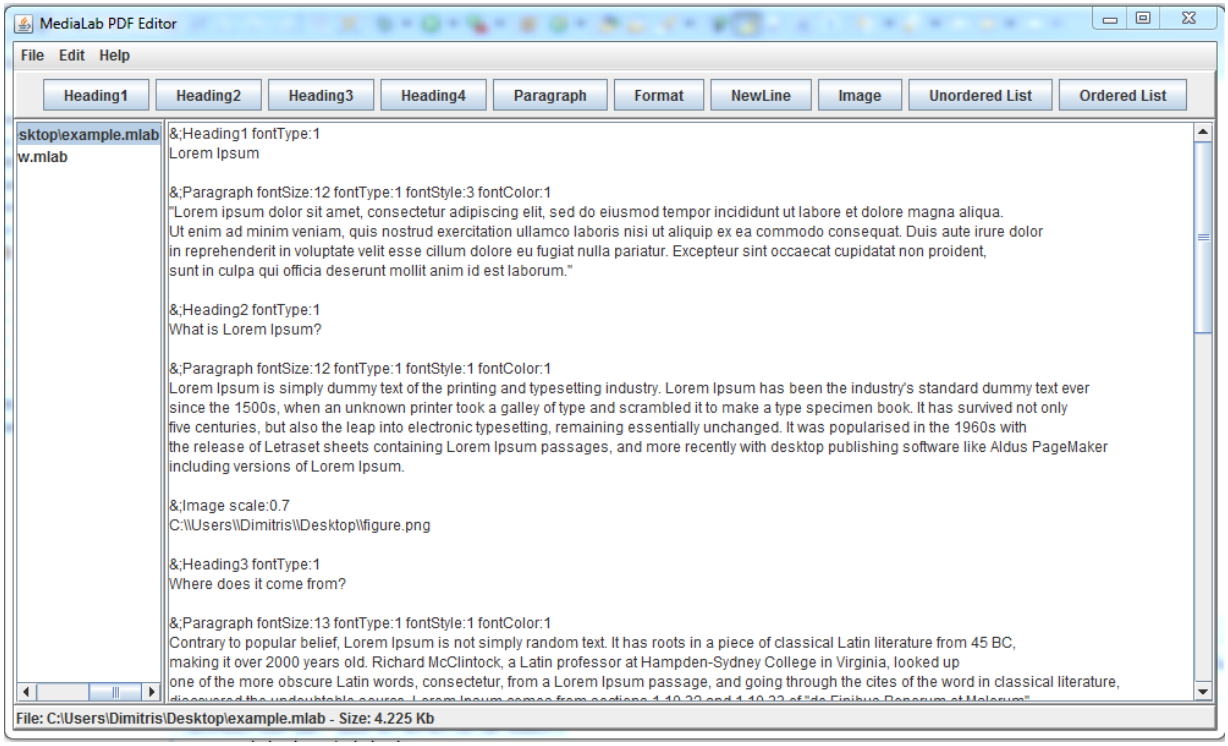
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνολογία Πολυμέσων

7ο εξάμηνο



Βιβλιοθήκη MediaLab PDF Editor σε Java

Αναφορά project

ΟΝΟΜΑΤΕΠΩΝΥΜΟ : **ΚΑΛΟΓΕΡΟΠΟΥΛΟΣ ΔΗΜΗΤΡΙΟΣ**

Α.Μ. : **03111179**

Σκοπός

Σκοπός του project είναι η ανάπτυξη μιας desktop εφαρμογής σε Java με δυνατότητες δημιουργίας και επεξεργασίας PDF αρχείων με βάση ένα σύνολο απλών εντολών μορφοποίησης και τη βιβλιοθήκη PDFBox.

Ξεκινώντας

Το project αυτό υλοποιήθηκε στο Eclipse IDE και έγινε χρήση της JavaSE-1.7.

Αρχικά, για τις ανάγκες της βιβλιοθήκης χρειάζεται να κάνουμε import στο project μας τα εξής jar αρχεία :

- *pdfbox-app-1.8.10.jar :* βιβλιοθήκη PDFBox έκδοσης 1.8.10 πάνω στην οποία βασίστηκε το project
- *java-json.jar :* επεξεργασία JSON δεδομένων που χρειάστηκε να επεξεργαστούμε στην κλάση OnlineContent για τη λήψη περιεχομένου από web υπηρεσία.

Α΄ μέρος – βιβλιοθήκη - σχόλια

Για το Α΄ μέρος του project δημιουργήσαμε 4 βασικές κλάσεις, οι οποίες είναι οι εξής:

1. *Mlab.java*

Υλοποιούμε τη διαχείριση αρχείων mlab για τη δημιουργία PDF. Συγκεκριμένα περιέχει τις μεθόδους για τις λειτουργίες New, Open, Save, Save As, Close, Delete, Status.
Γενικά, ο χρήστης καλείται να δώσει το path σε μορφή string για οποιαδήποτε από τις παραπάνω λειτουργίες. Για το περιεχόμενο των αρχείων, το διαχειριστήκαμε και αυτό ως string, το οποίο επιστρέφεται στο χρήστη από τη μέθοδο Open. Δεδομένου ότι το μέγιστο μήκος ενός string είναι Integer.MAX_VALUE (δηλαδή 2147483647 (2^{31} - 1) μας καλύπτει αρκετά και είναι μικρή έως απίθανη η πιθανότητα υπερχείλισης.
Αξίζει να αναφέρουμε τη χρήση ενός Map<filename, buffered reader> προκειμένου να αντιστοιχίζουμε τα paths των αρχείων με τον buffered reader που χρησιμοποιήθηκε για να γίνει το αντίστοιχο κλείσιμό τους, κατά την Close ή Delete.
Τέλος, κατά την μέθοδο Status, τυπώνουμε στο stdout τις ζητούμενες πληροφορίες και επιστρέφουμε ως string μόνο το μέγεθος του αρχείου, το οποίο θα χρησιμοποιηθεί αργότερα στο GUI.

2. *PDFTool.java*

Υλοποιούμε τη δημιουργία και διαχείριση αρχείων PDF. Συγκεκριμένα, παρέχονται οι μέθοδοι createEmptyPDF, addPage, mergePDF, splitPDF, splitConsecutive και splitTwoParts με τις οποίες υλοποιούμε βασικές λειτουργίες. Σε όλες τις περιπτώσεις

κάνουμε τους κατάλληλους ελέγχους για μη επιτρεπτή είσοδο και έχουμε και τα αντίστοιχα try/catch όπου χρειάζονται.
Γενικά, ο χρήστης καλείται να δώσει το path σε μορφή string για οποιαδήποτε από τις παραπάνω λειτουργίες.
Αξίζει να αναφέρουμε πως στη mergePDF, ο χρήστης, αφού καθορίζει το path του τελικού αρχείου, έπειτα μπορεί να δώσει μεταβλητό μήκος από paths αρχείων τα οποία θα συγχωνευθούν. Περιορισμός αποτελεί το γεγονός ότι τα αρχεία αυτά πρέπει να ανήκουν στο ίδιο directory. Στον κώδικα υπάρχουν αρκετά σχόλια με την λειτουργία των μεθόδων αυτών, για αυτό και δεν αναφέρουμε αναλυτικά τι κάνουμε και πώς.

3. *MlabToPDF.java*

Αποτελεί τη βασική κλάση της βιβλιοθήκης, μέσω της οποίας ένα αρχείο mlab μετατρέπεται σε PDF. Περιέχει τις απαραίτητες μεθόδους για την προσθήκη περιεχομένου στο PDF με βάση τις διάφορες εντολές μορφοποίησης. Επίσης, παρέχονται ως private οι μέθοδοι, μέσω των οποίων γίνεται η αντιστοίχιση των αναγνωριστικών των παραμέτρων με τα Fonts, Font Styles και Font Colors.
Τέλος, έχουμε μια βοηθητική μέθοδο addPDFPage, η οποία εκτελείται την περίπτωση που έχουμε φτάσει στο τέλος μιας σελίδας, δηλαδή στο κάτω επιτρεπτό όριο, και χρειάζεται να δημιουργήσουμε μια επιπλέον σελίδα PDF για την συνέχεια.
Ο έλεγχος για το αν χρειάζεται να αλλάξουμε σελίδα είναι αρκετά συχνός και συγκεκριμένα μετά από κάθε εκτύπωση ολόκληρης γραμμής στο PDF. Για την υλοποίηση της μεθόδου αυτής, χρειαζόμαστε να επιστρέφουμε 2 αντικείμενα(ContentStream, Page), τα οποία είναι αυτά που αναφέρονται στην καινούρια σελίδα(το ContentStream της προηγούμενης σελίδας κλείνει). Δεδομένου ότι δεν είναι εφικτό στην Java η επιστροφή 2 αντικειμένων, υλοποιήσαμε μια βοηθητική κλάση, ReturnPDValues, για να το πετύχουμε.
Προτού αναφέρουμε την κεντρική ιδέα για τη μέθοδο createPDFFromMlab, αξίζει να σημειώσουμε δύο βασικές μεταβλητές της κλάσης, οι οποίες είναι οι **offsetX**, **offsetY**. Αυτές ουσιαστικά προσδιορίζουν τον cursor μέσα στο MediaBox της σελίδας στο οποίο τυπώνουμε το κείμενο. Έτσι, χρειάζονται να ανανεώνονται κατάλληλα. Σημαντικό είναι το γεγονός ότι αρχή του MediaBox θεωρείται η πάνω αριστερή γωνία και κινούμαστε προς την κάτω δεξιά.
Επίσης, ως μεταβλητές κλάσης έχουμε θεωρήσει τις **fontSize**, **font**, **color**, **scaleValue**, οι οποίες αλλάζουν με κάθε εντολή μορφοποίησης που συναντάμε. Είναι, όμως, σταθερές για κάθε κομμάτι text κάτω από κάθε formatting command.
Τέλος, βασική μεταβλητή είναι το **height**, το οποίο καθορίζει κατά πόσο θα μετακινηθεί το offsetY μετά από κάθε drawString(), ώστε να μην συμπέσει η μία γραμμή πάνω στην άλλη καθώς και να έχουν και το κατάλληλο κενό στο ενδιάμεσο.
Σχετικά με τα περιθώρια κάθε σελίδας, έχουμε ορίσει ως στάνταρ το πάνω, κάτω και το αριστερό περιθώριο, ίσα με **margin**. Για το δεξί περιθώριο, έχουμε καθορίζει μια μεταβλητή **maxStringLength**, που προσδιορίζει το μέγιστο μήκος κάθε σειράς στο PDF.

Η βασική ιδέα για τη μέθοδο createPDFFromMlab είναι το διάβασμα μιας γραμμής από το αρχείο mlab, την οποία την σπάμε σε λέξεις. Έπειτα, υπάρχει ο διαχωρισμός, αν η γραμμή αυτή είναι εντολή μορφοποίησης, δηλαδή ξεκινάει με το χαρακτήρα &; ή αν πρόκειται για text. Σημαντικό είναι ότι οι κενές γραμμές του mlab file αγνοούνται. Στην πρώτη περίπτωση, ανάλογα την εντολή καθορίζουμε τη διαμόρφωση του κειμένου, προσθέτοντας αν χρειάζεται π.χ. κάποιο extra κενό καθώς και τις μεταβλητές **fontSize**, **font**, **color**, **scaleValue.** Διαφορετικά, πρόκειται για κείμενο το οποίο τυπώνουμε ανάλογα μετά από τους απαραίτητους ελέγχους κάθε φορά. Η προς εκτύπωση γραμμή στο PDF, η οποία είναι η μεταβλητή **nextLineToDraw** τύπου StringBuffer, κάθε φορά αυξάνεται κατά μία λέξη και ελέγχουμε αν χωράει στην γραμμή.
Στην περίπτωση εικόνας, δίνουμε το τελικό path στο οποίο βρίσκεται στο file system και λόγω του scaleValue, χρειάζεται προσοχή για το αν χωράει σε μια σελίδα PDF τόσο κατά ύψος όσο και κατά πλάτος. Για αυτό το λόγο έχουμε εφαρμόσει τους αντίστοιχους ελέγχους.
Στην περίπτωση εντολής Format, χρειάζονται αρκετοί έλεγχοι όπως φαίνεται και στον κώδικα, αφού πλέον τυπώνουμε στην συνέχεια της προηγούμενης **nextLineToDraw** που τυπώσαμε. Για αυτό χρειαστήκαμε μια βοηθητική μεταβλητή **tmpOffsetX** η οποία μετράει μήκος από την αρχή της γραμμής που βρισκόμαστε.

4. *OnlineContent.java*

Στην κλάση αυτή παρέχονται οι μέθοδοι για τη λήψη περιεχομένου από τη web υπηρεσία OMDb. Η λήψη των πληροφοριών που θέλουμε γίνεται με τη μέθοδο getOnlineContent σχηματίζοντας το κατάλληλο url, σύμφωνα με τη μορφή που προσδιορίζεται στο http://www.omdbapi.com/ . Έτσι, ανοίγοντας μέσω ενός url stream λαμβάνουμε ως αντικείμενο τύπου JSONObject τις ζητούμενες πληροφορίες τις οποίες καταχωρούμε σε ξεχωριστές μεταβλητές. Σε περίπτωση λάθους ή μη εύρεσης πληροφοριών για την ταινία/σειρά που ζητήσαμε, τότε επιστρέφουμε μήνυμα λάθους.
Με τη μέθοδο createMlab δημιουργούμε το ζητούμενο mlab αρχείο, που περιέχει τις πληροφορίες που ζητήσαμε καθώς και τις εντολές μορφοποίησης σύμφωνα με την εκφώνηση.
Τέλος, με τη μέθοδο createPDF δημιουργούμε το PDF βασισμένο στο mlab που φτιάξαμε.
Παρατήρηση : Μερικοί χαρακτήρες ίσως εμφανίζονται λάθος, το οποίο ίσως να επηρεάζει και τη στοίχιση. Αυτό όμως οφείλεται στην πληροφορία που λαμβάνουμε, και όχι σε δικό μας σφάλμα. Π.χ.café .

5. *Main.java*

Για την υλοποίηση του Α΄ μέρους παρέχουμε τη μέθοδο main με εντολές για κάθε(ή έστω τις περισσότερες) μεθόδους που έχουμε υλοποιήσει. Τα έχουμε βάλει σε σχόλια, οπότε ο χρήστης καλείται να το προσαρμόσει ανάλογα.

Β΄ μέρος – GUI – σχόλια

Σε αυτό το κομμάτι υλοποιήσαμε το γραφικό περιβάλλον της εφαρμογής με τη βοήθεια της βιβλιοθήκης Swing. Αρχικά, κάνουμε import το jar της βιβλιοθήκης που δημιουργήσαμε στο Α΄ μέρος, *medialab.jar*. Επίσης, κάνουμε import τα *pdfbox-app-1.8.10.jar* και *java-json.jar* . Στο GUI δεν υλοποιούμε καμία λειτουργικότητα σχετικά με τη διαχείριση των αρχείων mlab και PDF που θα έπρεπε να υλοποιείται στη βιβλιοθήκη. Για το γραφικό περιβάλλον, έχουμε μια κλάση μόνο *Main.java*, η οποία φτιάχνει το GUI. Τα κύρια στοιχεία του είναι τα εξής :

- **MainFrame** που αποτελεί το κεντρικό παράθυρο
- **menuBar** στο πάνω κομμάτι του MainFrame που περιλαμβάνει τα JMenu **menuFile**, **menuEdit**, **menuHelp** και το καθένα περιέχει τα ζητούμενα αντικείμενα JMenuItem .
- **MainPanel** που ενσωματώνεται στο MainFrame και στο οποίο προσθέτουμε τα επόμενα components
- **ToolBar** στο north κομμάτι του MainPanel, που περιλαμβάνει το **buttonsPanel** με buttons που τυπώνουν στο **editorTextArea** τις αντίστοιχες εντολές format
- **filesListPanel** στο west κομμάτι του MainPanel, όπου υλοποιούμε ένα Model-View-Controller(MVC). Συγκεκριμένα, έχουμε ένα **listbox** control, ένα **model** και ένα **fileList** (πρόκειται για ένα Map ανάμεσα στο αρχείο-path του listbox και του κειμένου που εμφανίζεται στο editorTextArea
- **EditorPanel** στο κέντρο του MainPanel, το οποίο περιλαμβάνει το **editorTextArea**
- **labelPanel** στο south κομμάτι του MainPanel, το οποίο περιλαμβάνει το **statusLabel** και **eventLabel**

Δεν θα σχολιάσουμε τον κώδικα, αφού μέσω comments και εκτυπώσεις στο stdout και stderr είναι εμφανές τι υλοποιείται. Μερικές παρατηρήσεις είναι οι εξής :

- Στο filesListPanel εμφανίζουμε όλο το path του αρχείου και όχι μόνο το τελικό όνομα, το οποίο ίσως ήταν πιο βολικό. Αυτό έγινε, λόγω του Map fileList που έχουμε υλοποιήσει. Διαφορετικά, γινόταν αρκετά πολύπλοκη αυτή η υλοποίηση. Με τον τρόπο μας όμως γίνεται ξεκάθαρο ποιο είναι το αρχείο και πού είναι αποθηκευμένο.
- Έχουμε scrollBars τόσο στο filesListPanel όσο και στο EditorPanel για διευκόλυνση.
- Εμφανίζουμε τα κατάλληλα παράθυρα με φίλτρα για το open και το save αρχείων με JFileChooser.
- Εμφανίζουμε τα κατάλληλα μηνύματα με DialogBox σε περιπτώσεις λάθους ή επιβεβαίωσης της ενέργειας από το χρήστη.
- Η εναλλαγή μεταξύ των αρχείων στο listbox δεν κάνει save τα καινούρια δεδομένα που έχουμε γράψει στο editorTextArea. Ο χρήστης πρέπει να κάνει πρώτα save ώστε να αποθηκευτούν στο αντίστοιχο αρχείο.

- Η μέθοδος createPDF φτιάχνει το PDF με βάση το mlab αρχείο και όχι με βάση το περιεχόμενο του editorTextArea. Έτσι, ο χρήστης πρέπει να κάνει save το αρχείο πριν φτιάξει το αντίστοιχο PDF.
- Κατά τη Save As, λαμβάνουμε υπόψη όλα τα δυνατά ενδεχόμενα και συγκεκριμένα, αν το αρχείο υπάρχει στο current directory και ο χρήστης θέλει να το αλλάξει καθώς και αν είναι open.
- Όταν επιλέγουμε Edit→Online PDF, δημιουργούμε το αντίστοιχο PDF με βάση κάποιο mlab αρχείο, το οποίο το θεωρούμε temporary και το διαγράφουμε κατά το exit από την εφαρμογή. Έχουμε σε σχόλια και $2^{ο}$ τρόπο, όπου το tmp αρχείο διαγράφεται αμέσως μόλις δημιουργηθεί το PDF.
- Εμφανίζονται συχνά μηνύματα στο stdout και stderr σε περίπτωση λάθους. Μερικά από αυτά ίσως ήταν χρησιμότερο να είχαμε popup παράθυρα με μηνύματα.
- Πιθανόν να ήταν πιο ευανάγνωστος ο κώδικας αν γινόταν χρήση περισσότερων μεθόδων, όμως θεωρήσαμε πως είναι πιο «μαζεμένος» έχοντας όλο τον κώδικα μαζί για κάθε actionListener.