PRG (ETS de Ingeniería Informática) - Curso 2022-2023

Práctica 2. Resolución recursiva de algunos problemas de `String`

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Índice

1. Contexto y trabajo previo

En esta práctica se propone la resolución de forma recursiva de dos problemas con `String`s. Para ello, se diseñarán los métodos correspondientes y las clases de prueba para asegurar que las soluciones de los problemas sean correctas.

Es conveniente haber estudiado la sección 10.6 *Recursividad con objetos de tipo* `String` de la 3ª edición del libro de la asignatura[1] y haber comprendido algunos ejemplos como el problema de contar el número de caracteres `'a'` en cierta `String s`.

Actividad 1: creación del paquete BlueJ `pract2`

Abre el proyecto *BlueJ* de trabajo de la asignatura (`prg`) y crea un nuevo paquete `pract2`. Agrega al paquete los ficheros `PRGString.java` y `TestPract2.java` que habrás descargado previamente de la carpeta `Recursos/Laboratorio/Práctica 2` de PoliformaT de PRG. La

[1] Si tienes la 2ª edición: sección 11.6.

clase `PRGString` es una clase de utilidades que incluye los métodos que resuelven el problema de contar el número de `'a'`s en una `String s` (sección 10.6 del libro) y los métodos (a completar) que resuelven los problemas que se te plantean a continuación. En posteriores actividades, se completará la clase `TestPract2` para realizar pruebas de los métodos de `PRGString` implementados.

2. Problema A. *Prefijo*

Dadas dos `String`s `a` y `b`, potencialmente vacías, se dice que `a` es *prefijo* de `b` cuando todos los caracteres de `a` están consecutivos, en el mismo orden original, al comienzo de `b`.

Consecuencia de la definición anterior es que la *cadena vacía* es prefijo de cualquier otra, incluso si esa otra también estuviese vacía. Nota, por otra parte, que una cadena no puede ser prefijo de otra si la primera es de longitud mayor que la segunda.

Actividad 2: método `isPrefix`

Define recursivamente un método `isPrefix` para comprobar si una cadena es prefijo de otra. Para ello:

- Establece los casos base y general de la recursión definiendo, además, la solución del problema en cada uno de dichos casos. La cabecera del método (en la que no hay parámetros posicionales) debe ser necesariamente la que sigue:

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

  de manera que devuelva `true` si `a` es prefijo de `b`, y `false` en caso contrario.
- Comprueba que el código del método sigue las normas de estilo. Puedes encontrarlas en la carpeta `Recursos/Laboratorio` de PoliformaT de PRG.

Actividad 3: validación del método `isPrefix`

La clase `TestPract2` que se ha añadido al paquete va a permitir hacer pruebas del método `isPrefix`.

Para ello, se probará el método con una batería de datos que reflejen las distintas situaciones que se pueden dar, tales como, por ejemplo: que ambas cadenas estén vacías, que lo esté solo una de ellas, que la primera cadena sea más larga que la segunda, que la primera cadena sea prefijo o no de la segunda, etc.

En la tabla siguiente se detallan los diferentes casos, con instancias concretas y el resultado esperado para cada caso.

En concreto, se debe completar el método `testIsPrefix` de la clase, en el cual se declara el siguiente array

```
String[] s = {"", "rec", "pecur",
    "recurso", "remursi",
    "123456789", "recursion"};
```

Caso	`a`	`b`	**Resultado**
1. `a` y `b` vacías	`""`	`""`	`true`
2. Solo `a` vacía	`""`	`"recursion"`	`true`
3. Solo `b` vacía	`"recursion"`	`""`	`false`
4. `a` de mayor longitud que `b`	`"recursion"`	`"rec"`	`false`
5. `a` y `b` de igual longitud y `a` es prefijo de `b`	`"recursion"`	`"recursion"`	`true`
6. `a` y `b` de igual longitud y `a` no es prefijo de `b`	`"123456789"`	`"recursion"`	`false`
7. `a` de menor longitud que `b` y `a` es prefijo de `b`	`"rec"`	`"recursion"`	`true`
8. `a` de menor longitud que `b` y `a` no es prefijo de `b`:			
- 8a. Por el primer carácter	`"pecur"`	`"recursion"`	`false`
- 8b. Por el último carácter	`"recurso"`	`"recursion"`	`false`
- 8c. Por un carácter intermedio	`"remursi"`	`"recursion"`	`false`

cuyas componentes forman un conjunto de palabras suficiente para probar los casos anteriores. Además, en la clase se proporciona un método privado auxiliar `compareIsPrefix(String a, String b)`, que muestra el resultado de `PRGString.isPrefix(a, b)` y el de `b.startsWith(a)`; este último es el método de la clase `String` que comprueba si los caracteres de `a` aparecen al inicio de `b`. Notar que, para que estos resultados aparezcan adecuadamente tabulados, se usa `System.out.printf`, cuyo primer argumento describe el formato con el que se escribirán los siguientes argumentos (incluyendo los espacios que ocupan).

De esta forma, por ejemplo, la prueba del caso 7 de la tabla anterior se puede obtener con la llamada `compareIsPrefix(s[1], s[6])`.

3. Problema B. *Subcadena*

Dadas dos `String`s `a` y `b`, potencialmente vacías, se dice que `a` es *subcadena* de `b` cuando todos los caracteres de `a` están consecutivos, en el mismo orden original, en algún lugar de `b`. O, lo que es lo mismo, cuando `a` es prefijo de `b` o de algún `substring` de `b`.

Naturalmente, igual que ocurría en el caso de `isPrefix`, se puede ver que la *cadena vacía* es subcadena de cualquier otra, incluso si esa otra también estuviese vacía. Además, una cadena no puede ser subcadena de otra si la primera es de longitud mayor que la segunda.

Actividad 4: método `isSubstring`

Define recursivamente, **en términos de** `isPrefix`, el método `isSubstring`, para poder comprobar si una cadena es subcadena de otra. Para ello:

- Enuncia los casos base y general de la recursión, definiendo la solución del problema en cada caso. La cabecera del método deberá ser necesariamente:

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

  de manera que devuelva `true` si `a` es subcadena de `b`, y `false` en caso contrario.

  Nota que, al igual que para la operación `isPrefix`, no hay parámetros posicionales en la cabecera anterior.

- Comprueba que el código del método sigue las normas de estilo. Puedes encontrarlas en la carpeta `Recursos/Laboratorio` de PoliformaT de PRG.

Actividad 5: validación del método `isSubstring`

De manera análoga a la actividad 3, se debe completar el método `testIsSubstring` de la clase `TestPract2`.

En este método se declara un array de `String` con una serie de palabras suficiente para probar los casos de `isSubstring`. De igual manera, en la clase se nos proporciona el método auxiliar `compareIsSubstring(String a, String b)`, que escribe en la salida el resultado de `PRGString.isSubstring(a, b)` y de `b.contains(a)`; este último es el método de `String` que comprueba si `b` contiene a `a`, es decir, si `a` es subcadena de `b`.

4. Evaluación

Esta práctica forma parte del primer bloque de prácticas de la asignatura que será evaluado en el primer parcial de la misma. El valor de dicho bloque es de un 50 % con respecto al total de las prácticas. El valor porcentual de las prácticas en la asignatura es de un 25 % de su nota final.