Metodologies de la Programació

Anàlisi d'etiquetes en diferents piulades

UNIVERSITAT ROVIRA i VIRGILI

Pedro Espada Oviedo i Sergi Vives Pellisé

Introducció i Objectiu

En aquesta pràctica hem d'implementar un programa que llegeixi piulades des d'un fitxer de text i que permeti realitzar diferents consultes sobre aquestes dades. Per a fer-ho es demana la implementació d'un Tipus Abstracte de Dades (TAD) que compleixi amb aquesta especificació i dues classes que implementin aquesta especificació, una mitjançant memòria estàtica i una altra mitjançant memòria dinàmica.

Disseny

El disseny i programació de l'algorisme i diagrama de classes es va realitzar de forma descendent. Es va començar definint la *Interface* `TADcjtEtiquetes` que especifica la firma dels mètodes que ha de posseir el nostre TAD partint del llenguatge natural de les especificacions. Cada un d'aquests mètodes es correspon amb una de les funcionalitats de consulta que es requereixen a l'enunciat.

Per a implementar de forma transparent utilitzant memoria dinàmica o estàtica es va implementar el TAD `TADLlistaGenerica<T extends Comparable<T>>` que defineix l'especificació que hauria de tenir una llista d'elements ordenats no repetits (considerem que no té sentit que una persona faci dos tweets idèntics simultàniament).

Vam definir dues classes que implementen l'anterior *interface*: `LlistaDinamica<T extends Comparable<T>>` i `LlistaEstatica<T extends Comparable<T>>` . La primera es una llista d'objectes *wrapper* que embolcallen el tipus T i aporten un punter al següent *wrapper*. La segona és una llista estàtica encadenada. Cada element de la llista és un *wrapper* que embolcalla al tipus T i aporta un índex al següent *wrapper*.

Aquestes dues classes utilitzen la classe *wrapper* `Obj<T extends Comparable<T>>`.
De la cual hereden `ObjCursor<T extends Comparable<T>>` i `ObjReferencia<T extends Comparable<T>>` que es corresponen amb els *wappers* propis de cada llista.

Després es van implementar les classes de dades. Aquestes són la classe `Piulada` i la classe `Etiqueta` . Aquestes dues classes contenen les dades del nostre programa i són les classes que parametritzaran les nostres llistes genèriques a l'hora de fer-les servir. La classe `Etiqueta` conté una propietat `TADLlistaGenerica<Piulada>` ja que una etiqueta està relacionada amb diverses piulades.

Es va crear una classe `Iterator` per a recórrer fàcilment les diferents llistes de forma transparent.

Després es va crear la classe `Etiquetes` que implementa la interfície `TADcjtEtiquetes`. En aquesta classe s'implementen totes les funcionalitats requerides a l'enunciat utilitzant la funcionalitat de les classes anteriorment definides. El constructor de la classe rep un paràmetre de tipus `TADLlistaGenerica<Etiqueta>` per a inicialitzar la

llista interna, de forma que de forma transparent s'inicialitzarà amb una `LlistaDinamica<Etiqueta>` o bé amb una `LlistaEstatica<Etiqueta>`. També es guarda una propietat `implementation` que diu quin tipus de llista s'està fent servir. Això es fa per a inicialitzar les llistes genèriques de cada etiqueta amb un tipus de llista o altre, de forma que totes les llistes (tant la general de les etiquetes com la de piulades de cada etiqueta) siguin del mateix tipus. Per a implementar la majoria de les funcionalitats es va recórrer de forma transparent les llistes internes utilitzant *iterators*.

Finalment, per a fer el joc de proves es va construir una petita interfície per interactuar amb el programa directament utilitzant l'*user input*. Aquesta interfície permet mostrar l'estat de l'estructura de dades i provar totes les funcionalitats requerides a l'exercici. A més, permet seleccionar quin tipus d'estructures de dades volem utilitzar.

Joc de proves

Es van escriure diferents piulades seguint el format donat a l'enunciat fins a tenir-ne una quantitat considerable (unes 100). Es va implementar la medició del temps per cada operació disponible al menú. Es van provar totes les operacions de forma exhaustiva utilitzant diferents dades com a input. El joc de proves s'inclou al projecte lliurat.

Valoració Temps / Cost

Llista dinàmica:

A part de les pròpies dades genèriques cal emmagatzemar un punter per cada element. A més cal emmagatzemar el primer punter i el nombre d'elements.

- **Consultar**: Cost lineal. Per a cercar a la llista hem de recorrer la llista.
- **Esborrar:** Cost lineal. Per a esborrar l'element hem de cercar-lo.
- **Afegir:** Cost lineal. Per a afegir l'element de forma ordenada cal trobar on hem de colocar-lo.

Llista estàtica encadenada

A part de les pròpies dades genèriques cal emmagatzemar un índex per cada element. A més cal emmagatzemar el índex, el nombre d'elements, una pila amb els espais buits i el nombre d'espais buits..

- **Consultar:** Cost lineal. Per a cercar a la llista hem de recorrer la llista.
- **Esborrar:** Cost lineal. Cal destacar que en una llista estàtica esborrar té el mateix cost, però és degut a que tant com cercar l'element com eliminar els "forats" de la llista és lineal. En una llista encadenada tenim cost lineal degut a que hem de buscar l'element, però no cal eliminar "forats" després de borrar-lo ja que utilitzem una pila per a controlar els espais buits.

- **Afegir:** Cost lineal. Per a afegir l'element de forma ordenada cal trobar on hem de colocar-lo.