Διευκρινήσεις κώδικα

Hy240b project 2014

phase 1

Parsers (1/4)

- Οι parsers γενικά, στη μορφή που θα τους συναντήσετε, είναι ο κώδικας που εξάγει δομημένη πληροφορία από strings (σειρά από χαρακτήρες). Στη δική μας περίπτωση οι parsers είναι αυτοί που θα πάρουν την πληροφορία από τα αρχεία και θα μας τα δώσουν έτοιμα με τη μορφή των class/struct.

- Στον κώδικα που σας δίνεται, υπάρχει μία κλάση **FileParsers** (και είναι αυτή που μας ενδιαφέρει), και δύο κλάσεις/structs **MovieData**, **EventData.**

- Η κλάση **FileParsers** περιέχει κώδικα που ανοίγει τα αρχεία, τα διαβάζει, και επιστρέφει τη πληροφορία που υπάρχει στα αρχεία με την μορφή των κλάσεων/structs **MovieData**, **EventData**.

- Οι κλάσεις/structs **MovieData**, **EventData** υπάρχουν μόνο για να πακετάρουν την πληροφορία των ταινιών και του event και δεν κάνουν κάτι ιδιαίτερο.

Parsers (2/4)

- Η κλάση **FileParsers** περιέχει 6 μεθόδους(συναρτήσεις).
  - **initialize()** : αρχικοποιεί τους parsers και ανοίγει τα αρχεία για διάβασμα
  - **finalize()**: κλείνει τα αρχεία που και τερματίζει τους parsers
  - **hasNextMovie() / hasNextEvent()** : επιστρέφουν true/false ανάλογα με το αν υπάρχει άλλη ταινία/event στο αρχείο
  - **getNextMovie() / getNextEvent()** : επιστρέφει το επόμενο MovieData / EventData που υπάρχει στο αρχείο ή null αν δεν υπάρχει επόμενο

- Αρχικά θα πρέπει να καλέσετε **FileParsers.initialize([path1], [path2]);** για να αρχικοποιηθούν οι parsers και να ορίσετε από ποια αρχεία θα διαβάσουν τις πληροφορίες, όπου:
  - [path1] : ένα string με το file path του αρχείου των movies
  - [path2] : ένα string με το file path του αρχείου των events

Parsers (3/4)

- Στη συνέχεια, θα εξετάζετε με την συνάρτηση **hasNextMovie()** αν υπάρχει επόμενο στοιχείο ταινίας στο αρχείο, και αν ναι, τότε καλώντας **getNextMovie()** θα παίρνετε την επόμενη ταινία με την μορφή της class/struct MovieData**.** (Αντίστοιχα γίνεται και για τα events).

- Αυτό που σας ενδιαφέρει σε αυτό το κομμάτι είναι να καταλάβετε με ποιο τρόπο θα καλείτε τις μεθόδους(συναρτήσεις) της κλάσης FileParsers για να παίρνετε τα δεδομένα από τα αρχεία.

- Υπάρχει παράδειγμα στην main() που σας δίνεται όπου μπορείτε να δείτε πως ακριβώς χρησιμοποιούνται.

Parsers (4/4)

Τα πεδία των κλάσεων/structs MovieData, EventData που παίρνουμε από τον FileParser περιγράφονται παρακάτω:

- **MovieData**
  - **Id** : το id της ταινίας
  - **Title** : ο τίτλος της ταινίας
  - **Year :** η χρονιά που προβλήθηκε η ταινία
  - **Rating** : η βαθμολογία της ταινίας
  - **Votes:** ο αριθμός των ψήφων που πήρε η ταινία
  - **Duration** : η διάρκεια της ταινίας σε λεπτά
  - **Genres** : μία λίστα με τις κατηγορίες τις οποίες ανήκει η ταινία
- **EventData**
  - **Operation** : είναι το γράμμα “I” ή το γράμμα “D” που περιγράφει αν θα πρέπει να γίνει Insert ή Delete από τη λίστα αντίστοιχα.
  - **Username** : το όνομα του χρήστη
  - **movieId** : το id της ταινίας που θέλει να παρακολουθήσει ή παρακολούθησε ο χρήστης

Movies List (1/5)

Οι κλάσεις (λίστες) που πρέπει να υλοποιήσετε είναι οι : **UnsortedMovieList, SortedMovieList, SelfAdjustingMovieList** και περιέχουν:

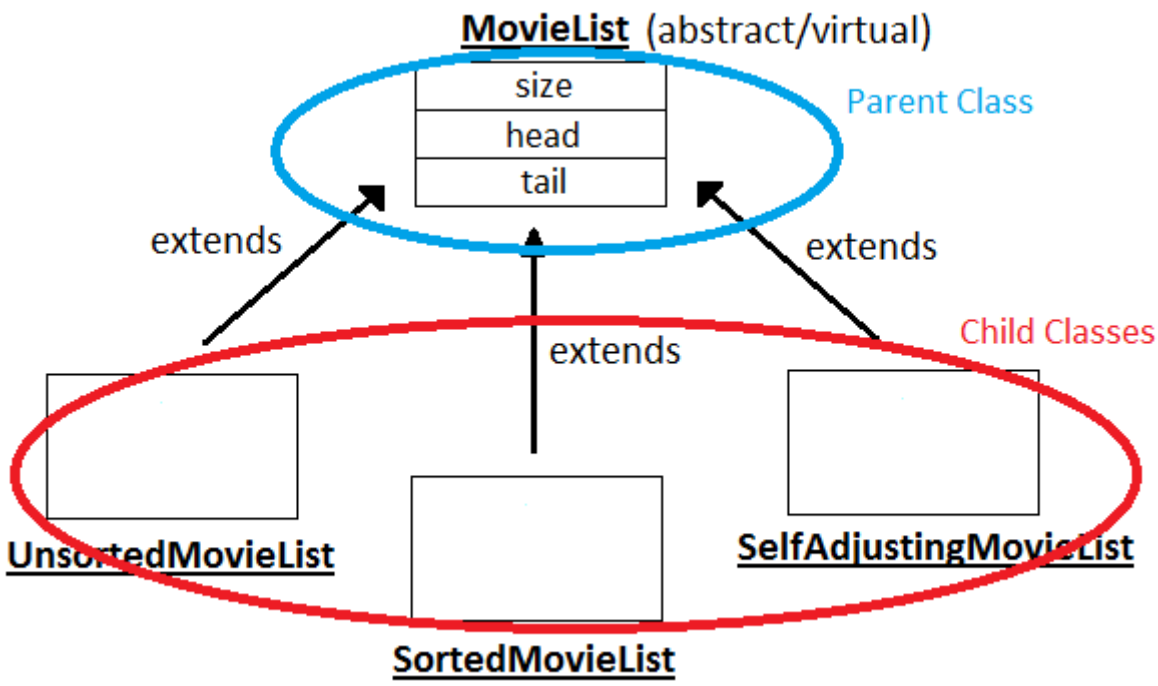
- Τα πεδία:
  - size : το μέγεθος της λίστας
  - head : περιέχει/δείχνει στον πρώτο κόμβο της λίστας
  - tail : περιέχει/δείχνει στον τελευταίο κόμβο της λίστας
- Τις μεθόδους(συναρτήσεις):
  - clear()
  - getSize()
  - isEmpty()
  - insert()
  - remove()
  - get()

Movies List (2/5)

- **Όμως** τα πεδία size, head, tail και οι συναρτήσεις clear(), getSize(), isEmpty() είναι **κοινά** και για τις 3 λίστες. Οπότε τα πεδία και ο κώδικας της υλοποίησης των συναρτήσεων μπήκαν σε μια **abstract**(java)**/virtual**(C++) **class** με όνομα **MovieList.**
- Το **abstract/virtual** σημαίνει ότι **δεν μπορείτε να δημιουργήσετε** instance/στιγμιότυπο(”μεταβλητή”) της κλάσης **MovieList**. Αυτή η κλάση **υπάρχει μόνο για να ορίζει κάποια βασικά στοιχεία** που πρέπει να υπάρχουν.
- Οι κλάσεις **UnsortedMovieList, SortedMovieList, SelfAdjustingMovieList** είναι **επεκτάσεις** της κλάσης **MovieList** (κληρονομούν τα στοιχεία της). Αυτό σημαίνει ότι οι 3 κλάσεις θα έχουν τα δικά τους πεδία, μεθόδους(συναρτήσεις) **+ τα πεδία και μεθόδους της κλάσης MovieList**

Movies List (3/5)

- Οι 3 κλάσεις που επεκτείνουν την MovieList, σχηματικά είναι κάπως έτσι:



Movies List (4/5)

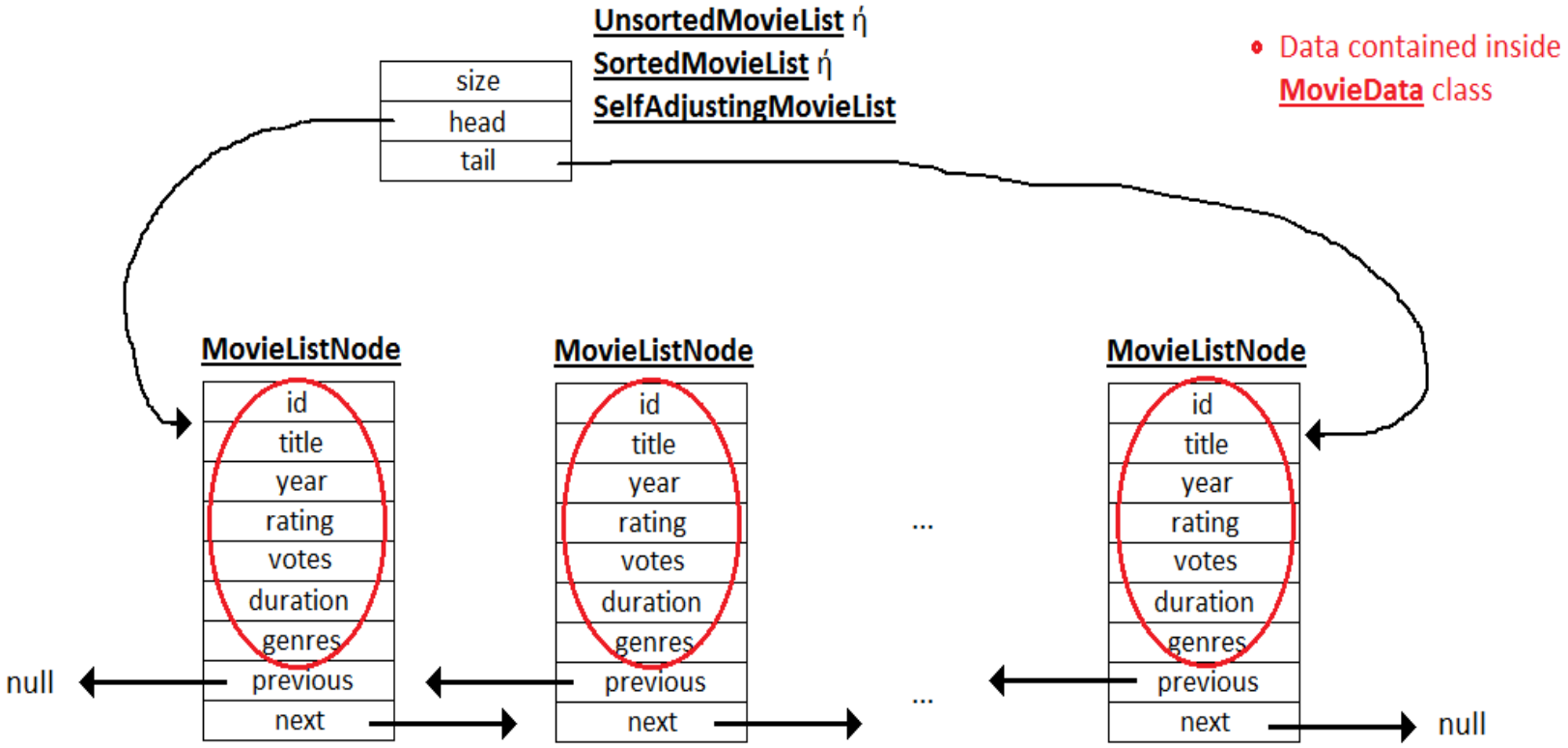
- **Πρακτικά**, αυτό που πρέπει να κάνετε εδώ είναι **η υλοποίηση των μεθόδων (συναρτήσεων)** που υπάρχουν μέσα στις MovieList, UnsortedMovieList, SortedMovieList, SelfAdjustingMovieList.
- Αυτό που πρέπει να γνωρίζετε είναι ότι τα πεδία και οι μεθόδοι(συναρτήσεις) που υπάρχουν στην MovieList **είναι σαν να υπήρχαν στην κάθε μια** από τις UnsortedMovieList, SortedMovieList, SelfAdjustingMovieList.

- Εκτός από τις 4 κλάσεις που είδαμε προηγουμένως, υπάρχει και η κλάση/struct **MovieListNode** που περιγράφει έναν κόμβο της λίστας.

  (Για τη Java: οι μεθόδοι(συναρτήσεις) get… /set… που υπάρχουν στην κλάση MovieListNode το μόνο που πρέπει να κάνουν είναι να επιστρέφουν /θέτουν τιμές από/στα πεδία της κλάσης αντίστοιχα)

Movies List (5/5)

- Σχηματικά, η λίστα των ταινιών είναι κάπως έτσι:



Users List (1/5)

- Για τη λίστα των Users, υπάρχουν 3 classes:
  - **UserList**: η λίστα των users
  - **UserListNode**: ο κόμβος της λίστας των users **ΚΑΙ** η λίστα των ταινιών που θέλει να παρακολουθήσει ο χρήστης
  - **WatchMovieListNode**: ο κόμβος της λίστας των ταινιών που θέλει να παρακολουθήσει ο κάθε χρήστης

Users List (2/5)

Η class **UsersList** περιέχει:

- Τα πεδία:
  - **head**: που περιέχει το πρώτο στοιχείο της λίστας

- Τις μεθόδους(συναρτήσεις) :
  - **insert()** : εισάγει ένα νέο κόμβο στη λίστα
  - **remove()** : διαγράφει τον κόμβο από τη λίστα αν υπάρχει
  - **get()** : επιστρέφει τον κόμβο από τη λίστα αν υπάρχει, αλλιώς επιστρέφει null
  - **isEmpty()** : επιστρέφει true/false ανάλογα με το αν η λίστα είναι άδεια ή όχι.

Users List (3/5)

Η class **UsersListNode** περιέχει:

- Τα πεδία:
  - **username**: που περιέχει όνομα του χρήστη
  - **next**: που δείχνει στο επόμενο στοιχείο της λίστας των Users
  - **watchMovieList**: που περιέχει το πρώτο στοιχείο της λίστας των ταινιών που θέλει να παρακολουθήσει ο χρήστης

- Τις μεθόδους(συναρτήσεις) :
  - **get()/set()** : επιστρέφουν/θέτουν τα ανάλογα πεδία της κλάσης
  - **insertMovie()** : εισάγει ένα καινούργιο κόμβο στη λίστα των ταινιών
  - **removeMovie()** : διαγράφει τον κόμβο από τη λίστα των ταινιών αν υπάρχει
  - **containsMovie()** : επιστρέφει true/false ανάλογα με το αν η λίστα των ταινιών περιέχει την ταίνια με παράμετρο
  - **isMoviesListEmpty()** : επιστρέφει true/false ανάλογα με το αν η λίστα των ταινιών είναι άδεια ή όχι.

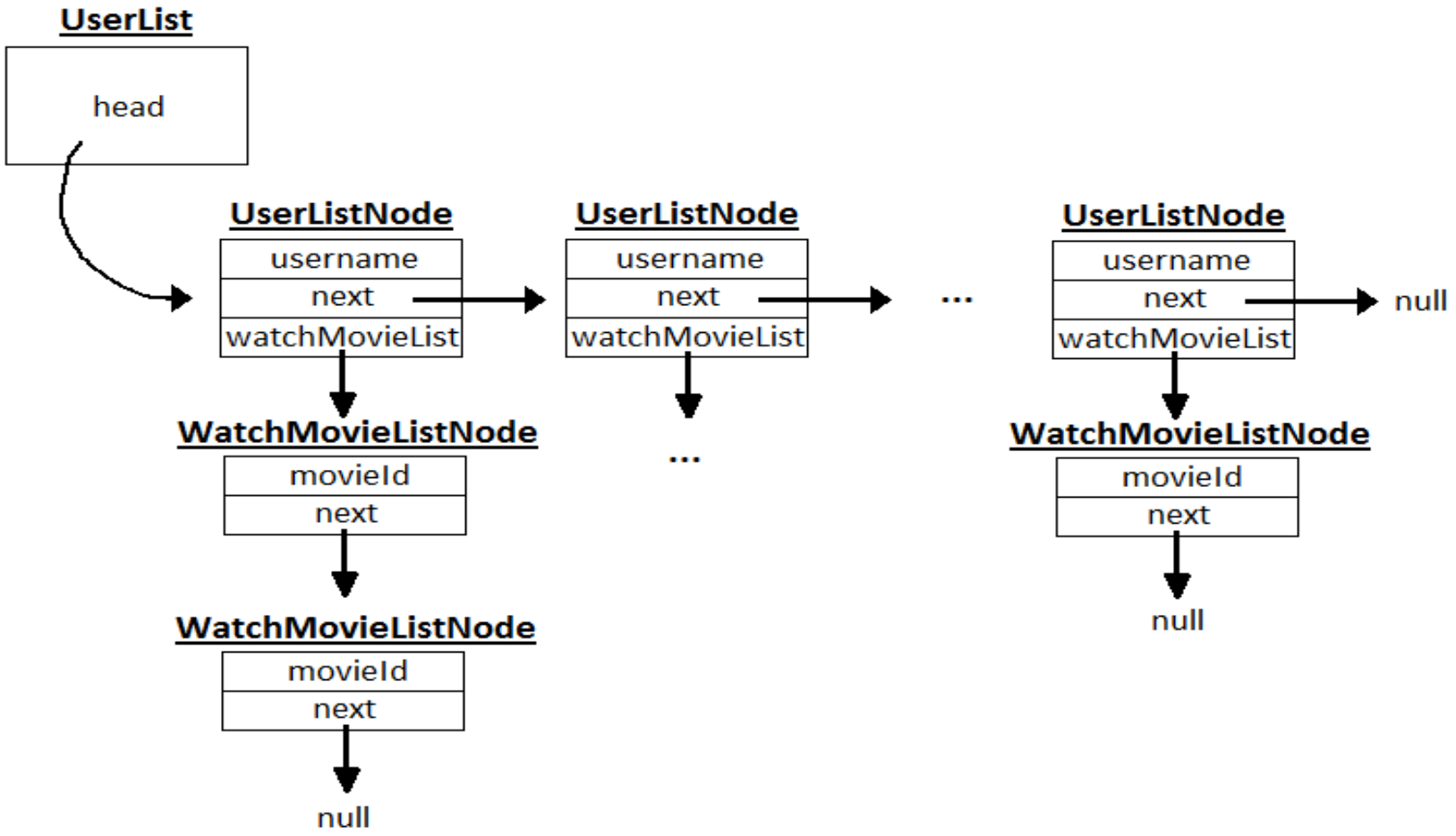
Users List (4/5)

Η class **WatchMovieListNode** περιέχει:

- Τα πεδία:
  - **movieId**: το id της ταινίας που θέλει να δει ο χρήστης
  - **next**: που δείχνει στο επόμενο στοιχείο της λίστας του watchMovieList
- Τις μεθόδους(συναρτήσεις) :
  - **get()/set()** : επιστρέφουν/θέτουν τα ανστίστοιχα πεδία της κλάσης

Users List (5/5)

- Σχηματικά η λίστα των Users έχει ως εξής:



Μετρήσεις χρόνων (1/1)

- Για να πάρετε μετρήσεις χρόνων που ζητείται στην εκφώνηση πχ. για την αρχικοποίηση της λίστας των ταινιών.
  1. Παίρνετε το χρόνο του συστήματος πριν αρχίσετε να εισάγετε στοιχεία στη λίστα.
  2. Παίρνετε το χρόνο μετά το τέλος όλων των εισαγωγών.
  3. Κάνετε τη διαφορά των 2 και βλέπετε πόση ώρα πέρασε.

- **Java:**
  - long currTime = System.currentTimeMillis();

(Επιστρέφει το χρόνο του συστήματος σε ms)

- **C++:**
  - #include <ctime>
  - int time_in_millis = (std::clock()) / (double)(CLOCKS_PER_SEC / 1000);