PRG (ETS d'Enginyeria Informàtica) - Curs 2016-2017

Pràctica 5. Implementació i ús d'estructures de dades lineals

(3 sessions)

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Índice

1. Context i treball previ

En el context acadèmic, aquesta pràctica correspon al *"Tema 5. Estructures de dades lineals"*.
Els objectius detallats de la pràctica són els següents:

- Treballar amb elements habituals en l'ús de la memòria enllaçada: referències, nodes, etc.
- Implementar una classe conjunt de `String` usant una llista o seqüència enllaçada ordenada, posant especial èmfasi en les operacions d'inserció, cerca i eliminació d'elements, així com en la intersecció i unió de conjunts.
- Usar la classe conjunt anterior per a la resolució de problemes d'anàlisi i comparació de les paraules que apareixen en una sèrie de textos.

Abans de la sessió de laboratori, has de llegir el butlletí de pràctiques tractant de resoldre, en la mesura del possible, els problemes proposats.

2. Plantejament del problema

Es desitja desenvolupar una classe `ConjuntString` els objectes de la qual representen conjunts de cadenes de caràcters o `String`.

Aquesta classe es pot usar en la resolució de certs problemes de comparació de textos. Per exemple, donats dos fitxers de text, obtenir el conjunt de paraules comunes a tots dos textos, o el conjunt unió de les paraules de tots dos textos.

Els mètodes de la classe correspondran a les operacions bàsiques de conjunts, i els seus perfils i especificació vindran donats per:

```
/** Crea un conjunt buit. */
public ConjuntString()

/** Insereix s en el conjunt.
 *  Si s ja pertany al conjunt, el conjunt no canvia.
 *  @param s String. Element que s'insereix en el conjunt.
 */
public void inserir(String s)

/** Comprova si s pertany al conjunt.
 *  @param s String.
 *  @return true sii s pertany al conjunt.
 */
 public boolean pertany(String s)

/** Elimina s del conjunt.
 *  Si s no pertany al conjunt, el conjunt no canvia.
 *  @param s String.
 */
public void eliminar(String s)

/** Retorna la talla o cardinal del conjunt.
 *  @return la talla del conjunt.
 */
public int talla()

/** Retorna el conjunt interseccio del conjunt i d'altre.
 *  @param altre ConjuntString.
 *  @return el conjunt interseccio.
 */
public ConjuntString interseccio(ConjuntString altre)

/** Retorna el conjunt unio del conjunt i d'altre.
 *  @param altre ConjuntString.
 *  @return el conjunt unio.
 */
public ConjuntString unio(ConjuntString altre)
```

Una representació possible consisteix que cada objecte de la classe emmagatzeme en una seqüència enllaçada els elements que formen part del conjunt. Aquesta és l'estructura de dades que es va a desenvolupar en aquesta pràctica. En l'assignatura d'*Estructures de Dades i Algorismes* de segon curs es presentaran altres representacions més especialitzades i eficients.

En la pràctica es van a completar les següents tasques:

1. Desenvolupament de la classe `ConjuntString`. En concret, les seccions 3 i 4 d'aquesta pràctica es dediquen a la implementació enllaçada de la classe.

2. Desenvolupament d'un programa d'aplicació a la comparació de textos. La secció 5 es dedica a desenvolupar una aplicació de la classe com l'esmentada a l'inici d'aquesta secció.

Les successives activitats a completar es van proposant al llarg d'aquestes seccions.

3. Implementació de la classe `NodeString`

Per a poder desenvolupar seqüències enllaçades de `String` es va a implementar com a primera activitat una classe els objectes de la qual siguen els nodes de la seqüència. Cada node haurà de tenir per tant una dada de tipus `String`.

Activitat 1: Atributs i constructors de la classe `NodeString`

Crear un projecte BlueJ `pract5` específic per a la pràctica.

Dins d'aquest projecte es crearà una classe `NodeString` els atributs de la qual i mètodes constructors seran anàlegs als de la classe `NodeInt` vista en teoria, amb l'excepció que les dades seran de tipus `String`.

4. Implementació enllaçada de la classe `ConjuntString`

La informació dels objectes de la classe s'estructurà en els següents components:

- Una seqüència enllaçada que continga en els seus nodes els elements del conjunt, sense elements repetits, i que es mantindrà ordenada ascendentment per l'ordre de `String`.
- Un enter que mantinga el nombre d'elements en el conjunt, és a dir, el cardinal del conjunt.

L'ordenació de la seqüència es deu a raons d'implementació dels mètodes. Com s'anirà discutint en les següents activitats, a més d'afavorir la cerca d'elements en el conjunt, fonamentalment permetrà aplicar estratègies eficients al problema de la unió i la intersecció.

Activitat 2: Atributs i constructors de la classe `ConjuntString`

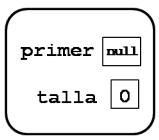
Agregar al projecte de la pràctica el fitxer `ConjuntString.java`, disponible en `PoliformaT` de `PRG`, en la carpeta `Recursos/Laboratorio/Práctica 5`.

La classe conté ja implementat un mètode `toString` que permet fer proves dels mètodes a mesura que es vagen desenvolupant.

Pel que s'ha comentat més amunt, en la classe s'han de declarar els atributs privats:

- `primer` de tipus `NodeString`, que donarà accés al primer element de la seqüència enllaçada en la qual es disposen ordenadament els elements del conjunt;
- `talla` de tipus `int`, que contindrà a cada moment el nombre d'elements en el conjunt.

El constructor crearà el conjunt buit, com es mostra en la figura:



Activitat 3: Implementació dels mètodes `inserir` i `talla`

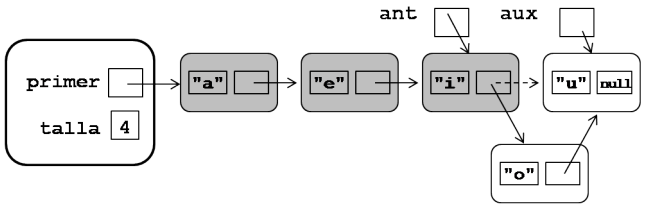
La primera operació a desenvolupar serà el mètode `inserir`, que permetrà anar construint un conjunt mitjançant successives insercions a partir del conjunt buit, mantenint l'ordenació de les dades. Cadascuna d'aquestes operacions haurà d'inserir el nou element en el lloc adequat de la seqüència i augmentar la talla del conjunt, sempre que aquest element no aparega prèviament en el conjunt.

En primer lloc, s'haurà de cercar el primer node en la seqüència amb una dada major o igual que `s` (segons la comparació de `String`), i davant del que caldrà inserir `s`.

Cal tenir present que si `s` ja apareix en la seqüència no cal inserir-lo, per la qual cosa es recomana controlar la cerca mitjançant una variable entera `compara` que s'actualitze passada a passada del bucle i que prenga un valor ≥ 0 quan es trobe un node amb una dada major o igual que `s`, tal com es mostra en el següent esquema de cos del mètode:

```
NodeString aux = this.primer; // node a revisar
NodeString ant = null; // anterior al node aux, inicialment null
int compara = -1; // valor inicial de la comparació
while (aux != null && compara < 0) {
    // comparar aux.dada amb s,
    // i si aux.dada és menor aleshores avançar en la seqüència
}
// Si no s'ha trobat un element igual a s, augmentar this.talla
// i inserir s en la seua posició:
//   - darrere d'ant si és diferent de null (l'últim element menor que s),
//   - a l'inici en cas contrari (cap element és menor que s).
```

Com en l'esquema típic de cerca en una seqüència enllaçada, el valor final de `ant` determina la posició darrere de la qual caldria inserir `s`, com en l'exemple de la següent figura, on s'han ombrejat els nodes pels quals ha avançat la cerca:



Cal tenir present que `s` no s'ha d'inserir si ja apareix en la seqüència, la qual cosa es pot comprovar mitjançant el valor de l'última comparació realitzada.

La implementació del mètode `talla` es resol retornant el valor de l'atribut del mateix nom.

Per a provar aquests mètodes, es recomana crear per exemple el conjunt de vocals mitjançant els següent passos:

1. Construir el conjunt buit en el banc de treball.
2. Inserir successivament els elements `"i"` (en el conjunt buit), `"a"` (davant del primer), `"u"` (darrere de l'últim), `"e"`, `"o"` (en posicions intermèdies).
3. Provar a inserir un element que ja està en el conjunt, `"u"` per exemple.

Inspeccionant l'objecte resultant de cada inserció, o amb els mètodes `toString` i `talla`, es comprovarà que el conjunt es crea correctament, i amb la talla que correspon.

Activitat 4: Implementació dels mètodes `pertany` i `eliminar`

La implementació d'aquests mètodes ha de tenir en compte que la seqüència d'elements està ordenada ascendentment.

El mètode `pertany` es pot escriure llavors com la cerca del primer node amb una dada major o igual que `s`. Acabada la cerca, cal retornar `true` sii la cerca ha acabat trobant una dada exactament igual a `s`.

El mètode `eliminar` pot seguir un esquema anàleg al de la inserció, llevat que s'eliminarà el node trobat solament si conté a `s` com a dada, i disminuint la talla en aquest cas.

Per a provar tots dos mètodes s'hauran de fer com a mínim les següents proves:

1. Construir el conjunt buit en el banc de treball i comprovar si `"a"` pertany al conjunt.
2. Afegir successivament al conjunt cadascun dels elements `"a"`, `"e"`, `"i"`, `"o"`, `"u"`, i comprovar si cada element que s'afig pertany al conjunt, abans i després d'inserir-lo.
3. En el conjunt de vocals obtingut per les insercions anteriors, provar a eliminar per aquest ordre els elements `"z"` (element que no pertany al conjunt), `"a"` (el primer), `"u"`(l'últim), `"i"` (posició intermèdia), `"e"`, `"o"` (elements restants), i de nou `"a"` (eliminació en el conjunt buit). Inspeccionant l'objecte resultant de cadascuna d'aquestes operacions, o amb els mètodes `toString` i `talla` es comprovarà que el conjunt s'actualitza correctament, i amb la talla que correspon.

Activitat 5: Implementació del mètode `interseccio`

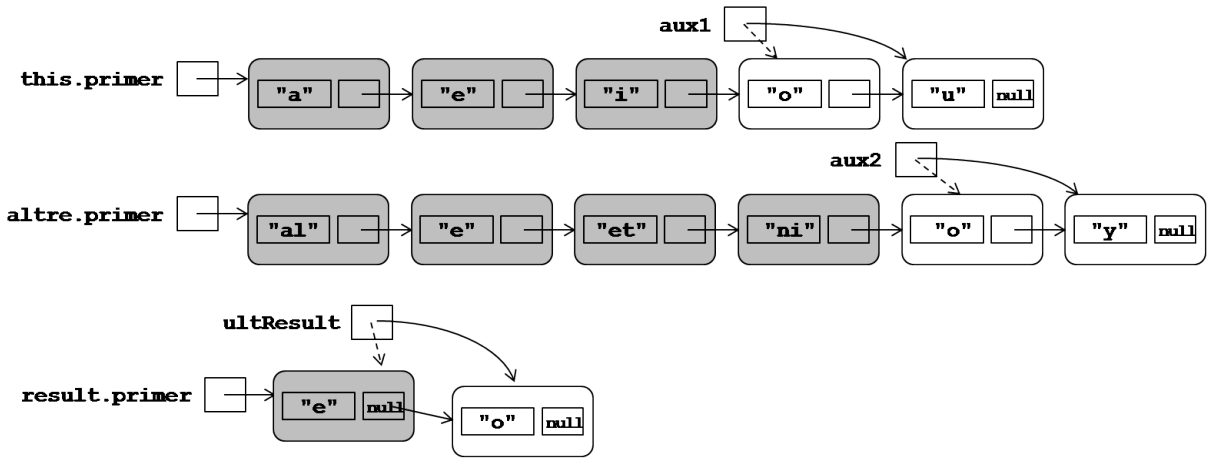
L'ordenació de les seqüències enllaçades amb la qual s'implementen els objectes de la classe, facilita que l'operació d'intersecció es puga resoldre amb un cost `O(this.talla + altre.talla)`, és a dir, recorrent com a molt una vegada els nodes de `this.primer` i `altre.primer`.

Per a açò es pot seguir una estratègia molt semblant a la de la *mescla natural* de dos arrays ordenats. És a dir, per a trobar els elements en comú es poden anar revisant les dades en el mateix ordre en el qual apareixen en les seqüències. Quan es trobe un element comú, s'incorporarà al final del conjunt resultant, mantenint així l'ordre preceptiu; a més, cal anar amb compte que els elements en comú només s'afigen una vegada al conjunt intersecció.

El cos del mètode pot seguir un esquema com el següent:

```
ConjuntString result = new ConjuntString();
NodeString aux1 = this.primer; // node a revisar del conjunt this
NodeString aux2 = altre.primer; // node a revisar d'altre
NodeString ultResult = null; // últim node de result, inicialment null
while (aux1 != null && aux2 != null) {
    // comparar les dades d'aux1 i aux2;
    // si són iguals:
    //    - inserir la dada darrere d'ultResult (a l'inici de result si és null),
    //    - avançar en les dues seqüències,
    //    - actualitzar la talla de result,
    // si no, avançar en la seqüència en la que apareix una dada menor.
}
```

L'exemple de la següent figura mostra un estat intermedi del càlcul de la intersecció de dos conjunts:



Quan s'esgota una de les dues seqüències, ja no pot haver-hi elements en comú, i s'ha d'acabar retornant el conjunt resultant construït.

Una vegada implementat el mètode, s'hauran de fer com a mínim les següents proves:

1. Crear dos conjunts buits `c1` i `c2` en el banc d'objectes i comprovar la seua intersecció.
2. Afegir a `c1` els elements `"a"`, `"c"` i `"e"`, i comprovar la intersecció amb `c2`.
3. Afegir a `c2` els elements `"o"`, `"p"` i `"q"`, i comprovar la intersecció amb `c1`.
4. Afegir a `c1` els elements `"d"`, `"f"`, i `"o"`, i a `c2` els elements `"d"`, i `"f"`, i comprovar la seua intersecció.
5. Comprovar la intersecció de `c1` amb si mateix.

Activitat 6: Implementació del mètode `unio`

La implementació del mètode d'unió dels conjunts també va a seguir una estratègia de mescla natural, per la qual cosa l'estructura serà molt semblant a la del mètode anterior.

Les diferències a tenir en compte resideixen en l'element que caldria afegir a `result` com a resultat de comparar les dades de `aux1` i `aux2`, i el corresponent avanç en les seqüències.

A més, quan una de les seqüències s'haja esgotat, quedarà per afegir a `result` els nodes restants de l'altra.

Una vegada implementat el mètode, s'hauran de fer proves anàlogues a les realitzades per al mètode d'intersecció.

5. Una aplicació de `ConjuntString` a la comparació de textos

En aquesta secció s'aplicarà el tipus de dades `ConjuntString` a la realització d'una xicoteta aplicació Java que permeta obtenir la unió o la intersecció de les paraules que es troben en dos fitxers de text.

En particular, donats dos fitxers de text: `f1.txt` i `f2.txt` es va a implementar una classe Java, `ComparaTextos`, que permetrà obtenir la unió o intersecció de les paraules en tots dos fitxers, mitjançant la seua execució[1] com:

```
$ java ComparaTextos opció f1.txt f2.txt
```

on `opció` és bé el modificador `-i`, bé `-u`, segons si es vol obtenir, respectivament, la intersecció o la unió dels fitxers `f1.txt` i `f2.txt`. El resultat de l'execució del comandament anterior (les paraules que són la unió o intersecció de les paraules en els fitxers) es mostra com resultat, en l'eixida estàndard.

Activitat 7: Revisió del material que es proporciona

Per a facilitar el treball anterior, es proporcionen dues classes Java en els seus fitxers font, perquè pugues revisar-les i, si és el cas, completar-les:

- `ParseString.java`. Aquesta classe es dóna completa, sols cal saber utilitzar-la.

  L'objectiu de la classe és poder descomposar, amb facilitat, una `String` qualsevol en les distintes paraules que la composen segons un grup de caràcters `separadors`, que identifiquen com limiten entre si les paraules.

  Per a usar-la és necessari crear, en primer lloc, un objecte de la classe `ParseString` i, a continuació, executar el seu mètode `separar(String)` que retornarà un array amb les distintes paraules contingudes en la `String` rebuda com argument. Per exemple:

  ```
  // Crear l'objecte ParseString (amb SEPARADORS predefinits):
  ParseString pS = new ParseString();
  // Obtenir un array amb les paraules en la String s1:
  String[] paraules1 = pS.separar(s1);
  // Obtenir un altre array amb les paraules en la String s2:
  String[] paraules2 = pS.separar(s2);
  ```

  És convenient provar en el banc d'objectes de BlueJ la funcionalitat que s'ha descrit per a assegurar-se del seu ús correcte.

- `ComparaTextos.java`. Aquesta classe es proporciona parcialment realitzada i s'ha de completar.

  És l'encarregada d'executar, mitjançant el seu `main` (que es dóna fet), el comandament que representa la classe `ComparaTextos`.

  A eixe respecte, convé recordar que quan s'executa el mètode `main` d'una classe (tant en l'entorn BlueJ com en mode comandament del sistema), es rep com argument un array de `String` amb les

[1] Per a executar el `main` de la classe com un comandament del sistema, és necessari obrir un terminal, situar-se en el directori on es troba el fitxer `bytecode` executable (`ComparaTextos.class`) i executar el comandament descrit.

que `main` pot operar, i que es corresponen a les cadenes de caràcters que apareixen a continuació del nom de la classe que s'executa.

Per exemple, quan s'executa en una terminal del sistema un comandament similar a l'esmentat anteriorment:

```
$ java ComparaTextos -i f1.txt f2.txt
```

el mètode `main` de la classe `ComparaTextos` rep com a argument un array[2] de tres elements `String`, els valors del qual en les posicions `0`, `1` i `2` són, respectivament, les `String`: `"-i"`, `"f1.txt"` i `"f2.txt"`.

En qualsevol cas, és necessari llegir el codi que apareix en el fitxer `ComparaTextos.java` detingudament, per a poder efectuar en ell les modificacions que es proposen en aquesta classe.

Activitat 8: Finalització de la classe `ComparaTextos`

Per a açò es recomana completar en primer lloc el mètode `private`:

```
/**
 *  Retorna el ConjuntString de les paraules llegides
 *  del Scanner s, segons els separadors donats per defecte
 *  en ParseString (ParseString.SEPARADORS).
 *  @param s Scanner.
 *  @return el conjunt de paraules llegides del Scanner s.
 */
private static ConjuntString lecturaConjunt(Scanner s)
```

l'objectiu del qual és llegir les línies que venen en el `Scanner s` que es rep com a argument, obtenint mitjançant la descomposició d'aquestes línies en les seues paraules (usant `ParseString.separar(String)`), el `ConjuntString` de les paraules que apareixen en aquest `Scanner`.

Fet l'anterior, poden realitzar-se els dos mètodes restants que, a partir dels noms dels fitxers, mostren en l'eixida estàndard el resultat de la intersecció o la unió. A manera d'exemple, es mostra el perfil del corresponent a l'obtenció de la intersecció:

```
/**
 *  Escriu en l'eixida estandard el resultat de la interseccio
 *  dels conjunts de paraules dels fitxers de text els
 *  noms del qual estan en nF1 i nF2.
 *  @param nF1 String, nom del primer fitxer.
 *  @param nF2 String, nom del segon fitxer.
 */
public static void interseccio(String nF1, String nF2) {
```

Allò que cadascun d'aquests mètodes ha de fer és:

1. Crear els objectes `File` i `Scanner` corresponents a cada nom de fitxer.
2. Crear els objectes `ConjuntString` amb els quals es va a operar. Aquests s'obtindran com a resultat de l'operació privada, realitzada abans, `lecturaConjunt(Scanner)`.
3. Obtenir el conjunt unió o intersecció segons el mètode que s'execute, aplicant l'operació corresponent de `ConjuntString`.
4. Mostrar aquest conjunt en l'eixida estàndard com a resultat.

[2] L'array `args` en el codi que es proporciona.

A més, en el procés anterior serà necessari tractar adequadament les excepcions d'entrada eixida necessàries, així com el tancament dels `Scanner` que s'hagen obert.

Activitat 9: Proves de funcionament de `ComparaTextos`

Algunes proves prèvies per a comprovar el funcionament correcte poden efectuar-se mitjançant la creació de fitxers de text que incloguen grups xicotets de paraules, per als quals s'obtindran les diferents combinacions possibles de la seua unió i intersecció. Aquesta tasca de comprovació se li deixa proposta a l'alumne.

Entre el material que es proporciona, es troba el fitxer `javaReserved.txt` que conté la llista de paraules reservades del llenguatge Java. Obtingueu les paraules reservades usades en `ComparaTextos.java`.

6. Avaluació

Aquesta pràctica forma part del segon bloc de pràctiques de l'assignatura que serà avaluada en el segon parcial de la mateixa. El valor d'aquest bloc és d'un 60 % respecte al total de les pràctiques. El valor percentual de les pràctiques en l'assignatura és d'un 20 % de la nota final.