PRÀCTICA 2
IMPLEMENTACIÓ DE TADs GENÈRICS

ALUMNE/S: Chenglong Zou
PROFESSORA: Jordi Duch
ASSIGNATURA: Estructura de Dades
ENSENYAMENT: Grau Enginyeria Informàtica
DATA: 29/03/2012.

ÍNDEX

Estudi Previ i Disseny

La pràctica demana implementar dos TADs (LinkedList i ShowManager).

...

Situant-nos al programa principal.

Ens demana que quan fem run:

1r demanar un enter que serà la capacitat del servidor:

Llegir una entrada per teclat.

2n demanar el tipus de gestió que farem servir:

Llegir una entrada per teclat, serà estàtica o dinàmica.

// aquí no he tingut clar si en la dinàmica calia donar-li una mida màxima, per suposició // li he ficat un límit.

3r independentment de la implementació, cal llegir les dades d'entrada que serà del fitxer XML.

Primer de tot pel node de Sèrie podem saber el nombre de series que haurem de emmagatzemar (les iteracions concretes que haurem de fer), ara creem instancia de sèrie i entrem en un altre iterador amb n iteracions on n es l'índex d'episodis segons el node episodis de la sèrie on ens trobem... una vegada tenim la sèrie ho introduïm al Show manager.

Quan tinguem omplert el Show Manager , entrem en el bucle del programa principal on hi intervindrà l'usuari, aquí l'usuari tindrà l'opció de consultar series, els episodis que té una sèrie, el nombre de series que hi ha al sistema, el nombre d'episodis de un any especificat, l' identitat d'una sèrie, afegir sèrie nova al sistema, afegir episodi a una sèrie, eliminar una sèrie i eliminar un episodi d'una sèrie.

El TAD LinkedList, l'estructura que he fet servir, és una llista doblement enllaçada circular.

Versió estàtica

Constructor, tenint per entrada una mida màxima, creem un Vector de nodes on cada node té un índex següent i anterior més un element abstracte(Sèrie) i un codi identificador, i tindrà tantes nodes com la mida màxima. Una vegada instanciat, haurem de actualitzar els índex següents i anteriors per primera vegada, i tenint que el següent del últim és el primer i el anterior del primer és el últim, he indexat la 1r posicio lliure a 0 i P també 0(suposant fins que no se esborri per primera vegada sempre introduirem al 0 com el primer)

"addElement" Per afegir he indicat que sempre afegeixi al següent lliure i vaig actualitzat el punter següent lliure. Quan s'omple del tot li dic que següent lliure es -1, només per senyalitzar-ho.

"removeElement" en lloc d'eliminar l'element, busco si es troba en el vector, si el trobo el desplaço davant del primer element indexat i després actualitza els enllaços corresponents.

"Es últim", agafo el índex que té seguentlliure marcant la posició anterior.

"Es buit", he crear un variable que comptabilitza el nombre d'elements que he afegit i eliminat, aleshores, sabrem que quan indiqui 0 implica que esta buit.

"get Element" tenint el codi identificador, recorro a partir de la posició del primer element i faig següent, tantes vegades com número d'elements que hi ha al sistema i comparo el codi amb els codis del sistema indexats.

"GetElements" retorno tants elements com el numero d'elements que hi ha al sistema des del primer afegit fins al últim afegit.

"get num Elements" retorna el nombre d'elements que hi ha al sistema que coincideix amb el variable numElems.

Versió dinàmica

Constructor, realment no sé si fa falta tindre'n.

A diferencia de la estàtica aquí només treballem amb nodes, i com a referència tenim un node "Raiz", per afegir , si és el primer element aleshores faig que s'apunti següent i anterior a si mateix, després d'aquest cada element que afegim tindrà com a següent el primer element de la llista i anterior l'últim element afegit a la llista.

Per esborrar , ara a diferencia de l'estàtica ens interessa alliberar la memòria, per tant fem que els nodes adjacents es interconectin i l'element a esborrar li fiquem un "Null"(suposant que amb això hi ha suficient per esborrar-lo).

"getElement" és recórrer la llista i anar mirant si coincideix el codi i "getElements" és passar del vector de nodes, a un vector d'elements(és un atribut de nodes).

El TAD ShowManager

ShowManager(rep per paràmetre la mida màxima del sistema de gestió de series(nº màxim de series que por suportar)i type(el tipus de linkedList que farem servir)

"addShow" és fer un add element a la instancia de show manager.

addshowEpisode, primer cal buscar si existeix la sèrie recorrent el sistema i comparant el codi amb cada element del sistema, si el trobem, caldrà afegir-li un capítol a la sèrie(això només és possible amb series noves que afegim(o en el cas que li esborrem un capítol de la sèrie per tornar-lo a inserir), ja que he fet que la mida de totes les series de l'XML tingui una mida màxima igual al nombre de series que hi havia).

getNumshows, retorna un variable "nEle" que és un comptador de nombre de series que hi ha al sistema, que incrementa amb cada addshow amb èxit.

getAllshows, aprofitant getelements(), temin que retorna totes les series que hi ha disponible al sistema, he fet un bucle for per retornar els elements amb un vector mida.

getAllEpisodes, un getelements i obtindre la sèrie amb l'id que li especifiquem i si lo que ens retorna no és null aleshores retornem una sèrie sinó retornem null.

Get Episodes by year, consisteix en fer gairebé lo mateix que el cas anterior, però filtrant els elements amb l'any que ens especifica per paràmetre.

removeShowEpisode, rebem per paràmetre l'id de la sèrie i l'id de l'episodi, mirem si existeix la sèrie i després mirem si es troba l'episodi dins de la sèrie, en cas afirmatiu haurem d'esborrar lo posant utilitzant el mètode remove Episode(dins de la classe ShowNO, perquè he optat per ShowNO).

removeShow, rebem per paràmetre l'id de la sèrie, utilitzem el mètode remove element del Linkedlist estàtic/dinàmic i decremento el comptador de numElements.

Part Opcionals:

- Excepcions: He creat unes classes d'excepcions, he fet saltar Excepcions estàndards.
- Gestió de memòria lliure, no he entès el concepte, suposo que en el meu cas "cua circular" seria dividir la memòria per una banda espai lliure i per un altre ocupat. Crec que lo que he implementat fa que cada vegada que afegeixo un element aquest passa a ocupar un lloc següent de l'últim afegit, i quan elimino un element faig que els dos veïns es connectin i allibero aquest espai i moc aquest espai darrere del primer element afegit i davant del l'últim espai lliure, i convertint-se en l'últim espai lliure.
- Afegir series de forma ordenada per Codi, no està implementat.

Conclusions

He aprofundit els meus coneixements sobre els TADs Genèrics i les seves implementacions.

Treball futur

- Optimitzar els codis.
- Ordenar una llista genèrica(títol i codi).
- Utilització de comentaris javadocs.
- Aprofundir l'utilització d'excepcions...
- He tingut problemes amb els nodes, en el cas dinàmic em sobre dos atributs.