ΗΥ-252 Αντικειμενοστραφής Προγραμματισμός

Χειμερινό Εξάμηνο 2012
Διδάσκων: Χριστοφίδης Βασίλης

2η Σειρά Ασκήσεων

Ημερομηνία Παράδοσης: 16/11/2012

Παρακάτω σας δίνονται οι ορισμοί τεσσάρων διαφορετικών Αφαιρετικών Τύπων Δεδομένων (ΑΤΔ). Για όλες τις λειτουργίες (operations) αυτών των ΑΤΔ, δώστε τις υπογραφές (signatures), το είδος τους ανάλογα με την επίδραση που έχουν στην κατάσταση των αντικειμένων (**constructors, accessors, transformers**), και τις εκ των προτέρων, τις εκ των υστέρων και τις αμετάβλητες συνθήκες (**preconditions**, **postconditions, invariants)**. Στη συνέχεια, υλοποιήστε τους ΑΤΔ χρησιμοποιώντας κλάσεις **Java**. Χρησιμοποιήστε τους μηχανισμούς των **JUnit Tests** της Java για να ελέγξετε την ορθότητα του κώδικά σας. Τεκμηριώστε τον κώδικα σας με την βοήθεια των **Javadocs**. Για κάθε άσκηση, πέρα από τον πηγαίο κώδικα που υλοποιεί τους ΑΤΔ που σας ζητούνται, θα πρέπει να συμπεριλάβετε και όλα τα .html αρχεία που παράγει η Javadoc, τα JUnit tests καθώς και μικρά προγράμματα πελάτες που χρησιμοποιούν τις λειτουργίες των ΑΤΔ που υλοποιήσατε.

Παραδοτέα:

```
Multiset/Multiset.java
LeftRightStack/LeftRightStack.java
MusicCollection/Track.java
MusicCollection/MusicCollection.java
MultiMap/MultiMap.java
```

Επίσης, πρέπει να παραδώσετε τα παρακάτω αρχεία με JUnit Tests:

```
Multiset/MultisetTest.java
LeftRightStack/LeftRightStackTest.java
MusicCollection/TrackTest.java
MusicCollection/MusicCollectionTest.java
MultiMap/MultiMapTest.java
```

ΠΡΟΣΟΧΗ!!! Σε καμία περίπτωση δε πρέπει να αλλάξετε τα ονόματα των μεθόδων των συμβολαίων, τα ονόματα των κλάσεων ή των πακέτων που σας δίνονται.

Άσκηση 1 – Πολυσύνολα (20%)
Όνομα πακέτου: `Multiset`
Όνομα αρχείου: `Multiset.java`

Σχεδιάστε και υλοποιήστε τον αφαιρετικό τύπο δεδομένων (ΑΤΔ) για **πολυσύνολα** (με όνομα `Multiset`).Ένα πολυσύνολο είναι μία συλλογή αντικειμένων, τα οποία δεν είναι ταξινομημένα και ανακτούμε τα αντικείμενα με τυχαία σειρά. Ένα πολυσύνολο αποτελείται από έναν πίνακα αντικειμένων με ορισμένη χωρητικότητα και έχει συγκεκριμένο μέγεθος. Θεωρούμε ένα συμβόλαιο για τον ΑΤΔ **Multiset** που περιλαμβάνει τις εξής λειτουργίες:

- Δημιουργία ενός νέου πολυσυνόλου (1%):

```
public Multiset();
```

- Δημιουργία ενός νέου πολυσυνόλου με προκαθορισμένα μέγεθος και χωριτηκότητα (1%):

```
public Multiset(int capacity, int m_size);
```

- Προσθήκη ενός αντικειμένου στο πολυσύνολο (3%):

```
public void addItem (Object Item);
```

- Ανάκτηση ενός αντικειμένου από το πολυσύνολο (3%):

```
public Object getItem();
```

- Διαγραφή ενός αντικειμένου από το πολυσύνολο (3%):

```
public Object deleteItem();
```

- Έλεγχος εάν το πολυσύνολο είναι γεμάτο (1%):

```
public boolean isFull();
```

- Έλεγχος εάν το πολυσύνολο είναι άδειο (1%):

```
public boolean isEmpty();
```

- Επιστροφή του μεγέθους του πολυσυνόλου (1%):

```
public int currentSize();
```

- Επιστροφή της χωρητικότητας του πολυσυνόλου (1%):

```
public int capacity();
```

Σημασιολογία:

Η χωρητικότητα του κενού πολυσυνόλου μπορεί να οριστεί ίση με 50.

- Η ανάκτηση ή η διαγραφή ενός αντικειμένου από το πολυσύνολο γίνεται με τυχαίο τρόπο. Μπορείτε να χρησιμοποιήσετε την κλάση Random της Java ως γεννήτρια τυχαίων αριθμών. Δείτε τα παρακάτω links για την κατανόηση και χρήση της κλάσης Random:

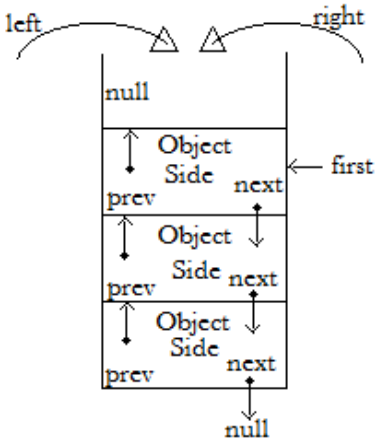
→ http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Random.html

→http://alvinalexander.com/java/java-random-class-random-int-integer-example

- Με την ανάκτηση ενός αντικειμένου από το πολυσύνολο δεν το διαγράφουμε.
- 5% του βαθμού της άσκησης παίρνουν τα Javadoc-JUnit test.

Άσκηση 2 – LeftRightStack (25%)
Όνομα πακέτου: `LeftRightStack`
Όνομα αρχείου: `LeftRightStack.java`

Σχεδιάστε και υλοποιήστε τον αφαιρετικό τύπο δεδομένων (ΑΤΔ) για την **LeftRightStack**, η οποία συμπεριφέρεται όπως η γνωστή στοίβα, εκτός από το γεγονός ότι τα στοιχεία που εισάγονται στη λίστα χαρακτηρίζονται από τα αναγνωριστικά **"left"** και **"right"**.



Θεωρούμε ένα συμβόλαιο για τον ΑΤΔ **LeftRightStack** που περιλαμβάνει τις εξής λειτουργίες:

- Δημιουργία μίας άδειας στοίβας (1%):

```
public LeftRightStack ();
```

- Έλεγχος αν η στοίβα είναι άδεια (1%):

```
public boolean isEmpty();
```

- Άδειασμα της στοίβας (1%):

```
public void makeEmpty();
```

- Επιστροφή μεγέθους της στοίβας (1%):

```
public int size();
```

- Εισαγωγή ενός νέου ”left” αντικειμένου στη στοίβα (2%):

```
public void pushLeft (Object Obj);
```

- Διαγραφή του πιο πρόσφατα εισαγόμενου ”left” αντικείμενου από τη στοίβα (1%):

```
public void popLeft();
```

- Επιστροφή του πιο πρόσφατα εισαγόμενου “left” αντικείμενου από τη στοίβα (2%):

```
public node topLeft();
```

- Επιστροφή και διαγραφή του πιο πρόσφατα εισαγόμενου “left” αντικείμενου από τη στοίβα (2%):

```
public node topAndPopLeft ();
```

- Εισαγωγή ενός νέου ”right” αντικειμένου στη στοίβα (2%):

```
public void pushRight (Object Obj);
```

- Διαγραφή του πιο πρόσφατα εισαγόμενου ”right” αντικείμενου από τη στοίβα (1%):

```
public void popRight();
```

- Επιστροφή του πιο πρόσφατα εισαγόμενου “right” αντικείμενου από τη στοίβα (2%):

```
public node topRight();
```

- Επιστροφή και διαγραφή του πιο πρόσφατα εισαγόμενου “right” αντικείμενου από τη στοίβα (2%):

```
public node topAndPopRight();
```

- Επιστροφή όλων των στοιχείων της στοίβας ως συμβ/ρές με το αναγνωριστικό “left” ή “right” (2%):

```
public String toString ();
```

Σημασιολογία:

- Κάθε στοίβα έχει ένα συγκεκριμένο μέγεθος το οποίο αλλάζει ανάλογα με τις λειτουργίες που εκτελούνται.
- Όπως βλέπετε στο παραπάνω σχήμα, η στοίβα αποτελείται από **Nodes** και κάθε **Node** έχει ορισμένα χαρακτηριστικά. Πριν ξεκινήσετε την υλοποίησή σας ορίστε τη συγκεκριμένη κλάση. Μπορεί να οριστεί και ως private class μέσα στην κλάση LeftRightStack.
- Σε περίπτωση που εκτελούνται οι εντολές popLeft/topAndPopLeft (αντίστοιχα για right) και δεν υπάρχει τέτοιος κόμβος, επιστρέφεται null.

- 5% του βαθμού της άσκησης παίρνουν τα Javadoc-JUnit test.

Άσκηση 3 – Μουσική συλλογή (25%)
Όνομα πακέτου: `MusicCollection`
Ονόματα αρχείων: `Track.java`
`MusicCollection.java`

Σχεδιάστε και υλοποιήστε τους Αφαιρετικούς Τύπους Δεδομένων (ΑΤΔ) ενός μουσικού κομματιού (**`Track`**) και μίας μουσικής συλλογής (**`MusicCollection`**). Ένα μουσικό κομμάτι αποτελείται από τον τίτλο του (title), το όνομα του καλλιτέχνη (artist), το όνομα του άλμπουμ (album), τις φορές που έχει αναπαραχθεί (plays) και τη διάρκειά του σε δευτερόλεπτα (duration). Μία μουσική συλλογή αποτελείται από το όνομά της (name), μία λίστα με τα τραγούδια που περιέχει (trackList) και το πλήθος των κομματιών που περιέχει (total). Θεωρούμε ένα συμβόλαιο για τον ΑΤΔ **`Track`** που περιλαμβάνει τις παρακάτω λειτουργίες:

- Δημιουργία ενός κομματιού (0.5%):

```
public Track(String newTitle, String newArtist,
String newAlbum, int newPlays, int newDuration);
```

- Επιστροφή του ονόματος του κομματιού (0.5%):

```
public String getTitle();
```

- Αλλαγή του ονόματος του κομματιού (0.5%):

```
public void setTitle(String newTitle);
```

- Επιστροφή του ονόματος του καλλιτέχνη (0.5%):

```
public String getArtist();
```

- Αλλαγή του ονόματος του καλλιτέχνη (0.5%):

```
public void setArtist(String newArtist);
```

- Επιστροφή του ονόματος του άλμπουμ (0.5%):

```
public String getAlbum();
```

- Αλλαγή του ονόματος του άλμπουμ (0.5%):

```
public void setAlbum(String newAlbum);
```

- Επιστροφή του πλήθους των αναπαραγωγών του κομματιού (0.5%):

```
public int getPlays();
```

- Επιστροφή της διάρκειας του κομματιού (σε δευτερόλεπτα) (0.5%):

```
public int getDuration();
```

- Αναπαραγωγή του κομματιού (0.5%):

```
public void play();
```

- Επιστροφή της αναπαράστασης σε String του κομματιού (1%):

```
public String toString();
```

Θεωρούμε ένα συμβόλαιο για τον ΑΤΔ **MusicCollection** που περιλαμβάνει τις παρακάτω λειτουργίες:

- Δημιουργία μίας μουσικής συλλογής (0.5%):

```
public MusicCollection(String newName);
```

- Επιστροφή του ονόματος της μουσικής συλλογής (0.5%):

```
public String getName();
```

- Αλλαγή του ονόματος της μουσικής συλλογής (1%):

```
public void setName(String newName);
```

- Προσθήκη ενός μουσικού κομματιού στη μουσική συλλογή (1%):

```
public boolean addTrack(Track tr);
```

- Διαγραφή ενός μουσικού κομματιού από τη μουσική συλλογή με βάση το δοθέντα τίτλο και άλμπουμ και επιστροφή του (2%):

```
public Track removeTrack(String title, String
album);
```

- Αλλαγή του ονόματος ενός κομματιού, δοθέντος του προηγούμενου τίτλου του και του άλμπουμ του (1%):

```
public void renameTrack(String oldTitle, String
album, String newTitle);
```

- Αλλαγή του άλμπουμ ενός κομματιού, δοθέντος του τίτλου του και του προηγούμενου άλμπουμ του (1%):

```
public void setAlbum(String title, String oldAlbum,
String newAlbum);
```

- Αλλαγή του καλλιτέχνη ενός κομματιού, δοθέντος του τίτλου του και του άλμπουμ του (1%):

```
public void setArtist(String title, String album,
String newArtist);
```

- Αναπαραγωγή ενός κομματιού με βάση το δοθέντα τίτλο και άλμπουμ του (1%):

```
public void playTrack(String title, String album);
```

- Επιστροφή του πλήθους των κομματιών της μουσικής συλλογής (0.5%):

```
public int getTotalTracks();
```

- Επιστροφή του συνολικού χρόνου αναπαραγωγής της λίστας κομματιών (σε δευτερόλεπτα) (1%):

```
public int getTotalTime();
```

- Επιστροφή του κομματιού στη μουσική συλλογή με τις περισσότερες αναπαραγωγές (1%):

```
public Track getMostPlayed();
```

- Διαγραφή όλων των κομματιών από τη λίστα κομματιών της μουσικής συλλογής (0.5%):

```
public void removeAll();
```

- Επιστροφή της αναπαράστασης σε String της μουσικής συλλογής (2%):

```
public String toString();
```

Σημασιολογία:

- Ο αριθμός των αναπαραγωγών (plays) κάθε κομματιού είναι ένας μετρητής που αυξάνεται κάθε φορά που το τραγούδι αναπαράγεται.
- Για να βρίσκετε και να επιστρέφεται ένα κομμάτι στη λίστα, θα σας βοηθήσει να υλοποιήσετε μία μέθοδο που θα εντοπίζει το index αυτού του κομματιού μέσα στη LinkedList με βάση τον τίτλο και το άλμπουμ του.
- Η μέθοδος getTotalTime() πρέπει να υπολογίζει το συνολικό χρόνο αναπαραγωγής της λίστας κομματιών. Αυτό γίνεται με το να υπολογίζει το άθροισμα του χρόνου αναπαραγωγής κάθε κομματιού, ο οποίος εξαρτάται από το πλήθος των αναπαραγωγών του (plays) και τη διάρκειά του (duration).
- Οι μέθοδοι toString πρέπει να εμφανίζουν τις πληροφορίες όσο το δυνατό καλαίσθητα και αυτές να «βγάζουν νόημα».
- 5% του βαθμού της άσκησης παίρνουν τα Javadoc-JUnit test.

Άσκηση 4 – Απεικόνιση (MultiMap) (30%)

Όνομα πακέτου: `MultiMap`
Ονόματα αρχείων: `MultiMap.java`

Σχεδιάστε έναν Αφαιρετικό Τύπο Δεδομένων (ΑΤΔ) Απεικόνιση (**`MultiMap`**). Το **`MultiMap`** αντιστοιχίζει ένα κλειδί σε ένα **σύνολο** αντικειμένων σε αντίθεση με το απλό **`Map`** που αντιστοιχίζει ένα κλειδί σε **ένα** αντικείμενο. Ο σχεδιασμός του **`MultiMap`** θα πρέπει να λαμβάνει υπόψη τις παρακάτω λειτουργίες:

- Δημιουργία μίας κενής απεικόνισης (1%):

```
public MultiMap();
```

- Έλεγχος αν η απεικόνιση είναι άδεια (2%):

```
public boolean isEmpty();
```

- Επιστροφή του πληθαρίθμου της απεικόνισης (2%):

```
public int size();
```

- Επιστροφή των αντικειμένων στην απεικόνιση, δεδομένου ενός δοσμένου κλειδιού (2%):

```
public LinkedList get(Object key);
```

- Έλεγχος για το αν δύο απεικονίσεις είναι ίσες (2%):

```
public boolean equals(MultiMap m);
```

- Επιστροφή του συνόλου των κλειδιών στην απεικόνιση (2%):

```
public TreeSet keySet();
```

- Διαγραφή όλων των εγγραφών στην απεικόνιση (1%):

```
public void clear();
```

- Προσθήκη μίας νέας εγγραφής στην απεικόνιση, δίνοντας το ζεύγος <κλειδί, αντικείμενο>. Επιστροφή των αντικειμένων, αν υπήρχαν, που ήταν συνδεδεμένα με το δοσμένο κλειδί πριν την εισαγωγή (3%):

```
public LinkedList put(Object key, Object value);
```

- Διαγραφή μίας εγγραφής από την απεικόνιση, δίνοντας το ζεύγος <κλειδί, αντικείμενο>. Επιστροφή του αντικειμένου που μόλις διαγράφηκε (3%):

```
public Object remove(Object key, Object value);
```

- Διαγραφή όλων των εγγραφών της απεικόνισης που αντιστοιχούν στο δοσμένο κλειδί. Επιστροφή της λίστας των αντικειμένων που ήταν συνδεδεμένα με το δοσμένο κλειδί πριν τη διαγραφή (3%):

```
public LinkedList remove(Object key);
```

- Αντικατάσταση όλων των εγγραφών της απεικόνισης με τις εγγραφές μίας άλλης δοθείσας απεικόνισης (2%):

```
public void replace(MultiMap m);
```

- Εκτύπωση των εγγραφών της απεικόνισης (2%):

```
public void printMultimap();
```

Στην εικόνα 1 φαίνεται ένα παράδειγμα απεικόνισης (`MultiMap`). Ισοδύναμα, και βοηθητικά ως προς την υλοποίησή σας, μπορείτε να δείτε και την απεικόνιση στην εικόνα 2.

Key	***Value***
K_1	V_1
K_1	V_2
K_1	V_3
K_2	V_4
...	...

Εικόνα 1

Key	***Value***
K_1	$\{V_1,V_2,V_3\}$
K_2	$\{V_4\}$
...	...
...	...
...	...

Εικόνα 2

Σημασιολογία:

- Η υλοποίηση του MultiMap θα πρέπει να γίνει με συνδεδεμένες λίστες.
- Το MultiMap περιέχει εγγραφές (records). Οι εγγραφές αυτές περιέχουν ένα μοναδικό κλειδί και μία λίστα αντικειμένων που αντιστοιχούν σε αυτό το κλειδί. Θα πρέπει να ορίσετε μία κλάση για τις εγγραφές που θα έχει αυτά τα χαρακτηριστικά. Αυτή η κλάση μπορεί να οριστεί σαν private στην κλάση MultiMap.
- Οι μέθοδοι remove και get θα πρέπει να επιστρέφουν null αν δεν βρεθούν οι αντίστοιχες εγγραφές.
- Για τη μέθοδο equals θα σας βοηθήσει να δημιουργήσετε μία επιπλέον μέθοδο που θα ελέγχει αν δύο LinkedList είναι ίδιες, ανεξαρτήτως της σειράς που εμφανίζονται τα αντικείμενα σε αυτές.
- 5% του βαθμού της άσκησης παίρνουν τα Javadoc-JUnit test.