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

Práctica 1. Resolució d'alguns problemes amb recursió

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Índex

1 Context i treball previ

Aquesta pràctica és la primera corresponent al tema 1 de l'assignatura, *"Recursió"*. En la mateixa se't proposen alguns problemes que hauràs de resoldre utilitzant el disseny recursiu. A més, hauràs d'elaborar els tests de proves necessaris per assegurar que les solucions dels problemes siguen correctes.

Perquè aprofites al màxim la sessió de pràctiques, has de realitzar abans una lectura comprensiva d'aquest butlletí. Nota que cada un dels problemes proposats exigeix certa anàlisi prèvia que has d'efectuar abans de començar la sessió de pràctiques. Així mateix és convenient que repasses les operacions de la classe `String`.

2 Exemple de tractament recursiu d'una String

Considera el problema de comptar el nombre de caràcters `'a'` en certa `String s`. Si fem una anàlisi per a una solució recursiva d'aquest, tenim:

- Cas base: la `String s` és buida, de manera que el nombre de `'a'`s que conté és 0.
- Cas general: la `String s` no és buida, per la qual cosa:
  - si el primer caràcter de `s` és `'a'`, llavors el total de `'a'`s és 1 més les `'a'`s que continga la substring de `s` obtinguda a partir del caràcter d'índex 1 en endavant,
  - en canvi, si el primer caràcter no és `'a'`, llavors el total de `'a'`s és les que continga la substring de `s` des del caràcter d'índex 1 en endavant.

Podem implementar l'anàlisi anterior mitjançant el mètode següent:

```
/**
 * Torna el nombre de 'a's en la String que rep.
 * @param s String en la qual es volen comptar les 'a's.
 * @return int.
 */
public static int comptaAs(String s) {
    // Cas base: String buida
    if (s.length() == 0) { return 0; }
    // Cas general: String no buida. Tractar la substring posterior
    else if (s.charAt(0) == 'a') { return 1 + comptaAs(s.substring(1)); }
    else { return comptaAs(s.substring(1)); }
}
```

A efectes comparatius, es pot resoldre el problema anterior de forma semblant a com s'han resolt problemes similars en teoria, utilitzant un paràmetre posicional que indique on es considera que comença la part de la `String` sobre la qual es treballa. L'anàlisi de casos és similar a l'anterior. En la seua implementació, que pots veure a continuació, s'ha de tenir en compte les implicacions de l'ús del nou paràmetre (`pos`):

```
/**
 * Torna el nombre de 'a's en la String que rep.
 * @param s String en la qual es volen comptar les 'a's.
 * @param pos posició en s on comença la substring.
 * @return int.
 * PRECONDICIÓ: pos >= 0
 */
public static int comptaAs(String s, int pos) {
    // Cas base: String buida
    if (pos >= s.length()) { return 0; }
    // Cas general: String no buida. Tractar la substring posterior
    else if (s.charAt(pos) == 'a') { return 1 + comptaAs(s, pos + 1); }
    else { return comptaAs(s, pos + 1); }
}
```

Naturalment, es pot fer una anàlisi i solució pràcticament idèntics als anteriors, però considerant l'últim caràcter de la `String` en lloc del primer, així com la substring inicial en lloc de la final.

El mètode següent implementa aquesta anàlisi. L'estudi dels casos apareix reflectit en els comentaris del mètode.

```
/**
 * Torna el nombre de 'a's en la String que rep.
 * @param s String en la qual es volen comptar les 'a's.
 * @return int.
 */
public static int comptaAs2(String s) {
    // Cas base: String buida
    if (s.length() == 0) { return 0; }
    // Cas general: String no buida. Tractar la substring anterior
    else if (s.charAt(s.length() - 1) == 'a') {
        return 1 + comptaAs2(s.substring(0, s.length() - 1));
    }
    else { return comptaAs2(s.substring(0, s.length() - 1)); }
}
```

Recorda que `s.substring(i, j)` és un objecte `String` que representa la substring de `s` formada amb els caràcters compresos entre el `i` i el `j - 1`.

3 Treball al laboratori. Entorn

Has de crear un projecte *BlueJ* específic per a aquesta pràctica. Crea en ell una classe llibreria `PRGString` que incloga els mètodes que resolen els problemes que se't plantegen a continuació.

4 Problema A. *Prefixe*

Donades dues `String`s `a` i `b`, potencialment buides, es diu que `a` és *prefixe* de `b` quan tots els caràcters de `a` estan consecutius, en el mateix ordre original, al començament de `b`.

Conseqüència de la definició anterior és que la *cadena buida* és prefixe de qualsevol altra, fins i tot si aquesta altra també estigués buida. Nota, d'altra banda, que una cadena no pot ser prefixe d'una altra si la primera és de longitud més gran que la segona.

4.1 Activitats inicials

Has definir recursivament un mètode `esPrefixe`, mitjançant el qual es podrà comprovar si una cadena és prefixe d'una altra. Has d'establir els casos base i general de la recursió definint, a més, la solució del problema en cada un d'aquests casos. La capçalera del mètode haurà de ser necessàriament:

```
public static boolean esPrefixe(String a, String b)
```

Nota que no hi ha paràmetres posicionals en la capçalera anterior.

4.2 Activitats al laboratori

1. Basant-te en la teua anàlisi prèvia escriu el mètode `esPrefixe`,
2. documenta adequadament el mètode, explicitant quins són els seus paràmetres, el tipus del seu resultat i, cas d'haver-ne, la seva precondició,
3. comprova que el mètode que has realitzat funciona correctament, per al que hauràs de:
   - Realitzar una llista de les diferents situacions que es poden donar en la seva execució, com ara per exemple: que ambdues cadenes estiguen buides, que ho estigui només una, que la primera cadena siga més llarga que la segona, que la primera cadena siga prefixe o no de la segona, etc.,
   - provar que el mètode funciona correctament en cadascuna de les situacions anteriors usant BlueJ. Per això últim pots comparar el resultat del teu mètode amb el de l'estàndard (a la classe `String`) `startsWith(String)`.

5 Problema B. *Subcadena*

Donades dues `String`s `a` i `b`, potencialment buides, es diu que `a` és *subcadena* de `b` quan tots els caràcters de `a` estan consecutius, en el mateix ordre original, en algun lloc de `b`. O, el que és el mateix, quan `a` és prefixe de `b` o d'alguna de les possibles subcadenes de `b`.

Naturalment, com passava en el cas de `esPrefixe`, es pot veure que la *cadena buida* és subcadena de qualsevol altra, encara que aquella altra també estigués buida. A més, una cadena no pot ser subcadena d'una altra si la primera és de longitud més gran que la segona.

5.1 Activitats inicials

Has de definir recursivament, **en termes de** `esPrefixe`, el mètode `esSubcadena`, per poder comprovar si una cadena és subcadena d'una altra. Enuncia els casos base i general de la recursió, definint la solució del problema en cada cas. La capçalera del mètode ha de ser necessàriament:

```
public static boolean esSubcadena(String a, String b)
```

Nota que, igual que per a l'operació `esPrefixe`, no hi ha paràmetres posicionals en la capçalera anterior.

5.2 Activitats al laboratori

1. Basant-te en la teua anàlisi escriu un mètode que implemente l'operació `esSubcadena`,
2. documenta'l adequadament, explicitant els seus paràmetres i resultat així com, cas d'haver-la, la seva precondició,
3. comprova que el mètode funciona correctament, pel que hauràs, com abans, de fer una llista de les diferents situacions que es poden donar, provant que en totes elles funciona adequadament. En aquest cas, pots comparar el resultat amb el del mètode estàndard `contains(String)`.

6 Problema C. *Palíndrom*

Ja coneixes que una certa frase (`String`) és un *palíndrom* quan la mateixa **es llegeix** igualment en un sentit (d'esquerra a dreta) que en l'altre (de dreta a esquerra). Així, la cadena: "I callada posa la sopera. Ja reposa la sopa d'all ací" és un palíndrom como també ho es: "A una nena nua llepa-li la pell, llepa-li la pell a una nena nua" o la prou coneguda "Senén té sis nens i set nenes".

Nota que l'existència d'espais en blanc, signes de puntuació, apòstrofs i guions, accents, així com caràcters en majúscules i minúscules **no són significatius** per determinar si una frase és o no un palíndrom.

6.1 Activitats al laboratori

Has de dissenyar un mètode recursiu en Java que permeta conèixer si una frase com qualsevol dels exemples anteriors, és o no un palíndrom. La capçalera que has d'utilitzar és la següent:

```
public static boolean esPalindrom(String a)
```

Per resoldre el problema has de seguir una tècnica similar a les que has utilitzat per resoldre qualsevol dels problemes anteriors que, en resum, consisteix en:

- Definir recursivament la solució del problema,
- documentar la solució,
- provar l'operació exhaustivament.

Nota: El mètode que construeixes no ha de processar prèviament la `String`, és a dir, s'ha de garantir que es fa un sol recorregut sobre tota la frase.

Com s'ha dit, per simplicitat, pots considerar que els únics caràcters significatius per veure si una frase és o no un palíndrom són els alfabètics habituals, és a dir, els compresos entre la 'a' i la 'z' amb la 'ç' (en majúscules o minúscules), mentre que pots tractar la resta de caràcters possibles com si fossin irrellevants (no pronunciables o espais en blanc). En Java és possible conèixer si cert caràcter és o no una lletra mitjançant el mètode de llibreria: `Character.isLetter(char)`.

D'altra banda, també pots considerar **inicialment** que les frases no tenen accents. Un cop el tingues resolt així, pots ampliar la teua solució per a que sí que tinga en compte els caràcters accentuats.

7 Avaluació

Aquesta pràctica forma part del primer bloc de pràctiques de l'assignatura que serà avaluat en el primer parcial d'aquesta. El valor d'eixe bloc és d'un 40% respecte al total de les pràctiques. El valor percentual de les pràctiques en l'assignatura és d'un 20% de la nota final.