Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών
HY252 – Αντικειμενοστεφής Προγραμματισμός (Χειμερινό εξάμηνο 2013-2014)
Διδάσκων: **Γιάννης Τζίτζικας**
Βοηθοί: **Παναγιώτης Παπαδάκος**

Γενικές Οδηγίες και Περιγραφή Παραδοτέων της Εργασίας

Φάσεις

- **Φάση Α: 10% τελικού βαθμού**
  - **Τρίτη 2 Δεκεμβρίου - Τετάρτη 18 Δεκεμβρίου**
- **Φάση Β: 15% τελικού βαθμού**
  - **Τετάρτη 18 Δεκεμβρίου - Τετάρτη 22 Ιανουαρίου**

Φάση Α

Σε αυτή τη φάση πρέπει να γίνει ο σχεδιασμός της εφαρμογής βάσει των ιδεών και των αρχών του αντικειμενοστρεφούς προγραμματισμού που έχετε διδαχτεί. Αποτέλεσμα αυτής της φάσης είναι ο καθορισμός των αντικείμενων, των χαρακτηριστικών και της συμπεριφοράς τους που απαιτούνται για να αναπαραστήσουν τις καταστάσεις και τις λειτουργίες του θέματος της εργασίας, όπως έχουν περιγραφεί στην εκφώνηση.

Παραδοτέα αυτής της φάσης είναι :

- **Γραπτή αναφορά** (όχι σε greeklish) η οποία θα περιγράφει τα παραπάνω στοιχεία και θα παρουσιάζει το σχέδιο υλοποίησης της προγραμματιστικής εργασίας, έτσι ώστε να είναι έτοιμο το πέρασμα στην επόμενη φάση της υλοποίησης. Θα πρέπει να συμπεριλαμβάνονται και **UML class diagrams.**
- **Πηγαίος κώδικας** που περιλαμβάνει τις **διεπαφές** (interfaces) και το περίγραμμα των κλάσεων (class outline) Java του προγράμματος σας, συνοδευόμενες από τα απαραίτητα **javadoc** σχόλια, τα οποία θα καθοδηγήσουν την υλοποίηση της επόμενης φάσης.

Επιγραμματικά, οι σημαντικότερες εργασίες που πρέπει να γίνουν σε αυτή τη φάση είναι:

- Αναγνώριση των κλάσεων και διεπαφών για κάθε μικρή και μεγάλη συνιστώσα του προγράμματος. Αναγνώριση των ευθυνών κάθε κλάσης και των πιθανών σχέσεων της με άλλες.
- Εύρεση των χαρακτηριστικών και των μεθόδων κάθε κλάσης.
- Εύρεση της συμπεριφοράς (behaviour) κάθε κλάσης και διεπαφής, καθώς και της επικοινωνίας μέσω μηνυμάτων (method calls) που χρειάζεται να έχουν μεταξύ τους.
- Οργάνωση των κλάσεων σε ιεραρχίες με στόχο την μέγιστη δυνατή επαναχρησιμοποίηση του κώδικα σας.
- Για κάθε κλάση που υλοποιεί μια διεπαφή δώστε τις υπογραφές (signatures) για όλες τις μεθόδους και τις εκ των προτέρων, εκ των υστέρων και αμετάβλητες συνθήκες (preconditions, postconditions, invariants) που τις διέπουν σε μορφή javadoc σχολίων.

Σημειώστε ότι όσο πληρέστερη και αναλυτικότερη δουλειά κάνετε στην σχεδίαση τόσο πιο σωστή και εύκολη θα είναι η υλοποίηση.

Φάση Β

Σε αυτή τη φάση πρέπει να γίνει η κυρίως υλοποίηση της εφαρμογής, βάσει της σχεδίασης που έχει προηγηθεί (φάση Α). Δεν επιβάλλεται να χρησιμοποιηθεί αυτούσια η σχεδίαση της φάσης Α, καθώς κάποιες σχεδιαστικές επιλογές αποδεικνύεται στην πορεία ότι χρειάζονται αναθεώρηση. Εντούτοις, η τελική βαθμολογία θα εξαρτηθεί και από τη συνέπεια της τελικής υλοποίησης ως προς την αρχική σχεδίαση.
Σε αυτή τη φάση, παραδοτέα είναι :

- ο **πηγαίος κώδικας** που υλοποιεί την εργασία
- αναλυτικές οδηγίες για το πώς μεταγλωττίζεται και πώς τρέχει το πρόγραμμά σας (README, Ant, Maven κλπ)
- αναφορά, στην οποία θα αναλύεται :
    - η τελική σχεδίαση της εφαρμογής,
    - ποιές αλλαγές έγιναν σε σχέση με τη σχεδίαση της Α' φάσης (και γιατί),
    - οι αλγόριθμοι που χρησιμοποιήθηκαν
    - τυχόν διαφοροποιήσεις στους κανόνες σε σχέση με τους κανόνες που δίνονται παραπάνω
    - οι σχεδιαστικές και προγραμματιστικές αποφάσεις που ελήφθησαν και πώς αυτό αντανακλάται στον τελικό χρήστη (π.χ. ευκολία/δυσκολία χειρισμού)
    - τα προβλήματα που αντιμετωπίστηκαν
    - τα junit tests που φτιάχτηκαν για τον έλεγχο της ορθότητας
    - γενικά ό,τι άλλο κρίνετε απαραίτητο να αναφερθεί

Βαθμολογία Εργασίας

Για τη βαθμολογία της εργασίας σας θα συνεκτιμηθούν:

- εάν (και πόσο) η σχεδίαση της εφαρμογής εφαρμόζει τις έννοιες και τεχνικές του αντικειμενοστρεφούς προγραμματισμού που διδαχθήκατε στο μάθημα
- εάν (και πόσο) υλοποιήθηκαν οι υποχρεωτικές λειτουργίες της εφαρμογής
- η πληρότητα της τελικής αναφοράς, η οποία θα καταγράφει και θα τεκμηριώνει την σχεδίαση και υλοποίηση της εφαρμογής.

Για διευκρινήσεις σχετικά με την παραπάνω εργασία μπορείτε να στέλνετε μηνύματα με απορίες σας στο σχετικό forum στην ιστοσελίδα του moodle. Ερωτήσεις που στέλνονται στην λίστα του μαθήματος hy252-list@csd.uoc.gr **δε θα απαντώνται**.

Καλή Εργασία

Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών
HY252 – Αντικειμενοστεφής Προγραμματισμός (Χειμερινό εξάμηνο 2013-2014)
Διδάσκων: **Γιάννης Τζίτζικας**
Βοηθοί: **Παναγιώτης Παπαδάκος**

project 

Εκπαιδευτικοί Στόχοι

Προδιαγραφή και σχεδίαση συστήματος
Προδιαγραφή Αφαιρετικών Τύπων Δεδομένων (ΑΤΔ) που απαιτούνται για την επιτυχή ολοκλήρωσή του συστήματος
Υλοποίηση εξαρτημάτων (ΑΤΔ) του συστήματος των οποίων η προδιαγραφή δίνεται
Δημιουργία Γραφικής Διεπαφής
Απεξάρτηση του πυρήνα του συστήματος από τη Γραφική Διεπαφή
Επαναχρησιμοποίηση διεπαφών και κλάσεων
Χρήση JFC (Java Collection Framework)
Χρήστη νημάτων (threads)
Τεκμηρίωση, Έλεγχος

Σημείωση: Δεν απαιτείται να έχετε καμία γνώση μουσικής.

Σύντομη Περιγραφή Εργασίας

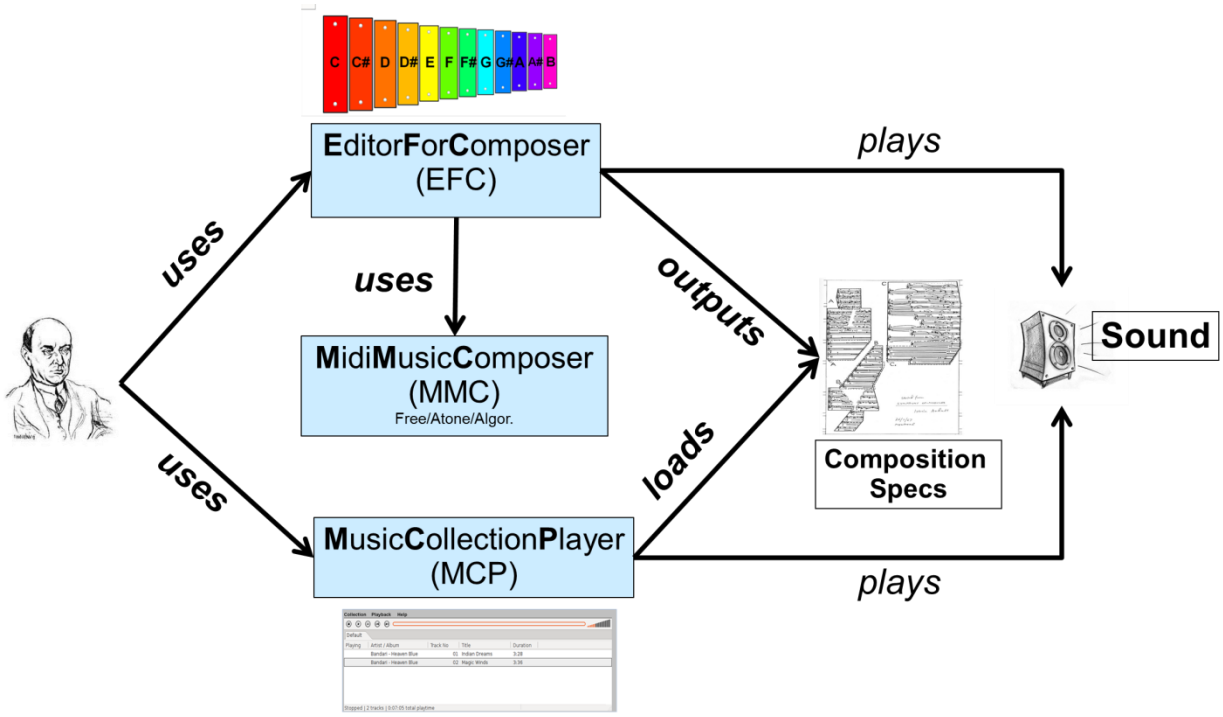
Στόχος. Σκοπός της εργασίας είναι να φτιάξετε έναν συνθέτη MIDI μουσικής και ένα Σύστημα για αναπαραγωγή των κομματιών.

Έναρξη. Με τη βιβλιοθήκη jfugue που σας δίνεται μπορείτε να δημιουργήσετε και να ακούσετε MIDI μουσική. Για παράδειγμα με τον κώδικα:

```
Player player = new Player();
player.play("C D E");
```

θα ακούσετε το Ντο Ρε Μι, ενώ αν αντί για “C D E” βάλετε το "E7 D7 E7 F7 D7 C7 D7 E7 E7 D7 E7 D7 C7i D7i B6 A6" θα ακούσετε ένα τμήμα από το Χαμόγελο της Τζοκόντα του Χατζηδάκη (πληροφορίες για τις δυνατότητες της jfugue μπορείτε να βρείτε στο http://www.jfugue.org/ και στο www.jfugue.org/jfugue-chapter2.pdf).

Βασικά Εξαρτήματα. Καλείστε να σχεδιάσετε και να υλοποιήσετε τρία βασικά εξαρτήματα (ή υποσυστήματα). Ο ρόλος τους κάθε ενός περιγράφεται αδρομερώς στην Εικόνα 1.



Εικόνα 1

A/ MIDIMusicComposer (MMC): Εξάρτημα το οποίο θα παράγει MIDI strings (που εν συνεχεία θα μπορεί να παίξει ο Player που παρέχει η βιβλιοθήκη jfugue). Ουσιαστικά θα προσφέρει λειτουργίες που διαχειρίζονται και εν τέλει παράγουν *συμβολοσειρές* που συμβολίζουν ακολουθίες από μουσικές νότες που μπορεί να ηχοποιήσει η βιβλιοθήκη jfugue. Θα υποστηρίζει έναν ελεύθερο τρόπο σύνθεσης (όπου ο χρήστης θα εισάγει κάθε νότα της σύνθεσης), αλλά και «αλγοριθμική μουσική»:

- Μια περίπτωση αλγοριθμικής μουσικής που πρέπει να υποστηρίζεται είναι η **ατονική μουσική.** Ο χρήστης του εξαρτήματος (συνθέτης αλγοριθμικής μουσικής) μέσω διαφόρων παραμέτρων ελέγχου θα προσδιορίζει τις ακολουθίες που θέλει να παραχθούν.
- Μια άλλη περίπτωση αλγοριθμικής μουσικής που πρέπει να υποστηρίζεται είναι η **<<AM>>μουσική** (όπου AM ο … αριθμός μητρώου σας), δηλαδή ένας δικός σας τρόπος αλγοριθμικής παραγωγής μουσικής.

B/ EditorForComposers (EFC): Ο ρόλος αυτού του εξαρτήματος είναι να κάνει πιο εύκολη την σύνθεση μουσικής από το χρήστη προσφέροντας μια γραφική διεπαφή για τις παραμέτρους ελέγχου που προσφέρει ο MMC. Επίσης θα προσφέρει τη δυνατότητα ελεύθερης σύνθεσης με .. ξυλόφωνο.

C/ MusicCollectionPlayer (MCP): Εξάρτημα με γραφική διεπαφή το οποίο υποστηρίζει την έννοια της μουσικής συλλογής και επιτρέπει την εκτέλεσή της. Μία συλλογή αποτελείται από κομμάτια, αυτά που έχει κάποιος συνθέσει μέσω του EFC. Ο χρήστης θα μπορεί να δει τα στοιχεία της συλλογής, να προσθέσει ή να διαγράψει στοιχεία, ή να αλλάξει τη σειρά τους. Επίσης θα πρέπει να μπορεί να εκτελέσει τα κομμάτια της συλλογής.

Αναλυτική Περιγραφή της Εργασίας

A/ MIDIMusicComposer (MMC) (20% του βαθμού)

Α.1/ Ελεύθερη Σύνθεση Μουσικής Δημιουργία μίας νέας ελεύθερης σύνθεσης. Ουσιαστικά πρόκειται για μία συμβολοσειρά από νότες.

Α.2/ Ατονική Μουσική (Δωδεκαφθογγισμός) Σχετικά με την ατονική μουσική, μια σύνθεση προσδιορίζεται δίνοντας μια σειρά από 12 διαφορετικές νότες, και πράξεις που «πολλαπλασιάζουν με ειδικό τρόπο» αυτή τη σειρά των 12 νοτών (δείτε το σχετικό παράρτημα). Για παράδειγμα η σειρά από 12 νότες μπορεί να είναι η **<C, C#, D, D#, E, F, F#, G, G#, A, A#, B>** και το μουσικό κομμάτι να ορίζεται από την έκφραση: **Retrograde, Transpose(1), doNothing, Reflect(0).** Ενδεικτικά η παραπάνω έκφραση θα δημιουργήσει την εξής σειρά από 192 νότες:

C C# D D# E F F# G G# A A# B
B A# A G# G F# F E D# D C# C
C# D D# E F F# G G# A A# B C
C B A# A G# G F# F E D# D C#
C C# D D# E F F# G G# A A# B
B A# A G# G F# F E D# D C# C
C# D D# E F F# G G# A A# B C
C B A# A G# G F# F E D# D C#
C B A# A G# G F# F E D# D C#
C# D D# E F F# G G# A A# B C
B A# A G# G F# F E D# D C# C
C C# D D# E F F# G G# A A# B
C B A# A G# G F# F E D# D C#
C# D D# E F F# G G# A A# B C
B A# A G# G F# F E D# D C# C
C C# D D# E F F# G G# A A# B

Α.3/ Η Δική σας Μουσική. Εκτός της ατονικής μουσικής μπορείτε να επινοήσετε και έναν δικό σας τρόπο. Για το σκοπό αυτό μπορεί να βοηθήσει το γεγονός ότι μπορείτε να προσδιορίσετε μια νότα και με αριθμό (από το 0 έως το 127). Οι αντιστοιχήσεις φαίνονται παρακάτω.

Octave	C	C#/D*b*	D	D#/E*b*	E	F	F#/G*b*	G	G#/A*b*	A	A#/B*b*	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	70
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

Εικόνα 2

Με `player.play("[81] [83]")` θα ακούσετε τις αντίστοιχες νότες.
Άρα κάποιος μπορεί να σκεφτεί άπειρους τρόπους αλγοριθμικής παραγωγής μουσικής. Π.χ.

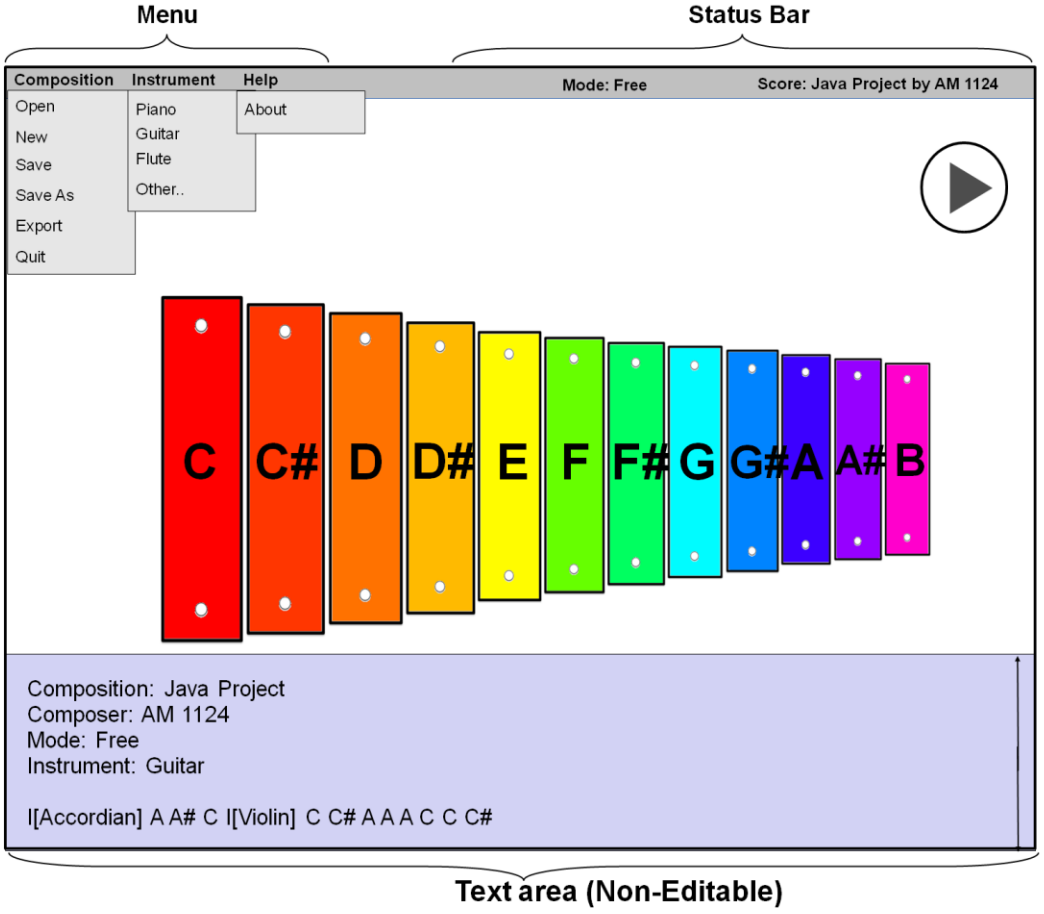
- Τυχαία επιλογή αριθμών από το 0 έως το 127
- Ο συνθέτης δίνει ως είσοδο ένα κείμενο και οι νότες προκύπτουν αν για κάθε γράμμα του κειμένου υπολογίσουμε το υπόλοιπο της διαίρεσής του με το 128
- Με διάφορα εμφωλευμένα loops, …

B/ EditorForComposers (EFC) (40% του βαθμού)

Ο ρόλος του συντάκτη (editor) είναι να κάνει πιο εύκολη τη σύνθεση αλγοριθμικής ή μη μουσικής προσφέροντας μια γραφική διεπαφή για τις παραμέτρους ελέγχου του MMC. Πρέπει να υποστηρίζονται οι γενικές λειτουργίες: Δημιουργία νέας σύνθεσης (Composition>New), όπου ο συνθέτης επιλέγει το είδος σύνθεσης που θέλει να κάνει και το default μουσικό όργανο, και άρα το UI προσαρμόζεται ανάλογα, άνοιγμα σύνθεσης από αρχείο (Composition>Open) για αυξητική τροποποίησή της, και αποθήκευση σύνθεσης σε αρχείο (Composition>Save και Composition>Save As). Ο τρόπος υποστήριξης των τριών τρόπων σύνθεσης περιγράφεται παρακάτω. Σε κάθε περίπτωση ο συντάκτης πρέπει να επιτρέπει στο συνθέτη να ακούσει άμεσα την υπο κατασκευή σύνθεσή του. Επίσης πρέπει να προσφέρεται η δυνατότητα για επιλογή των επιθυμητών οργάνων (από αυτά που υποστηρίζει η jfugue), και ότι άλλο κρίνετε ότι χρειάζεται και σας ικανοποιεί ως … συνθέτες **(επιπλέον λειτουργικότητα από αυτή που περιγράφεται θα ληφθεί υπόψιν σαν bonus**).

B.1/ Ελεύθερη Σύνθεση: Για σύνθεση μουσικής με ελεύθερο τρόπο. Οι νότες δίνονται πατώντας πάνω στο ξυλόφωνο. Κάθε φορά που ο χρήστης πατάει πάνω σε μία νότα ακούγεται η νότα που πάτησε μέσω της jfugue και εισάγεται η νότα στο Composition. Το TextArea του Composition είναι non-editable και scrollable. Εμφανίζει όλες τις πληροφορίες που σχετίζονται με το συγκεκριμένο composition, και ενημερώνεται καθώς ο χρήστης πατάει νέες νότες στο ξυλόφωνο. Συνάμα ο χρήστης μπορεί μέσω του μενού Instrument να επιλέξει το τρέχον όργανο. Όταν πατάει μια νότα στο ξυλόφωνο, η νότα θα αναπαράγεται βάσει του οργάνου επιλογής.

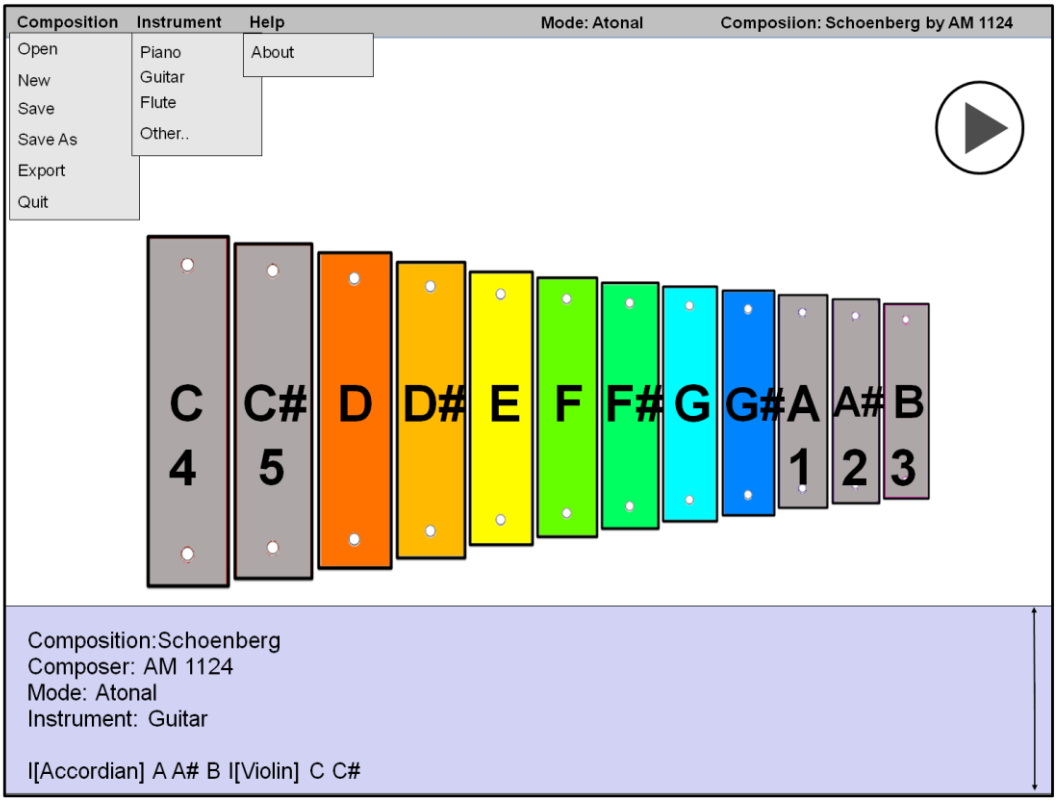
Επιπλέον, όταν πατιέται το κουμπί ► (Play) θα εκτελείται μουσικά η τρέχουσα σύνθεση, και το εικονίδιο του κουμπιού θα γίνεται ■ (Stop). Με πάτημα του ■ (Stop) θα σταματάει η εκτέλεση και το εικονίδιο θα ξαναγίνεται ►(Play). Μία ενδεικτική γραφική διεπαφή δίνεται παρακάτω.



Εικόνα 3

Β.2/ Συνθέτοντας Ατονική Μουσική (Δωδεκαφθογγισμός): Για σύνθεση ατονικής μουσικής ο γραφικός συντάκτης πρέπει να επιτρέψει σε κάποιον να επιλέξει (και να τροποποιήσει) με εύχρηστο τρόπο τη σειρά των 12 φθόγγων και τις «πολλαπλασιαστικές πράξεις».

Για την επιλογή των 12 φθόγγων μπορεί να χρησιμοποιηθεί πάλι το ξυλόφωνο με την εξής όμως διαφορά: όποτε πατιέται ένα πλήκτρο αυτό θα γίνεται ανενεργό (διότι κάθε νότα πρέπει να εμφανίζεται μόνο μία φορά) και θα εμφανίζεται το νούμερό του στην ατονική σειρά. Στην Εικόνα 3 φαίνεται ότι ο χρήστης έχει πατήσει 5 πλήκτρα τα οποία έχουν γίνει ανενεργά, με την συγκεκριμένη σειρά.

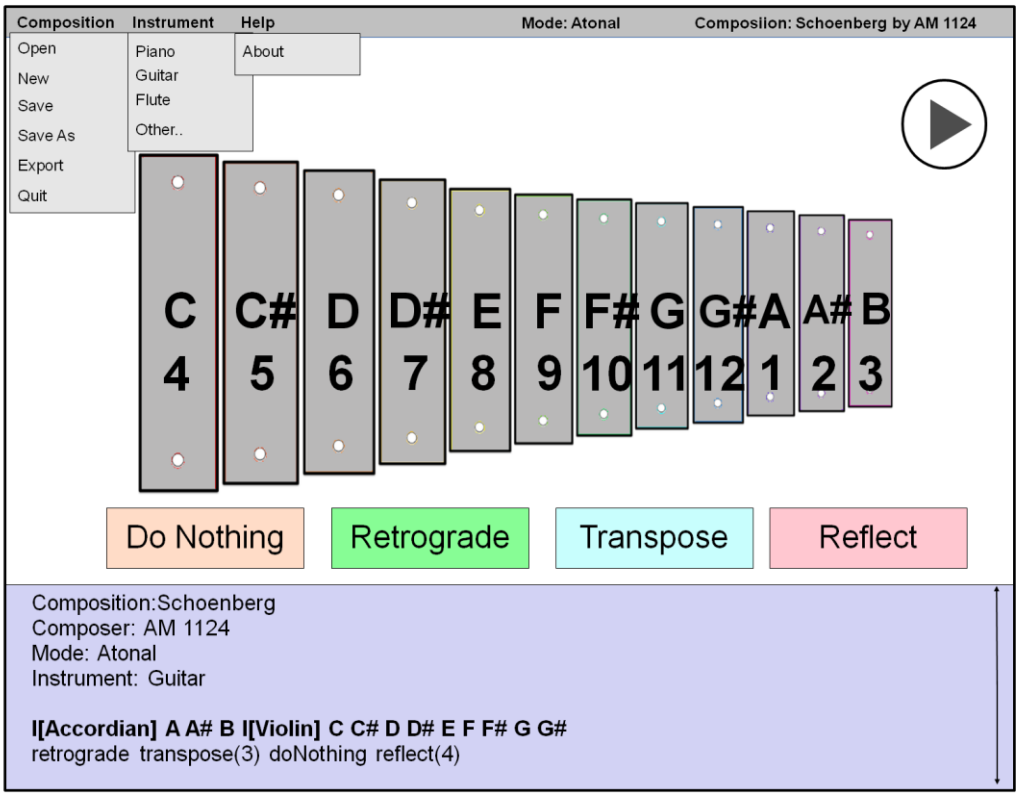


Εικόνα 4

Όταν ο χρήστης δώσει τη σειρά, τότε θα εμφανιστούν οι πολλαπλασιαστικές πράξεις. Συγκεκριμένα χρειάζεται να υπάρχουν τα εξής τέσσερα κουμπιά (Εικόνα 5).

- doNothing: Αναπαράγει τη σειρά
- Retrograde: Αναπαράγει τη σειρά ανάποδα
- Transpose: Εδώ ο χρήστης πρέπει να δώσει σαν είσοδο τα βήματα ολίσθησης, δηλαδή έναν αριθμό από το 1 έως το 11 . Η είσοδος του χρήστη μπορεί να ληφθεί μέσω JOptionPane.
- Reflection: Εδώ ο χρήστης πρέπει να δώσει σαν είσοδο έναν αριθμό από το 0 έως το 11 (είναι η κορυφή από την οποία περνάει ο άξονας της ανάκλασης). Ξανά η είσοδος του χρήστη μπορεί να ληφθεί μέσω JOptionPane

Παραπάνω πληροφορίες για τις πράξεις δίνονται στο παράρτημα.



Εικόνα 5

Β.3/ Συνθέτοντας τη Δική σας Μουσική: Ως προς το δικό σας τρόπο παραγωγής μουσικής, ο συντάκτης θα πρέπει να προσφέρει την αντίστοιχη υποστήριξη (π.χ. ο συντάκτης δίνει κείμενο το οποίο γίνεται μουσική βάσει του αλγόριθμου σας).

Γ/ MusicCollectionPlayer (MCP) (40% του βαθμού)

Εξάρτημα με γραφική διεπαφή το οποίο υποστηρίζει την έννοια της *μουσικής συλλογής* και την *αναπαραγωγή* της (μουσική εκτέλεση). Μία συλλογή αποτελείται από μία γραμμική σειρά από κομμάτια. Πρέπει να υποστηρίζονται τα κομμάτια που παράγει ο EFC, δηλαδή να μπορεί να διαβάσει τα αρχεία που αποθήκευσε ο Editor.

Συγκεκριμένα πρέπει να υποστηρίζονται λειτουργίες διαχείρισης μουσικής συλλογής (Collection>New Collection, Collection> Add File, Collection> Add Folder, Collection>Open, Collection>Save). Συγκεκριμένα, δημιουργώντας ο χρήστης μία νέα λίστα, θα μπορεί να δώσει όνομα (και ότι άλλα στοιχεία επιθυμεί) και στη συνέχεια να τη σώσει σε ένα αρχείο ή να τη φορτώσει από το σκληρό. Επιπλέον θα μπορεί να προσθέσει (Add) μουσικά κομμάτια επιλέγοντας τα αντίστοιχα αρχεία (π.χ. μέσω JFileChooser) από το δίσκο (αυτά που έχει γράψει ο editor), είτε ένα ένα, είτε δίνοντας ένα φάκελο που περιέχει τέτοιου είδους αρχεία.

Ο χρήστης θα μπορεί να κάνει διάφορες ενέργειες πάνω στα αρχεία της λίστας που έχουν φορτωθεί, μέσω ενός μενού που θα εμφανίζεται με δεξί click πάνω από κάθε αρχείο. Οι ενέργειες που θα μπορεί να κάνει ο χρήστης μέσω του menu είναι η αύξηση ή μείωση της θέσης του στην διατεταγμένη λίστα, καθώς και η αφαίρεση του κομματιού από τη λίστα. **Επιπλέον αν υποστηρίζεται η αλλαγή της σειράς των κομματιών μέσω drag \& drop θα δοθεί κάποιο bonus.**

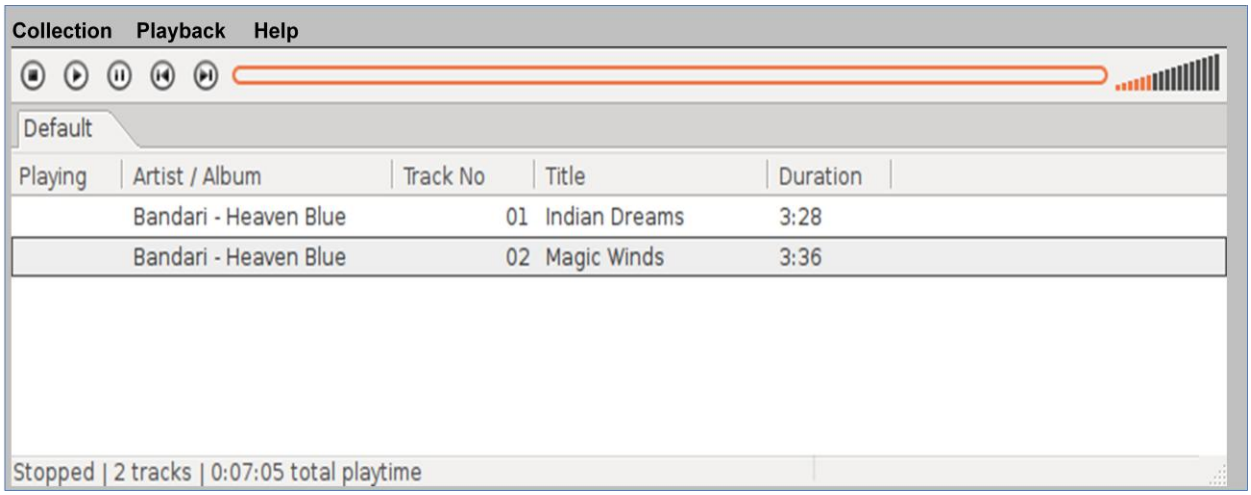
Συνάμα, πέρα από την εκτέλεση ενός κομματιού από το menu, ο χρήστης θα μπορεί μέσω των κατάλληλων κουμπιών να εκτελέσει μουσικά τη συλλογή (κουμπιά play/pause/stop/previous/next). Ο χρήστης πρέπει να μπορεί να παύσει προσωρινά ή οριστικά την εκτέλεση, να βλέπει το κομμάτι που εκείνη τη στιγμή εκτελείται, ή να κάνει κλικ σε ένα κομμάτι πριν ή μετά από αυτό που παίζει και η αναπαραγωγή να αρχίζει από εκείνο το σημείο. Επιπλέον θα πρέπει να δίνονται οι διάφορες πληροφορίες που έχει κάθε composition (π.χ. όνομα σύνθεσης, συνθέτης, αριθμός από νότες, κτλ.)

Επιπλέον ο χρήστης θα μπορεί μέσω του menu Playback, να ορίσει τις παραμέτρους της αναπαραγωγής. Για παράδειγμα θα μπορεί να ορίσει ότι τα κομμάτια θα αναπαράγονται σε τυχαία σειρά αντί της default γραμμικής σειράς, καθώς και να δηλώσει ότι η όλη λίστα ή ένα κομμάτι θα αναπαράγεται επ' αόριστο.

Ενδεικτικά σας δίνεται το παρακάτω UI.

Προαιρετικά θα μπορούσατε να υποστηρίξετε τη μπάρα αναπαραγωγής η οποία θα εκτιμηθεί σαν bonus για τον τελικό βαθμό.

Για τις ανάγκες του MusicCollectionPlayer ίσως χρειαστεί να έχετε πάνω από ένα νήμα (thread) εκτέλεσης. Διαβάστε την αντίστοιχη ενότητα από το "Γεύση Java".


Collection
Playback
Help
Default
Playing
Artist / Album
Track No
Title
Duration
Bandari - Heaven Blue
01
Indian Dreams
3:28
Bandari - Heaven Blue
02
Magic Winds
3:36
Stopped | 2 tracks | 0:07:05 total playtime

Υποδείξεις για Σχεδίαση/Υλοποίηση

Κατά τη διάρκεια της σχεδίασης των παραπάνω εφαρμογών (και εν γένει οποιουδήποτε λογισμικού ή τεχνικού έργου), επιδιώκεται η αποσύνθεση του συστήματος σε μικρότερα τμήματα, με στόχο την ανάθεση σαφώς ορισμένων και καθορισμένων αρμοδιοτήτων σε κάθε τμήμα και την επικύρωση ότι όλα τα τμήματα μαζί επιτυγχάνουν τους σκοπούς του συστήματος. Επομένως, η σχεδίαση είναι μια διαδικασία επίλυσης και κατακερματισμού του αρχικού προβλήματος σε επιμέρους μικρότερα και ευκολότερα επιλύσιμα υπο-προβλήματα που θα ικανοποιούν τις λειτουργικές απαιτήσεις και θα υπόκεινται σε συγκεκριμένες αρχές καλής σχεδίασης.

Μια τέτοια αρχή αποτελεί η αποσύνδεση του μοντέλου (model), που περιγράφει τα δεδομένα, τη συμπεριφορά τους και το σύνολο των κανόνων που τα διέπει, από την απεικόνιση τους (view). Βασικός στόχος μιας τέτοιας αποσυνδεδεμένης προσέγγισης είναι η ελαχιστοποίηση των απαιτούμενων επεμβάσεων σε κώδικα που μπορούν να έχουν μελλοντικές αλλαγές είτε στο μοντέλο είτε στο τρόπο/μέσο απεικόνισης. Αυτό οδηγεί σε καλύτερη ποιότητα κώδικα, με μικρότερο κόστος συντήρησης, επέκτασης και επαναχρησιμοποίησης. Για να γίνει πιο κατανοητό αυτό θεωρήστε το ακόλουθο σενάριο. Υποθέστε ότι αρχικά έχετε σχεδιάσει μια εφαρμογή που για διάφορούς λόγους εκτυπώνει τα αποτελέσματα της στην κονσόλα. Έχοντας ακολουθήσει μια τέτοια αρχιτεκτονική 'χαλαρής' σύνδεσης η μετάβαση σε ένα γραφικό παραθυρικό περιβάλλον γίνεται ομαλά, απλώς τροποποιώντας κατάλληλα (επεκτείνοντας) τις διαδικασίες εκείνες που ήταν υπεύθυνες για την αποτύπωση του μοντέλου στην κονσόλα.

Πιο συγκεκριμένα, σύμφωνα με το MVC πρότυπο θα πρέπει να διαχωρίζεται η ανάπτυξη της γραφικής διεπαφής των εφαρμογών (View) από τον πυρήνα που περιέχει όλη την πληροφορία κατάστασης (Model) και από τον μηχανισμό διαχείρισης και ενημέρωσης των ενεργειών της εφαρμογής με τη γραφική του απεικόνιση (Controller). Πιο συγκεκριμένα το Model μπορεί να θεωρηθεί ότι αποτελείται από οτιδήποτε σχετίζεται με τα δεδομένα της εφαρμογής. Υπό αυτήν την έννοια οι «Compositions», οι «MMC», και οι «Collections» αποτελούν μέρος του μοντέλου της εκάστοτε εφαρμογής καθώς περιγράφουν τα εκάστοτε δεδομένα που καλείται να διαχειριστεί ο Controller κάθε εφαρμογής.

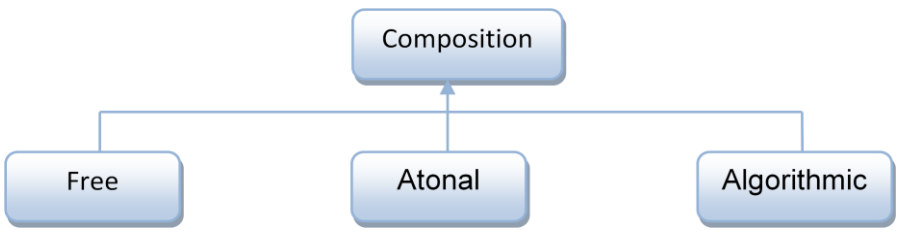
Ο Controller επιφορτίζεται με τη διαχείριση της αλληλεπίδρασης της γραφικής διεπαφής με το μοντέλο. Ο Controller έχει πρόσβαση στα δεδομένα της εφαρμογής και στην διεπαφή. Από την διεπαφή λαμβάνει πληροφορία για τις ενέργειες του συνθέτη και βάσει αυτών θα πρέπει να ανανεώσει το μοντέλο. Αντίστοιχα η ανανέωση του μοντέλου (π.χ. φόρτωμα μιας Collection) θα πρέπει να χρησιμοποιείται από τον Controller για την ανανέωση της διεπαφής.

Model

Παρακάτω σας δίνονται οι βασικές κλάσεις για το Model κάθε εφαρμογής.

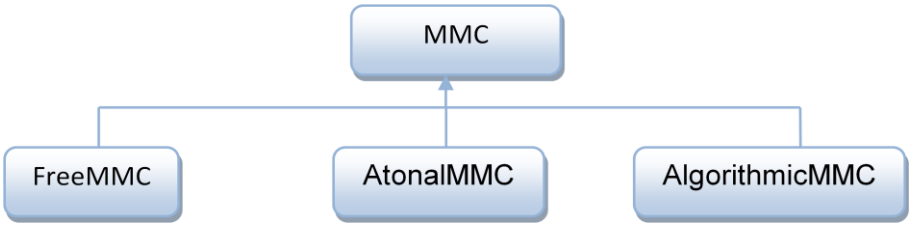
MMC (MidiMusicComposer)

Μια βασική οντότητα/κλάση του MMC είναι η abstract κλάση Composition. Η κλάση αυτή μοντελοποιεί μια σύνθεση και ορίζει τόσο τα γνωρίσματα της (μεταβλητές στιγμιότυπων, πχ νότες, όργανα, όνομα σύνθεσης, συνθέτης, κτλ.), όσο και τις χρήσιμες μεθόδους της (πχ μέθοδο πρόσβασης στις νότες της σύνθεσης). Χρησιμοποιώντας αυτή τη κλάση θα πρέπει να υποστηρίξετε τα 3 είδη συνθέσεων (free, atonal, algorithmic).



EFC (EditorForComposers)

Αντίστοιχα για τον EFC σημαντική οντότητα/κλάση είναι η abstract κλάση MMC, η οποία θα κάνει generate τα αντίστοιχα Compositions.



MCP (Music Collection Player)

Η βασική οντότητα/κλάση του MPC είναι η κλάση Collection. Κάθε Collection έχει ένα όνομα και αποτελείται από μία διατεταγμένη σειρά από Compositions, καθώς και το path τους στο δίσκο.

Σας δίνονται

- Συμπιεσμένο αρχείο με:
  - Βιβλιοθήκη jfugue και απλό πρόγραμμα που τη χρησιμοποιεί
  - Οι υπογραφές για του ΑΤΔ του δωδεκαφθογγισμού

Βοηθητικά

Παρακάτω σας δίνονται κάποια ενδεικτικά links τα οποία μπορούν να σας βοηθήσουν στην υλοποίηση:
***Ενδεικτικό βοήθημα για το MVC μοντέλο σε Java.**
http://www.newthinktank.com/2013/02/mvc-java-tutorial/
***Ενδεικτικό βοήθημα για threads σε Java.**
http://oreilly.com/catalog/expjava/excerpt/index.html
***Ενδεικτικό βοήθημα για Drag & Drop.**
http://www.dreamincode.net/forums/topic/209966-java-drag-and-drop-tutorial-part-1-basics-of-dragging/

Καλή εργασία

ΠΑΡΑΡΤΗΜΑΤΑ

Παράρτημα Α: Ατονική Μουσική: Δωδεκαφθογγισμός

Ο δωδεκαφθογγισμός είναι ένας τρόπος σύνθεσης μουσικής που δεν βασίζεται σε κάποια διατονική κλίμακα και αναπτύχθηκε στις αρχές του 1900 από τον Άρνολντ Σένμπεργκ. Η συγκεκριμένη μέθοδος στοχεύει στην οργάνωση της ατονικότητας μέσω της χρήσης μίας επιλεγόμενης διαδοχής των 12 φθόγγων <C, C#, D, D#, E, F, F#, G, G#, A, A#, B> (χρησιμοποιούμε την ορολογία του αγγλόφωνου κόσμου, όπου το A αντιστοιχεί στο Λα, το B στο Σι, το C στο Ντο, κ.ο.κ.). Η εκάστοτε διαδοχή των φθόγγων που επιλέγεται από τον συνθέτη ονομάζεται «δωδεκάφθογγη σειρά» ή απλώς «σειρά» και αποτελείται από 12 φθόγγους (δηλαδή κάθε σειρά περιλαμβάνει και τους 12 φθόγγους μία μόνο φορά) π.χ. η σειρά: <A, B, C, D, E, F, G, A#, C#, D#, F#, G#>.

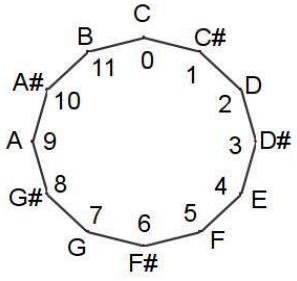
Για τις ανάγκες της εργασίας καλείστε να υλοποιήσετε τα εξής.

Κλάση AtonalRow: Ένα στιγμιότυπο της κλάσης είναι μία ατονική σειρά, άρα μια ακολουθία από φθόγγους (που είναι έγκυρη αν έχει 12 φθόγγους και κανένα διπλότυπο). Η κλάση **AtonalRow** πρέπει να υλοποιεί το interface **Symmetry** που σας έχει δοθεί.

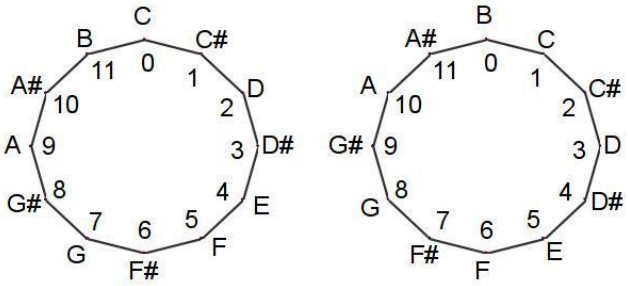
Κλάση AtonalComposition: Ένα στιγμιότυπο της είναι στην ουσία ένα μουσικό κομμάτι, άρα μια μουσική σύνθεση. Είναι στην ουσία μία ακολουθία από AtonalRows. Προσοχή, η AtonalComposition, για εξοικονόμηση χώρου, δεν αποθηκεύει όλες τις νότες, παρά μόνο την αρχική seed AtonalRow που δίνει ο χρήστης και τα συγκεκριμένα actions που εφαρμόζει σε αυτή καθώς και τη σειρά τους. Υλοποιεί το interface **Symmetry** που σας έχει δοθεί.

Σημασιολογία Πράξεων Συμμετρίας Ατονικής Γραμμής (AtonalRow)

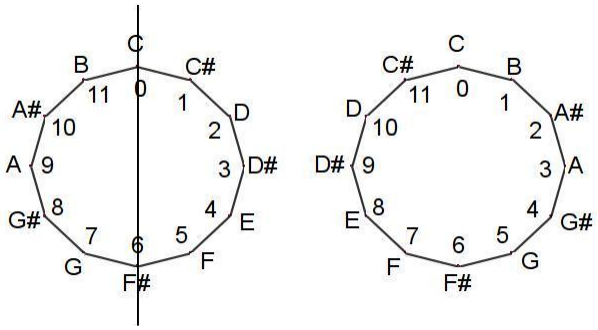
- **doNothing:** Η συγκεκριμένη ενέργεια δεν κάνει τίποτα, άρα αφήνει ανέπαφη τη γραμμή.
- **retrograde**: Αντιστρέφει τη γραμμή, π.χ. αν έχουμε τη γραμμή “A A# B C C# D D# E F F# G G#” με την οπισθοδρόμηση θα πάρουμε τη γραμμή “G# G F# F E D# D C# C B A# A”.
- **transpose(x)**: Στην ουσία γίνεται μια ολίσθηση. Δείτε το παρακάτω δωδεκάγωνο στις κορυφές του οποίου έχουμε τοποθετήσει τους 12 μουσικούς φθόγγους



Κατά την πράξη της **μεταφοράς** γίνεται περιστροφή του πολυγώνου κατά x κορυφές σύμφωνα με τη φορά των δεικτών του ρολογιού και αντιστοίχηση στις νέες νότες. Για παράδειγμα σε μία μεταφορά κατά μία κορυφή, η νότα C θα αντιστοιχηθεί στη B, η C στη C#, κ.ο.κ.



- **reflect(x)**: Ανάκλαση του πολυγώνου γύρω από τον άξονα που ορίζεται από ένα φθόγγο x και τον απέναντί του ((x+6) % 12). Το παρακάτω παράδειγμα δείχνει την ανάκλαση πάνω από τον άξονα που περνάει από τις κορυφές 0 και 6. Άρα η νότα C παραμένει C, η C# γίνεται B#, η D γίνεται A# κ.ο.κ.



Σημασιολογία Πράξεων Συμμετρίας Ατονικής Σύνθεσης (AtonalComposition)

Η σημασιολογία των πράξεων συμμετρίας στην AtonalComposition είναι διαφορετική (σε σχέση με την AtonalRow). Κάθε πράξη συμμετρίας πάνω σε ένα AtonalComposition αφήνει ανέπαφες τις ήδη υπάρχουσες σειρές και δημιουργεί τον αντίστοιχο αριθμό από AtonalRows εφαρμόζοντας την αντίστοιχη πράξη συμμετρίας πάνω στις ήδη υπάρχουσες AtonalRows.

doNothing(): Επεκτείνει την τρέχουσα ακολουθία από AtonalRows προσθέτοντας στο τέλος της αντίτυπα τους με την ίδια σειρά. Π.χ. αν ενα AtonalComposition περιλαμβάνει τις AtonalRows a και b με τη σειρά <a b>, τότε μετά τη κλήση της doNothing() η AtonalScore θα περιλαμβάνει τις σειρές <a b a b>.
retrograde(): Επεκτείνει την τρέχουσα ακολουθία από AtonalRows προσθέτοντας στο τέλος της αντίτυπά τους με την ανάποδη σειρά, αφού έχει εφαρμόσει σε κάθε σειρά την αντίστοιχη retrograde συμμετρία. Άρα, αν η AtonalComposition περιλαμβάνει τις a και b με τη σειρά <a b>, τότε μετά τη κλήση της retrograde(), η AtonalComposition θα περιλαμβάνει τις σειρές <a b b.retrograde() a.retrograde()>.
transpose(x): Επεκτείνει την τρέχουσα ακολουθία από AtonalRows προσθέτοντας στο τέλος της αντίτυπά τους με την ίδια σειρά, αφού έχει εφαρμόσει σε κάθε σειρά την αντίστοιχη transpose συμμετρία. Άρα, αν η AtonalComposition περιλαμβάνει τις a και b με τη σειρά <a b>, τότε μετά τη κλήση της transpose(1) η AtonalScore θα περιλαμβάνει τις σειρές <a b a.transpose(1) b.transpose(1)>.
reflect(x): Επεκτείνει την τρέχουσα ακολουθία από AtonalRows προσθέτοντας στο τέλος της αντίτυπά τους με την ίδια σειρά, αφού έχει εφαρμόσει σε κάθε σειρά την αντίστοιχη reflect συμμετρία. Άρα, αν η AtonalComposition περιλαμβάνει τις a και b με τη σειρά <a b>, τότε μετά τη κλήση της reflect(0) η AtonalComposition θα περιλαμβάνει τις σειρές <a b a.reflect(0) b.reflect(0)>.