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

*Pràctica 2. 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 segona 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 de realitzar les proves necessàries, estudiant els casos possibles, per assegurar-te 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`

Suposa que disposem de certa `String s`, entre els caràcters de la qual poden existir dígits (caràcters compresos entre el `'0'` i el `'9'`) i es planteja el problema de determinar la suma dels valors d'aquests dígits. Per exemple, davant la `String "0ac320dX16?"` obtindríem, com a resultat en la resolució del problema, el valor `12`.

Si fem una anàlisi per a una solució recursiva del problema, es té:

- Cas base: la `String s` és buida, per la qual cosa la suma del valor dels seus dígits és 0,
- Cas general: la `String s` no és buida, per la qual cosa:
  - si el primer caràcter de `s` és un dígit, llavors la suma total demanada serà **el valor numèric** del dígit contingut en aquesta posició més la suma que s'obtinga dels dígits a partir del caràcter d'índex 1 endavant,
  - en canvi, si el primer caràcter no és un dígit, llavors la suma total serà la que s'obtinga dels dígits a partir del caràcter d'índex 1 endavant.

Podem implementar l'anàlisi anterior mitjançant el mètode següent:

```
/** Torna la suma dels dígits continguts en la String que rep.
 *  @param s String, cadena de caràcters sobre els quals s'opera.
 *  @return int. */
public static int sumaDigits(String s) {
    // Cas base: String buida
    if (s.length() == 0) { return 0; }
    else {
    // Cas general: String no buida. Tractar substring posterior.
        char c = s.charAt(0);
        if (c >= '0' && c <= '9') {
            return (c - '0') + sumaDigits(s.substring(1));
        }
        else { return sumaDigits(s.substring(1)); }
    }
}
```

Nota que l'expressió `c - '0'` permet calcular **el valor numèric** del dígit en `c`[1]. Una forma alternativa per fer el mateix consisteix en utilitzar `Character.getNumericValue(c)`.

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`):

[1] Ja que qualsevol caràcter, `c`, es representa internament com el enter corresponent al seu codi, la diferència entre la codificació de qualsevol dígit i la del 0, és la distància en nombre de caràcters entre 0 i eixe dígit, és a dir, el valor com a nombre del dígit.

```
/** Torna la suma dels dígits en la String s des de la posició pos.
 *  @param s String, cadena de caràcters sobre els quals s'opera.
 *  @param pos int, posició en s des d'on es treballa.
 *  @return int.
 *  PRECONDICIÓ: pos >= 0. */
public static int sumaDigits(String s, int pos) {
    // Cas base: String buida
    if (pos >= s.length()) { return 0; }
    else {
    // Cas general: String no buida. Tractar substring posterior.
        char c = s.charAt(pos);
        if (c >= '0' && c <= '9') {
            return (c - '0') + sumaDigits(s, pos + 1);
        }
        else { return sumaDigits(s, pos + 1); }
    }
}
```

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.

```
/** Torna la suma dels dígits continguts en la String que rep.
 *  @param s String, cadena de caràcters sobre els quals s'opera.
 *  @return int. */
public static int sumaDigits2(String s) {
    // Cas base: String buida
    if (s.length() == 0) { return 0; }
    else {
    // Cas general: String no buida. Tractar substring anterior.
        int len = s.length();
        char c = s.charAt(len - 1);
        if (c >= '0' && c <= '9') {
            return (c - '0') + sumaDigits(s.substring(0, len - 1));
        }
        else { return sumaDigits(s.substring(0, len - 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 (`$HOME/DiscoW/prg/pract2`). Crea en ell una classe llibreria `PRGParseNum` que incloga els mètodes que resolen els problemes que se't plantegen a continuació.

# 4 Problema A: *Obtenció del* `int` *representat amb una* `String`

Considera certa `String s` que conté els dígits d'un nombre enter, com pot ser per exemple: `"112"`, `"-2359"` o `"0"`. Es planteja el problema d'obtenir l'enter corresponent que, per als exemples anteriors seria, respectivament, `112`, `-2359` i `0`.

Podem simplificar aquest problema, resolent en primer lloc el problema de la lectura d'una `String` que emmagatzeme un enter sense signe per a, a continuació, resoldre el problema quan pot existir (o no) un signe inicial. Es tracta, per tant, que facis els dos mètodes següents :

## 4.1 Part 1, mètode `parseUnsignedInt(String s)`

On, donada certa `String s`, no buida, que conté els dígits d'un nombre enter major o igual a 0, haurà de tornar el valor `int` corresponent.

### Activitats inicials

Has de definir **recursivament** el mètode `parseUnsignedInt(String s)`, mitjançant el qual s'obtindrà un enter positiu a partir d'una `String` que contindrà els dígits del mateix. L'esmentada `String` contindrà, almenys, un caràcter.

Pots suposar que l'enter que conté la `String` està adequadament format; això és, no té errors, com ara caràcters que no siguen dígits, blancs, etc.

Has d'establir els casos base i general de la recursió definint, a més, la solució del problema en cada un d'aquests casos.

Nota que la descomposició recursiva, en el cas general, d'una `String` així, `s`, pot tenir una de les dues formes següents:

- `s.charAt(0)` + `s.substring(1)`, això és: un caràcter i la subcadena restant,
- `s.substring(0, s.length() - 1)` + `s.charAt(s.length() - 1)`, això és: una subcadena fins al penúltim caràcter i l'últim caràcter.

La capçalera del mètode que facis haurà de ser necessàriament :

```
/** PRECONDICIÓ: s conté un enter, >= 0, ben format. */
public static int parseUnsignedInt(String s)
```

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

### Activitats al laboratori

1. Basant-te en la teva anàlisi prèvia escriu el mètode `parseUnsignedInt(String s)`,
2. documenta adequadament el mètode, explicitant quins són els seus paràmetres, el tipus del seu resultat i, en cas d'haver-la, la seua 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 seua execució, com ara per exemple: que la cadena continga un sol dígit o més d'un dígit, que el valor siga 0, 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 `Integer`) `parseUnsignedInt(String)`.

## 4.2 Part 2, mètode `parseInt(String s)`

Si l'enter representat en la `String` pot tenir signe, pot ocórrer que la `String` tinga com a primer caràcter un signe menys (`-`) , per la qual cosa el nombre serà negatiu; que tinga un signe més (`+`) o que no en tinga, sent en aquests dos últims casos el nombre positiu.

**Activitats al laboratori**

Implementa un mètode amb capçalera:

```
/** PRECONDICIÓ: s conté un enter, pot ser amb signe, ben format. */
public static int parseInt(String s)
```

Per a això:

1. Escriu el mètode (fent una anàlisi de casos) utilitzant `parseUnsignedInt(String s)`,
2. documenta'l adequadament, explicitant els seus paràmetres i resultat així com la seua precondició,
3. comprova que el mètode funciona correctament, per al 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 de nou el resultat amb el del mètode estàndard `Integer.parseInt(String s)`.

# 5 Problema B: *Obtenció del* `double` *representat amb una* `String`

El càlcul del nombre en coma flotant contingut en una `String` caràcter a caràcter, pot efectuar-se mitjançant el fraccionament d'aquesta `String` en les diferents parts de les que pot constar un nombre així: **part entera**, **fraccionària** i **exponent**[2].

Per simplificar el plantejament, resoldràs el problema quan el nombre no està en format científic, és a dir, quan està compost ben sol d'una part entera o bé d'una part entera i una altra fraccionària separades entre si mitjançant un punt (`.`) o una coma (`,`). Exemples d'ambdós formats són, respectivament: `"39073"`, `"23,6729"` o `"-0.0112"`. Els valors que, en cada cas, s'hauran d'obtenir són, respectivament: `39073`, `23.6729` i `-0.0112`

Per això has de resoldre, prèviament, els dos subproblemes següents:

[2]**NOTA:** d'ara en endavant entendrem que els valors per la part entera i l'exponencial (d'existir) d'un nombre en coma flotant bé format, estaran restringits als d'un valor `int` Java vàlid; mentre que el corresponent a la part fraccionària, si existeix, estarà restringit a un valor `int` Java vàlid sense signe.

## 5.1 Part 1, posició del separador de la part fraccionària

Es tracta de determinar **recursivament** la posició en una `String` del primer caràcter punt (`'.'`) o coma (`','`) que aparega; retornant -1 en cas que aquest no existeixca.

### Activitats inicials

Has de definir un mètode recursiu `posFracSep(String s)`, mitjançant el qual s'obtindrà la posició en una `String`, `s`, del primer caràcter punt (`'.'`) o coma (`','`) que aparega; retornant -1 si aquest no existeix. La capçalera del mètode que facis haurà de ser necessàriament:

```
/** PRECONDICIÓ: s conté un nombre en coma flotant ben format. */
public static int posFracSep(String s)
```

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

Has d'establir els casos base i general de la recursió definint, a més, la solució del problema en cada un d'aquests casos.

### Activitats al laboratori

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.

## 5.2 Part 2, obtenció del valor associat a una part fraccionària

Suposa que disposem dels dígits de la part fraccionària d'un nombre en coma flotant com caràcters en una `String` i desitgem obtenir el valor en coma flotant corresponent. Per exemple, suposa que tenim la `String`: `"0325671"` que representa la part fraccionària de cert nombre, llavors, el mètode que facis haurà de tornar el nombre `double 0.0325671`.

### Activitats al laboratori

Implementa un mètode amb capçalera:

```
/** PRECONDICIÓ: s conté els dígits de la part fraccionària
 *  d'un nombre double ben format. */
public static double parseFrac(String s)
```

Fixa't que aquest problema el pots resoldre fàcilment utilitzant el mètode que has realitzat abans `parseUnsignedInt(String s)` per llegir el valor com a un enter i, a continuació, dividint el valor enter obtingut per la potència de 10 corresponent.

### 5.3 Part 3, mètode `parseDouble(String s)`

Tenint en compte el que s'ha dit anteriorment, això és que un valor en coma flotant pot estar format per només una part entera, o per una part entera i una fraccionària separades mitjançant un caràcter de separació (coma o punt), és senzill resoldre el problema inicial: obtenció del valor `double` representat mitjançant una `String`.

Exemples d'alguns casos possibles, són: `"2345"`, el valor calculat és `2345`; `"312,49"` amb valor calculat `312.49` (això és, **la suma** de `312` i `0.49`) i `"-312.49"` amb valor calculat `-312.49` (això és, **la resta** de `-312` i `0.49`).

#### Activitats al laboratori

Implementa un mètode amb capçalera:

```
/** PRECONDICIÓ: s conté un double, pot ser amb signe i/o
 *  part fraccionària, ben format. */
public static double parseDouble(String s)
```

Per això:

1. Dissenya (realitzant una anàlisi de casos) i escriu aquest mètode utilitzant els mètodes que et siguen necessaris d'entre els que has resolt prèviament,
2. documenta'l adequadament, explicitant els seus paràmetres i resultat així com la seua precondició,
3. comprova que el mètode funciona correctament, per al 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 de nou comparar el resultat amb el del mètode estàndard `Double.parseDouble(String s)`. **Assegura't** de realitzar proves amb valors negatius (com el de l'exemple anterior).

## 6 Ampliacions recomanades

Els següents són problemes interessants que convé que et planteges:

1. Implementa un mètode recursiu que determine la posició del caràcter `'e'` o `'E'` en una `String s` que conté un valor `double` ben format. Aquest caràcter indica l'aparició d'un exponent en un nombre representat en format científic.
2. Millora `parseDouble(String s)` perquè admeta notació científica. Exemples possibles són: `"1e10"`, `"0.003141592e3`, `"3141592E-05"` i `"-3923,12804e-33"`.
   **Suggeriment:** Realitza una anàlisi de casos.
3. Dissenya i implementa **recursivament** el mètode `parseFrac(String s)` mitjançant el qual s'obté el valor en coma flotant associat a una part fraccionària. **Suggeriment:** Utilitza per a la `String s`, la descomposició recursiva en la que la cadena es veu com un caràcter i la subcadena restant: `s.charAt(0)` + `s.substring(1)`. Comprova'l exhaustivament.

4. Dissenya i implementa **recursivament** el mètode:

```
/** PRECONDICIÓ: s conté un enter, >= 0, ben format en
 *  base radix. 10 >= radix >= 0. */
public static int parseUnsignedInt(String s, int radix)
```

   Similar al que ja has realitzat: `parseUnsignedInt(String s)`, però que té en compte la base de numeració (`radix`) utilitzada en el nombre que conté `s`. Per exemple, per al número binari: `"1001110110"`. El mètode recursiu haurà de retornar el valor `630`. Compara la seua execució amb la del mètode estàndard: `Integer.parseUnsignedInt(String s, int radix)`.

5. Utilitza el mètode anterior per implementar el mètode:

```
/** PRECONDICIÓ: s conté un enter ben format, en base radix.
 *  10 >= radix >= 0. */
public static int parseInt(String s, int radix)
```

   Compara'l amb `Integer.parseInt(String s, int radix)`.

6. Dissenya i implementa **recursivament** el mètode:

```
public static String toString(int i)
```

   que ha de tornar la representació com `String` de l'enter `i`. Pots comparar el seu resultat amb el que s'obté mitjançant un dels mètodes del Java: `Integer.toString(int i)` o `String.valueOf(int i)`.

   **Suggeriment:** Defineix primer un mètode per resoldre el problema per nombres sense signe. Defineix després un nou mètode que, utilitzant l'anterior, resolga el problema general.

7. Dissenya i implementa **recursivament** el mètode:

```
/** PRECONDICIÓ: 10 >= radix >= 0. */
public static String toString(int i, int radix)
```

   que ha de tornar la representació com `String` del enter `i` en base `radix`. Pots provar-lo, comparant-lo amb el mètode estàndard: `Integer.toString(int i, int radix)`.

   **Suggeriment:** El mateix que en el problema anterior.

# 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.