηνύματος δημιουργίας αρχείου.

Όταν ο χρήστης πατήσει το κουμπί «Επανεκκίνηση» τότε η εφαρμογή τερματίζεται και δημιουργείται ένα νέο παράθυρο με την εφαρμογή έτσι ώστε να κάνει νέα κλήρωση (βλ. Εικόνα 3.3.1.9).

```
if (e.getSource() == restart) {
    f.setVisible(false);
    new ShuffleGr();

}
```

Εικόνα 3.3.1.9: Επανεκκίνηση εφαρμογής.

3.3.2 Γραφικό περιβάλλον για χειροκίνητη είσοδο λίστας υποψηφίων

Δημιουργούμε πίνακες(arrays) για τα “JTextFields” (πεδία κειμένου) και για τα “JButtons” (κουμπιά) . Ο πρώτος πίνακας περιέχει τα κουμπιά για την αφαίρεση της γραμμής, ο δεύτερος για την προσθήκη νέας γραμμής και ο τρίτος και τελευταίος για τη διακοπή της λίστας (βλ. Εικόνα 3.3.2.1).

```
private JTextField textFields[] = new JTextField[50];
private JButton buttonsRem[] = new JButton[50];
private JButton buttonsAdd[] = new JButton[50];
private JButton buttonsEndLine[] = new JButton[50];
```

Εικόνα 3.3.2.1*: Πίνακες για τα "JTextFields" και για τα "JButtons"*

Δημιουργούμε ένα βρόχο επανάληψης (for loop) για να δημιουργήσουμε τα 50 κουμπιά και τα 50 πεδία κειμένου που χρειαζόμαστε. Στη συνέχεια χρησιμοποιούμε την εντολή "switch" για να επιλέξουμε κάθε περίπτωση "case" από όλες αυτές που θα χρειαστούμε (βλ. Εικόνα 3.3.2.2).

```
for (int i = 0; i < 50; i++) {
    txtField = new JTextField();
    btnRemove = new JButton("x");
    btnAdd = new JButton("✓");
    btnEndList = new JButton("#");

    textFields[i] = txtField;
    buttonsRem[i] = btnRemove;
    buttonsAdd[i] = btnAdd;
    buttonsEndLine[i] = btnEndList;

    firstLine(textFields,buttonsRem,buttonsAdd,buttonsEndLine,0);

    switch(i){
        case 0:
            btnAdd.addActionListener(e -> {
                addToList(textFields,0);
                increaseNumberOfSelection(textFields,0);
                buttonsRem[0].setEnabled(true);
                buttonsEndLine[0].setEnabled(true);
                addNewLine(textFields,buttonsRem,buttonsAdd,buttonsEndLine,1,1);
            });
            btnRemove.addActionListener(e -> {
                removeFromList(textFields,0);
                decreaseNumberOfSelection(textFields,0);
                removeLine(textFields,buttonsRem,buttonsAdd,buttonsEndLine,0);
            });
            btnEndList.addActionListener(e -> {
                endList(textFields,buttonsAdd,0);
                increaseNumberOfSelectionForEndLineBtn(textFields,0);
            });
            break;
```

Εικόνα 3.3.2.2: Βρόχος επανάληψης για τη δημιουργία "JTextField" και "JButtons".

Όταν ο χρήστης πατήσει το κουμπί για τη προσθήκη νέας γραμμής εκτελούνται 3 μέθοδοι (functions). Η πρώτη μέθοδος "addToList" λαμβάνει το κείμενο που έχει γράψει ο χρήστης στο "JTextField". Εάν δεν είναι κενό και δεν υπάρχει ήδη στη λίστα τότε το προσθέτει και αυξάνει το μέγεθος της λίστας κατά ένα. (βλ. Εικόνα 3.3.2.3)

```
public void addToList(JTextField[] jtf,int i){
    textFieldLine = jtf[i].getText();
    if(!textFieldLine.equals("")&&!list.contains(textFieldLine)){
        list.add(textFieldLine);
        listSize = list.size();
    }
}
```

Εικόνα 3.3.2.3: Μέθοδος "addToList".

Η μέθοδος "increaseNumberOfSelection" κάνουν τους ίδιους ελέγχους και προσθέτει στην αναπτυσσόμενη λίστα (dropdown box) το στοιχείο που πληκτρολόγησε έτσι ώστε στη συνέχεια να επιλέξει τον αριθμό υποψηφίων που θα κληρωθούν (βλ. Εικόνα 3.3.2.4).

```
public void increaseNumberOfSelection(JTextField[] jtf,int i){
    textFieldLine = jtf[i].getText();
    listSize = list.size();
    listSizeToString = String.valueOf(listSize);
    if(!textFieldLine.equals("")){
        listOfRequiredSelection.add(listSizeToString);
        model.addElement(listSizeToString);
        f.validate();
        f.repaint();
    }
}
```

Εικόνα 3.3.2.4: Μέθοδος "increaseNumberOfSelection".

Η μέθοδος "addNewLine" προσθέτει νέα γραμμή με τα 3 κουμπιά και το πεδίο κειμένου στο γραφικό περιβάλλον για να πληκτρολογήσει ο χρήστης νέο υποψήφιο στη λίστα (βλ. Εικόνα 3.3.2.5).

```
public void addNewLine(JTextField[] jtf,JButton[] del,JButton[] addNew,JButton[] endOf,int gridy,int i){
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx=12;
        c.ipady = 10;
        c.gridx = 0;
        c.gridy = gridy;
        panel.add(jtf[i], c);

        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx=1;
        c.ipady = 3;
        c.gridx = 1;
        c.gridy = gridy;
        panel.add(del[i], c);

        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx=1;
        c.ipady = 3;
        c.gridx = 2;
        c.gridy = gridy;
        panel.add(addNew[i], c);

        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx=1;
        c.ipady = 3;
        c.gridx = 3;
        c.gridy = gridy;
        panel.add(endOf[i], c);

        panel.validate();
        panel.repaint();
}
```

Εικόνα 3.3.2.5: Μέθοδος “addNewLine”.

Όταν ο χρήστης επιλέξει να αφαιρέσει κάποιο στοιχείο από τη λίστα τότε το κουμπί που πραγματοποιεί αυτή τη λειτουργία χρησιμοποιεί με τη σειρά του 3 μεθόδους. Η πρώτη μέθοδος “removeFromList” αφού κάνει τον έλεγχο που είχε αναφερθεί προηγουμένως τότε αφαιρεί από τη λίστα τον υποψήφιο (βλ. Εικόνα 3.3.2.6).

```
public void removeFromList(JTextField[] jtf,int i){
    textFieldLine = jtf[i].getText();
    if(!textFieldLine.equals("")){
        list.remove(textFieldLine);
        listSize = list.size();
    }
}
```

Εικόνα 3.3.2.6: Μέθοδος “removeFromList”.

Η μέθοδος “decreaseNumberOfSelection” αφαιρεί από το dropdonw box το στοιχείο αυτό μειώνοντας έτσι τις επιλογές του χρήστη όταν επιλέξει το πλήθος των υποψηφίων που θέλει να κληρωθούν (βλ. Εικόνα 3.3.2.7).

```
public void decreaseNumberOfSelection(JTextField[] jtf,int i){
    if(!jtf[i].getText().equals("")){
        listOfRequiredSelection.remove(listSizeToString);
        model.removeElement(listSizeToString);
        f.validate();
        f.repaint();
        listSizeToString = String.valueOf(listSize);
        if(listOfRequiredSelection.equals("0")){
            listOfRequiredSelection.remove(listSizeToString);
            model.removeElement(listSizeToString);
            f.validate();
            f.repaint();
        }
    }
}
```

Εικόνα 3.3.2.7: Μέθοδος “decreaseNumberOfSelection”.

Η μέθοδος “removeLine” αφαιρεί τελείως τη γραμμή από το γραφικό περιβάλλον (βλ. Εικόνα 3.3.2.8).

```
public void removeLine(JTextField[] jtf,JButton[] del,JButton[] addNew,JButton[] endOf,int i){
    panel.remove(jtf[i]);
    panel.remove(del[i]);
    panel.remove(addNew[i]);
    panel.remove(endOf[i]);
    panel.validate();
    panel.repaint();
}
```

Εικόνα 3.3.2.8: Μέθοδος “removeLine”.

Εάν ο χρήστης επιλέξει το κουμπί « ■ » τότε καλούνται 2 μέθοδοι. Η πρώτη μέθοδος

“endList” σταματάει τη λειτουργία του κουμπιού που προσθέτει νέα γραμμή έτσι ώστε ο χρήστης να μη μπορεί να το χρησιμοποιήσει. Τέλος προσθέτει στη λίστα το όνομα του υποψηφίου που πληκτρολόγησε ο χρήστης στο πεδίο κειμένου (βλ. Εικόνα 3.3.2.9).

```
public void endList(JTextField[] jtf,JButton[] del,int i){
    del[i].setEnabled(false);
    textFieldLine = jtf[i].getText();
    if(!textFieldLine.equals("")&&!list.contains(textFieldLine)){
        list.add(textFieldLine);
        listSize = list.size();
    }
}
```

Εικόνα 3.3.2.9: Μέθοδος “endList”.

Η μέθοδος “increaseNumberOfSelectionForEndLineBtn” αυξάνει τον αριθμό στο dropdown box (βλ. Εικόνα 3.3.2.10).

```
public void increaseNumberOfSelectionForEndLineBtn(JTextField[] jtf,int i){
    textFieldLine = jtf[i].getText();
    listSize = list.size();
    listSizeToString = String.valueOf(listSize);
    if(!textFieldLine.equals("")){
        listOfRequiredSelection.add(listSizeToString);
        model.addElement(listSizeToString);
        f.validate();
        f.repaint();
    }
}
```

*Εικόνα 3.3.2.10: Μέθοδος “*increaseNumberOfSelectionForEndLineBtn*”.*

3.3.3 Χειροκίνητη είσοδος μέσω γραμμής εντολών (cmd)

Δημιουργούμε ένα ArrayList για να αποθηκεύσουμε τα στοιχεία που θα πληκτρολογήσει ο χρήστης και τυπώνουμε στη γραμμή εντολών τις οδηγίες για το πως πρέπει ο χρήστης να πληκτρολογήσει τα στοιχεία. Όταν ο χρήστης πληκτρολογήσει “END” τότε σταματάει η διαδικασία εισαγωγής υποψηφίων στη λίστα. Όταν πληκτρολογήσει “exit” τότε τερματίζεται η εφαρμογή. Η εντολή “scanner” διαβάζει το τι έχει πληκτρολογήσει ο χρήστης και το αποθηκεύει στη μεταβλητή “input” και τα εισάγει στη λίστα Εμφανίζεται στην οθόνη ο αριθμός των υποψηφίων που έχει εισάγει ο χρήστης. Στη συνέχεια εμφανίζεται μήνυμα στη γραμμή εντολών το οποίο είναι το πλήθος υποψηφίων από τη λίστα. Εάν η λίστα είναι κενή δηλαδή αν ο χρήστης δεν έχει πληκτρολογήσει κανένα στοιχείο τότε εμφανίζει μήνυμα στη γραμμή εντολών να πληκτρολογήσει ο χρήστης “Enter” για να ξεκινήσει από την αρχή η διαδικασία εισαγωγής. Τέλος, με την εντολή “System.exit(0)” τερματίζεται η εφαρμογή (βλ. Εικόνα 3.3.3.1).

```
boolean start = true;
while (start) {
    ArrayList<String> list = new ArrayList<String>();
    Scanner scan = new Scanner(System.in);
    System.out.println("Πληκτρολογήστε το στοιχείο προς κλήρωση \n -- εισάγετε 1 στοιχείο ανά γραμμή "
            + "\n -- πατήστε ENTER για να μεταβείτε στην επόμενη γραμμή "
            + "\n -- πληκτρολογήστε END για λήξη της διαδικασίας καταγραφής "
            + "\n -- πληκτρολογήστε exit για τερματισμό της εφαρμογής");
    String input = scan.nextLine();
    if (input.equalsIgnoreCase("exit")) {
            System.exit(0);
    }

while (!input.equalsIgnoreCase("END")) {
    list.add(input);
    input = scan.nextLine();
}

if(list.size() == 0){
    System.out.println("Δεν καταγραφήκατε στοιχεία \n --πατήστε ENTER για να ξεκινήσετε από την αρχή");
    input = scan.nextLine();
    if(input.equalsIgnoreCase("exit")){
        System.exit(0);
    }
}
System.out.println("Καταγραφήκατε " + list.size() + " στοιχεία");
System.out.println("Πόσα στοιχεία θέλετε να κληρωθούν; "
        + "\n -- πληκτρολογήστε ακέραιο αριθμό από 1 έως "
        + list.size() + "");
Scanner scan2 = new Scanner(System.in);
```

Εικόνα 3.3.3.1: Εισαγωγή και εμφάνιση πλήθους υποψηφίων.

Στη συνέχεια γίνεται έλεγχος για το αν ο χρήστης πληκτρολόγησε έγκυρο αριθμό. Σε περίπτωση που ο αριθμός δεν είναι ανάμεσα στο 1 και στο πλήθος της λίστας τότε ζητάει να πληκτρολογήσει έναν έγκυρο αριθμό. Ο έλεγχος αυτός εκτελείται μέχρι και 3 φορές, αν υπερβεί τον αριθμό αυτό τότε η εφαρμογή τερματίζεται. Τέλος τυπώνει τη λίστα των υποψηφίων και εκτελούνται οι αλγόριθμοι Fisher – Yates και Durstenfeld – Knuth shuffle αντίστοιχα (βλ. Εικόνα 3.3.3.2).

```
if(scan2.hasNextInt()){
    int user_num = scan2.nextInt();
    while (user_num == 0 || user_num > list.size()) {
        System.out.println("Πληκτρολογήστε ακέραιο αριθμό από 1 έως " + list.size())
    if(scan2.hasNextInt()){
        user_num = scan2.nextInt();
    }
    else{
        String strg = scan2.next();
        if(strg.equalsIgnoreCase("exit")){
          System.exit(0);
        }
        }
    }

    Collections.shuffle(list);
    System.out.println("Τα στοιχεία που κληρώθηκαν είναι:");
    int index = 1;
    for (String items : list.subList(0, user_num)) {
            System.out.println((index++) + ". " + items);
    }
}
else{
    String strg = scan2.next();
    if(strg.equalsIgnoreCase("exit")){
        System.exit(0);
    }
}
```

Εικόνα 3.3.3.2: Έλεγχος για τον αριθμό που πληκτρολόγησε ο χρήστης και εκτέλεση των αλγορίθμων Fisher – Yates και Durstenfeld – Knuth shuffle αντίστοιχα.

Τέλος εμφανίζεται μήνυμα για το αν επιθυμεί ο χρήστης να πραγματοποιήσει νέα κλήρωση. Σε περίπτωση που πληκτρολογήσει “no” η εφαρμογή τερματίζεται. Εάν πληκτρολογήσει “yes” τότε εκτελείται από την αρχή η εφαρμογή (βλ. Εικόνα 3.3.3.3).

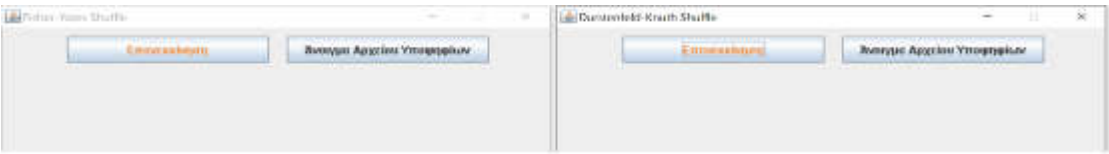
```
Scanner scan3 = new Scanner(System.in);
System.out.println("Επιθυμείτε να εκτελέσετε ξανά την εφαρμογή; "
        + "\n -- πληκτρολογήστε yes για επανεκκίνηση "
        + "\n -- πληκτρολογήστε no για τερματισμό της εφαρμογής");
String repeat = scan3.nextLine();

if (!repeat.equals("yes")) {
        System.exit(0);
}
}
}
```

Εικόνα 3.3.3.3: Επιλογή νέας κλήρωσης ή τερματισμός προγράμματος.

3.4 Επίδειξη λειτουργίας

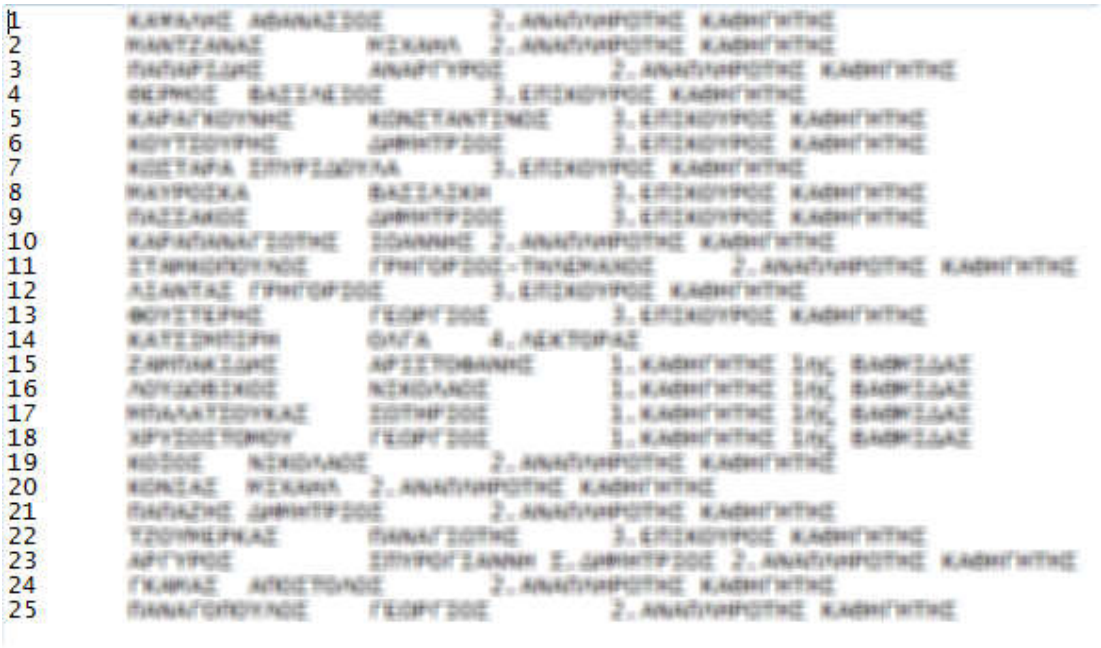
3.4.1 Γραφικό περιβάλλον για αυτόματη είσοδο λίστας υποψηφίων

Και τα δύο παράθυρα εφαρμογών, το Fisher-Yates shuffle και το Durstenfeld-Knuth shuffle, παρέχουν μια επιλογή για το άνοιγμα ενός αρχείου κειμένου που περιέχει τους υποψηφίους και μια επιλογή για επανεκκίνηση του προγράμματος (Εικόνα 3.1.1).



Εικόνα 3.4.1.1: Γραφικό περιβάλλον εφαρμογών.

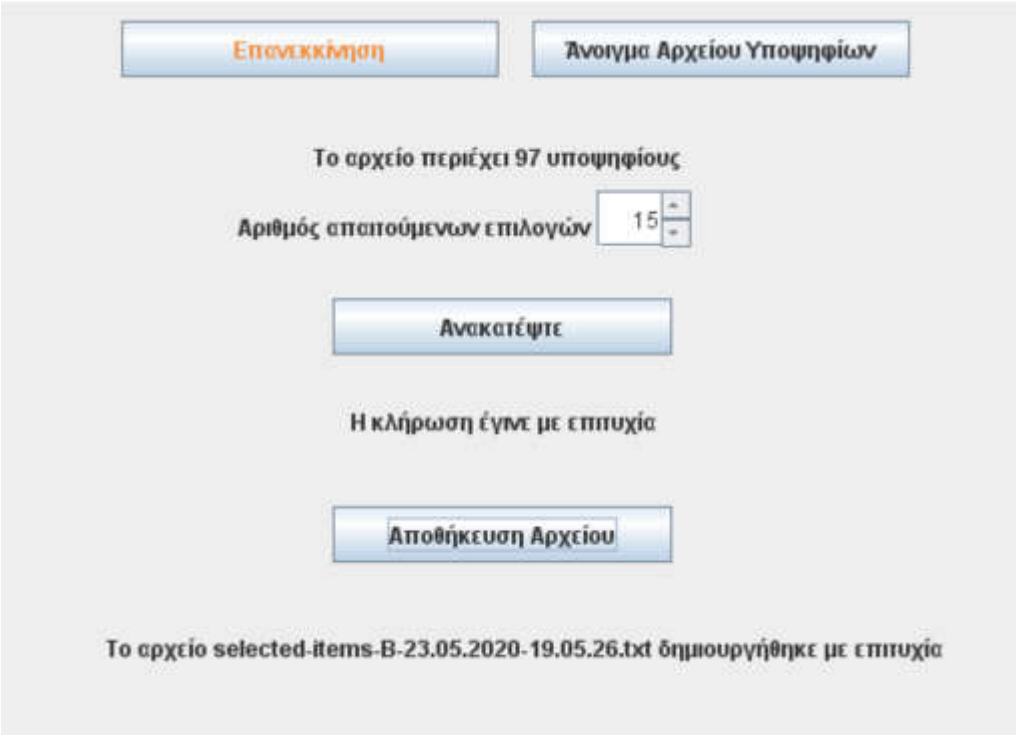
Χρησιμοποιούμε 3 δοκιμαστικά αρχεία που περιέχουν το επώνυμο των καθηγητών σε ελληνικά πανεπιστήμια (δημόσιες πληροφορίες). Το αρχείο "input-list-A.txt" περιέχει 97 καταχωρήσεις, το input-list-B.txt περιέχει 542 καταχωρήσεις και το input-list-C.txt περιέχει 10.013 καταχωρήσεις. Θεωρούμε ότι αυτά τα αρχεία είναι ενδεικτικά ως προς το μέγεθος και των πληροφοριών που υπάρχουν, καθώς αυτό το είδος αρχείων («μητρώα») χρησιμοποιείται ως ομαδοποίηση για τη σύσταση 22-μελών επιτροπών για την εκλογή / προώθηση των μελών της σχολής στην πράξη. Ένα αρχείο υποψηφίων πρέπει να είναι ένα απλό αρχείο κειμένου που περιέχει έναν - διακριτό - υποψήφιο ανά γραμμή (βλ. Εικόνα 3.4.1.2).



Εικόνα 3.4.1.2: Ένα έγκυρο αρχείο υποψηφίων πρέπει να είναι ένα απλό αρχείο κειμένου που περιέχει έναν – διακριτό – υποψήφιο ανά γραμμή.

Για τη διεξαγωγή δοκιμαστικής κλήρωσης (βλ. Εικόνα 3.4.1.3), επιλέγουμε πρώτα "Άνοιγμα Αρχείου Υποψηφίων" και στη συνέχεια αναζητούμε το αρχείο "input-list-A.txt". Ανοίγοντας το αρχείο εισόδου, οι καταχωρήσεις καταμετρώνται αυτόματα και εμφανίζονται στην οθόνη μαζί με ένα μετρητή υποψηφίων που θα κληρωθούν. Ο μετρητής για τους ζητούμενους υποψηφίους ξεκινά από το 1 και δεν μπορεί να υπερβαίνει τον συνολικό αριθμό συμμετεχόντων. Παρέχοντας τον απαιτούμενο αριθμό υποψηφίων και μετά επιλέγοντας "Ανακατέψτε", επικαλούμαστε την εκτέλεση των αλγορίθμων Fisher-Yates shuffle και Durstenfeld-Knuth shuffle σε όλους τους καταχωρημένους υποψηφίους που επιστρέφει μια πραγματικά τυχαία παραλλαγή της λίστας εισόδου. Στη συνέχεια, επιστρέφεται ένα πρόθεμα μεγέθους ίσο με τον υποβαλλόμενο αριθμό των απαιτούμενων στοιχείων των υπολογισμένων υποψηφίων και μπορεί να αποθηκευτεί σε ένα απλό αρχείο κειμένου μέσω του "Αποθήκευση αρχείου". Το αρχείο εξόδου είναι ένα αρχείο κειμένου που καθορίζεται από το χρήστη και περιέχει τα υπολογισμένα μέλη, ένα ανά γραμμή

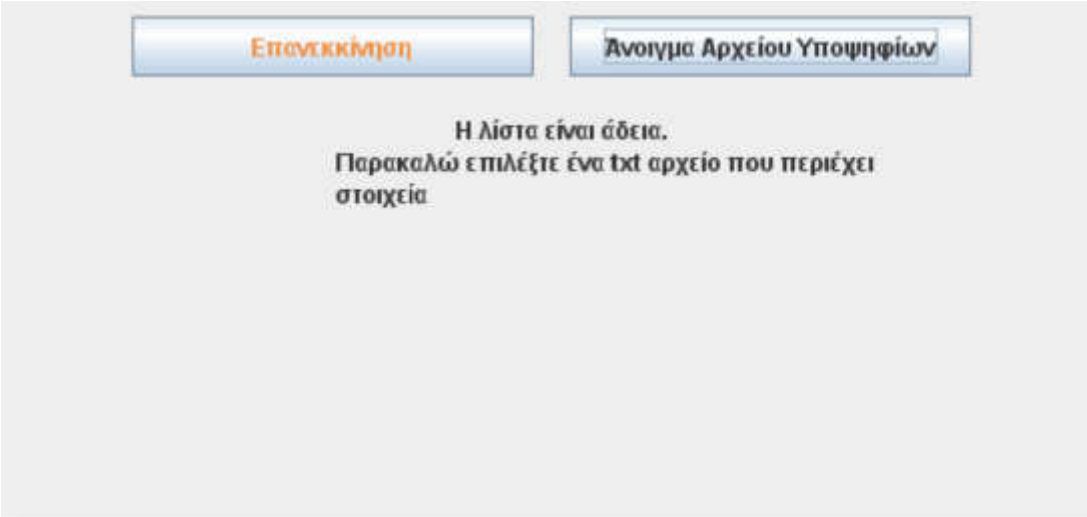
(βλ. Εικόνα 3.4.1.4). Μπορεί εύκολα να παρατηρηθεί ότι ο χρόνος που απαιτείται για τον υπολογισμό μιας ανακατεμένης λίστας εισόδου είναι εξαιρετικά μικρός. Δεδομένης της τρέχουσας τεχνολογίας και των σχετικά σύντομων λιστών εισόδου (που χρησιμοποιούνται στο πλαίσιο ενός πανεπιστημιακού τμήματος), η εκτέλεση της εφαρμογής διαρκεί ένα κλάσμα του δευτερολέπτου, όπως αναμενόταν.



Εικόνα 3.4.1.3: Ενδεικτική ηλεκτρονική κλήρωση σε μία λίστα εισόδου 97 υποψηφίων.

Εικόνα 3.4.1.4: Το αρχείο εξόδου που περιέχει τα ζητούμενα μέλη της επιτροπής.

Εάν η λίστα εισόδου δεν περιέχει στοιχεία τότε εμφανίζεται μήνυμα στο παράθυρο της εφαρμογής (βλ. Εικόνα 3.4.1.5) ότι η λίστα που ανέβηκε είναι κενή και ζητάει από το χρήστη να επιλέξει διαφορετικό αρχείο που δεν είναι λανθασμένο.



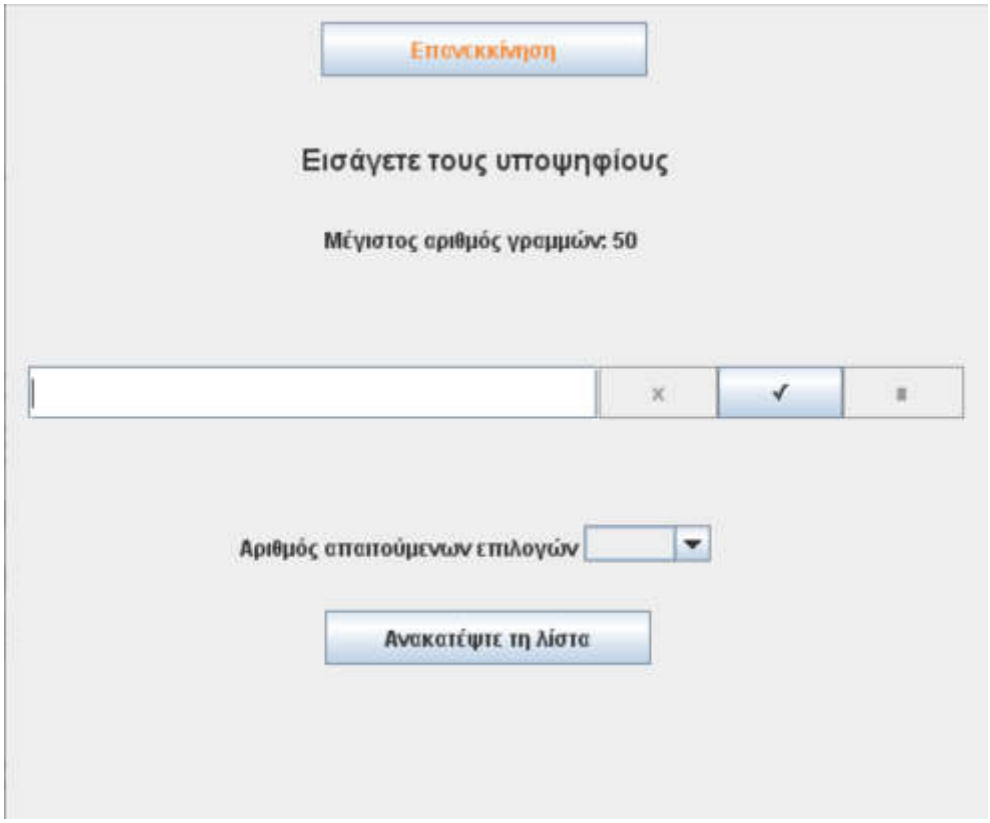
Εικόνα 3.4.1.5: Το αρχείο δεν περιέχει κανένα υποψήφιο.

Μετά την ολοκλήρωση μιας κλήρωσης, μπορούμε είτε να κλείσουμε την εφαρμογή κλείνοντας το παράθυρο εκτέλεσης είτε μπορούμε να προχωρήσουμε σε μια νέα κλήρωση επιλέγοντας "Επανεκκίνηση" (βλ. Εικόνα 3.4.1.6).

Εικόνα 3.4.1.6: Επιλογή κουμπιού «Επανεκκίνηση» για νέα κλήρωση.

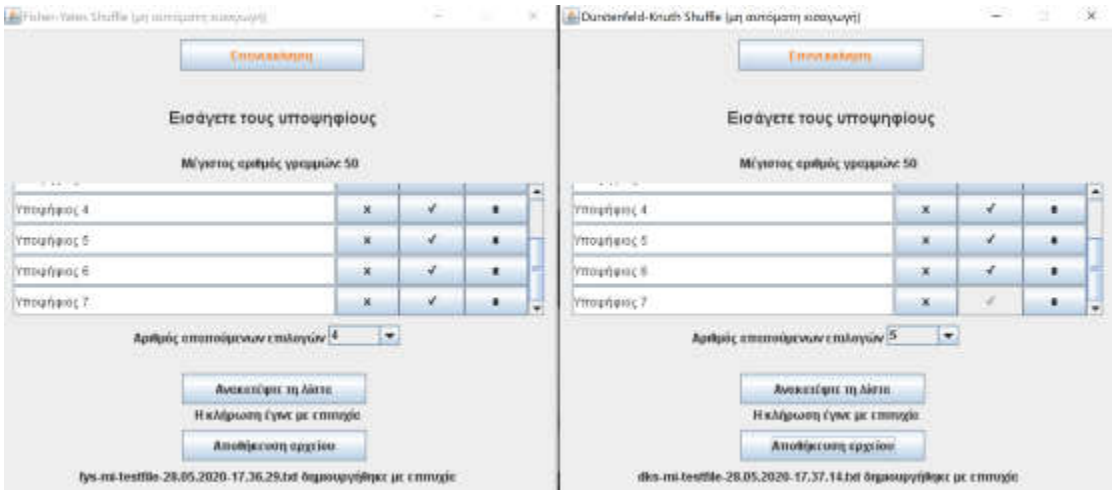
3.4.2 Γραφικό περιβάλλον για χειροκίνητη είσοδο λίστας υποψηφίων

Και τα δύο παράθυρα εφαρμογών, το Fisher-Yates shuffle και το Durstenfeld-Knuth shuffle, παρέχουν μια επιλογή για τη χειροκίνητη εισαγωγή λίστας υποψηφίων και μια επιλογή για επανεκκίνηση του προγράμματος (βλ. Εικόνα 3.4.2.1). Στη συνέχεια, παρουσιάζουμε το Durstenfeld-Knuth shuffle. Ένα παρόμοιο GUI με παρόμοιες λειτουργίες παρέχεται για το Fisher-Yates shuffle. Μπορούν να πληκτρολογηθούν έως και 50 υποψήφιοι, ένας ανά γραμμή. Δίπλα σε κάθε γραμμή, παρέχονται οι επιλογές για κατάργηση της τρέχουσας γραμμής, προσθήκη νέας γραμμή ή ολοκλήρωση της λίστας.



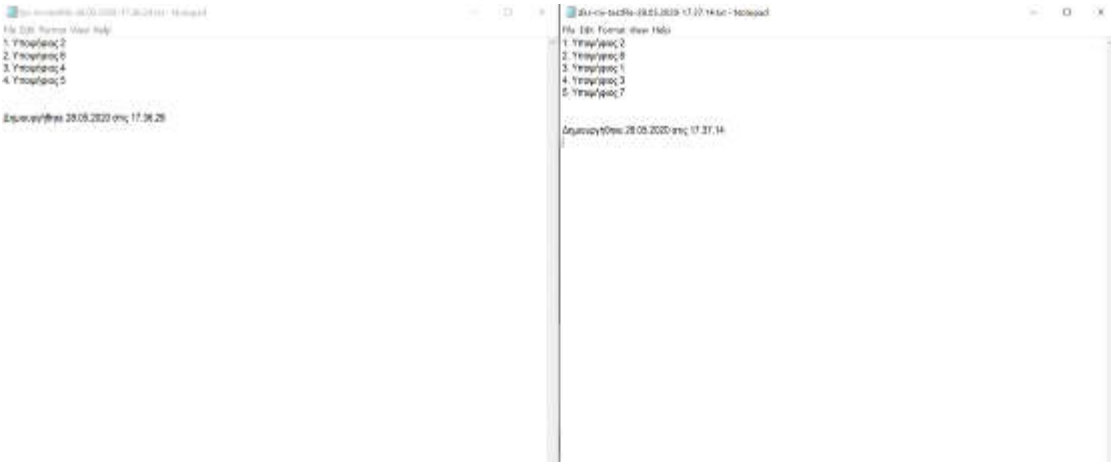
Εικόνα 3.4.2.1: Εκτέλεση ηλεκτρονικής κλήρωσης πληκτρολογώντας υποψηφίους.

Για τον σκοπό της επίδειξης αυτών των στοιχείων της εφαρμογής, πληκτρολογούμε 7 υποψηφίους, δηλαδή, Υποψήφιος 1 έως Υποψήφιος 7. Ολοκληρώνουμε τη λίστα (πατώντας το μικρό κουτί δίπλα στην τελευταία γραμμή) και μετά ζητάμε την τυχαία επιλογή δυο ομάδων 4 και 5 μελών μέσω του Fisher-Yates shuffle και Durstenfeld-Knuth shuffle, αντίστοιχα. Επιλέγοντας "Ανακατέψτε τη λίστα", ολοκληρώνουμε την κλήρωση. Στη συνέχεια, μας ζητείται να αποθηκεύσουμε τη λίστα τοπικά, σε ένα αρχείο κειμένου. Εμφανίζονται πλήρεις πληροφορίες για τα αποθηκευμένα αρχεία που περιέχουν τα αποτελέσματα κλήρωσης και οι διαδικασίες τερματίζονται (βλέπε Εικόνα 3.4.2.2).



Εικόνα 3.4.2.2: Ολοκλήρωση ηλεκτρονικών κληρώσεων πληκτρολογώντας υποψηφίους.

Τα αρχεία εξόδου είναι αρχεία κειμένου που καθορίζονται από το χρήστη και περιέχουν τα υπολογισμένα μέλη, ένα ανά γραμμή. Προστίθεται στο τέλος των στοιχείων της λίστα η ημερομηνία και η ώρα που πραγματοποιήθηκε η κλήρωση (Εικόνα 3.4.2.3). Μπορεί εύκολα να παρατηρηθεί ότι ο χρόνος που απαιτείται για τον υπολογισμό μιας ανακατεμένης λίστας εισόδου είναι εξαιρετικά μικρός.



Εικόνα 3.4.2.3: Λίστες εξόδου με την ημερομηνία και ώρα που δημιουργήθηκαν.

3.4.3 Χειροκίνητη είσοδος μέσω γραμμής εντολών (cmd)

Όταν αποφασίζουμε να παρέχουμε την είσοδο χειροκίνητα, δηλαδή απλά πληκτρολογώντας τα στοιχεία εισόδου, χρησιμοποιείται το περιβάλλον γραμμής εντολών. Τα στοιχεία εισαγωγής πρέπει να υποβάλλονται ένα ανά γραμμή. Μετά την υποβολή του αριθμού των ζητηθέντων στοιχείων (εντός του εύρους του διαθέσιμου πληθυσμού στη λίστα εισαγωγής), τα τυχαία επιλεγμένα στοιχεία εμφανίζονται στην οθόνη με τη μορφή λίστας. Στην Εικόνα 3.4.3.1 φαίνεται ένα ενδεικτικό παράδειγμα που εκμεταλλεύεται τον αλγόριθμο Durstenfeld-Knuth shuffle. Ένα παρόμοιο GUI με παρόμοιες λειτουργίες παρέχεται για τον αλγόριθμο Fisher-Yates shuffle.

```
Πληκτρολογήστε τα στοιχεία προς κλήρωση
 -- εισάγετε 1 στοιχείο ανά γραμμή
 -- πατήστε ENTER για να μεταβείτε στην επόμενη γραμμή
 -- πληκτρολογήστε END για λήξη της διαδικασίας καταχώρησης
 -- πληκτρολογήστε exit για τερματισμό της εφαρμογής
Ονομα Επωνυμο 1
Ονομα Επωνυμο 2
Ονομα Επωνυμο 3
Ονομα Επωνυμο 4
Ονομα Επωνυμο 5
Ονομα Επωνυμο 6
Ονομα Επωνυμο 7
END
Καταχωρήσατε 7 στοιχεία
Πόσα στοιχεία θέλετε να κληρωθούν;
 -- πληκτρολογήστε ακέραιο αριθμό από 1 έως 7
3
Τα στοιχεία που κληρώθηκαν είναι:
1. Ονομα Επωνυμο 1
2. Ονομα Επωνυμο 4
3. Ονομα Επωνυμο 2
Επιθυμείτε να εκτελέσετε ξανά την εφαρμογή;
 -- πληκτρολογήστε yes για επανεκκίνηση
 -- πληκτρολογήστε no για τερματισμό της εφαρμογής
```

Εικόνα 3.4.3.1: Το αρχείο εξόδου που περιέχει τα ζητούμενα μέλη της επιτροπής.

4 Επίλογος

Με κίνητρο την αναποτελεσματικότητα που παρατηρείται στην πράξη στο πλαίσιο της διαδικασίας σύστασης τυχαία επιλεγμένων σωμάτων από μια συγκεκριμένη ομάδα σε πανεπιστημιακά τμήματα, σχεδιάσαμε και υλοποιήσαμε μια απλή, εύχρηστη εφαρμογή για τη διεξαγωγή ηλεκτρονικών κληρώσεων προκειμένου να υπάρχει η δυνατότητα σύστασης επιτροπών διοικητικών ή εκπαιδευτικών φορέων μέσω πραγματικά τυχαίας επιλογής των μελών τους.

Η πρότασή μας έχει ως στόχο να δώσει μια αποδοτικότερη εναλλακτική λύση για τις κληρώσεις που διεξάγονται χειροκίνητα στο παραδοσιακό περιβάλλον με "λαχνό". Η προσέγγισή μας είναι θεωρητικά ορθή, δεδομένου ότι στην πραγματικότητα εφαρμόζει μια βελτιωμένη έκδοση του Fisher-Yates shuffle, δηλαδή, τον αλγόριθμο Durstenfeld-Knuth shuffle, ο οποίος δημιουργεί αποδεδειγμένα τυχαίες και αμερόληπτες μεταθέσεις λιστών. Υποθέτοντας ότι τα αρχεία εισόδου δεν είναι

σκόπιμα παραποιημένα, η εφαρμογή μας είναι αποδοτική, δηλαδή εγγυάται πραγματικά τυχαία, αμερόληπτη και δίκαιη κλήρωση που διεξάγεται γρήγορα.

Επιπλέον, στοιχειώδεις δεξιότητες και γνώσεις στη χρήση ηλεκτρονικών υπολογιστών από το εμπλεκόμενο διοικητικό προσωπικό, αρκούν για τη χρήση της εφαρμογής, η οποία, επιπλέον, μπορεί να εκτελεστεί σε οποιονδήποτε δημόσιο ή προσωπικό υπολογιστή χωρίς να απαιτείται τροποποίηση ή εγκατάσταση πρόσθετων πακέτων λογισμικού.

Σκοπεύουμε να αξιολογήσουμε την εφαρμογή μας στην πράξη ώστε σύντομα να έχουμε στατιστικά στοιχεία χρήσης. Σχεδιάζουμε επίσης να εμπλουτίσουμε αυτήν την αρχική έκδοση με πρόσθετες λειτουργίες, όπως, για παράδειγμα, λεπτομερή αυτόματο έλεγχο της λίστας εισόδου για ασυνέπειες, χρονική σήμανση του αρχείου εξόδου κ.λπ. Σχεδιάζουμε επίσης να βελτιώσουμε το γραφικό περιβάλλον διεπαφής χρήστη (GUI) αντικαθιστώντας μηνύματα κειμένου με στοιχεία εικόνας και εστιάζοντας επίσης στην απόκρισή του.

Οι αλγόριθμοι, η επιστήμη των υπολογιστών και η σύγχρονη τεχνολογία μπορούν να υποστηρίξουν σε μεγάλο βαθμό την απλοποίηση των διοικητικών διαδικασιών, αυξάνοντας παράλληλα τη διαφάνεια και την εμπιστοσύνη. Η εφαρμογή μας συνεισφέρει προς αυτή την κατεύθυνση και αποτελεί ένα μικρό αλλά ενδιαφέρον βήμα προς την επίτευξη της απλούστευσης των διαδικασιών στο οικοσύστημα της εκπαίδευσης ειδικά αλλά και γενικότερα στο ευρύτερο περιβάλλον της δημόσιας διοίκησης.

Αναφορές

1. N. Charitos, E. Papaioannou. "DRAWS": an application for small-scale electronic draws. Technium: Romanian Journal of Applied Sciences and Technology (ISSN: 2668-778X), vol. 2, no 3, pp. 43-48, 2020.
2. N. Charitos, E. Papaioannou. A java application for the generation of randomly selected bodies. In Proc. of the 12th annual International Conference on Education and New Learning Technologies (EDULEARN 20), 2020, to appear.
3. R. Durstenfeld. Algorithm 235: Random permutation. Communications of the ACM, vol. 7, no. 7, p. 420, 1964.
4. R. A. Fisher, F. Yates. Statistical tables for biological, agricultural and medical research (3rd edn). London: Oliver & Boyd, 1938.
5. D. E. Knuth. The Art of Computer Programming. Reading, Massachusetts: Addison-Wesley, 1969.
6. https://netbeans.apache.org/
7. https://netbeans.org/features/java-on-client/swing.html
8. https://sourceforge.net/projects/launch4j/files/launch4j-3/3.12/
9. https://jrsoftware.org/isinfo.php