Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Algoritmos y Estructuras de Datos

CC3001

Tarea 2

Expansión

Autor: Ilana Mergudich Thal
E-Mail: ilanamergudich@gmail.com

Fecha de entrega: 24 de abril de 2017

Índice de Contenidos

1. Introducción

En computación, muchos códigos hacen referencias a otros archivos, por lo que surge la necesidad de "expandirlos", es decir, imprimir los contenidos de un archivo, incluyendo directamente los contenidos de los archivos a los que se hace referencia en este. El objetivo de esta tarea es lograr "expandir" una lista de archivos. Para ello, se leerá línea por línea y en cada una se iterará sobre las palabras, verificando si ellas son archivos. En caso de serlo, estos se leerán de manera recursiva y sino, simplemente se imprimirán.

2. Análisis del problema

El problema consiste en expandir un número de archivos, esto es, imprimir sus contenidos y si se hace referencia a otro archivo, insertar esos contenidos donde corresponde. Para hacer una referencia se debe poner el nombre de la siguiente forma: <<<archivo>>>.

Por ejemplo, si se tienen los siguientes archivos:

1. **El archivo1 dice:**
   Este es el primer archivo. \n
   En la siguiente línea se encuentra el archivo2: \n
   <<<archivo2>>> \n
   Fin del archivo1.

2. **El archivo2 dice:**
   "Este es el segundo archivo".

al implementar el programa para

```
archivo1 archivo2
```

se debe mostrar lo siguiente:

Este es el primer archivo. \n
En la siguiente línea se encuentra el archivo2: \n
"Este es el segundo archivo". \n
\n
Fin del archivo1. \n
"Este es el segundo archivo". \n

Se observa que se expandió primero el archivo1, que incluye al archivo2, y luego se imprimieron los contenidos del segundo archivo.

Para lograr estas lecturas se creará una función (**lector()**) que expanda un archivo y luego se creará otra (**lectorVarios()**) que aplique **lector()** a cada uno de los archivos que se ingresen.

Para la función **lector()** se leerá el archivo mediante el uso de las clases **FileReader** y **BufferedReader**. Se leerá cada linea y a su vez se revisará cada palabra de la línea. Si una palabra no es un archivo, esta se guardará en un string que se imprimirá en **lectorVarios()**, en caso contrario se aplicará **lector()** de manera recursiva y el resultado se agregará al string final. **lectorVarios** imprimirá todos los strings entregados por **lector()** para cada archivo. La función **lector()** crea un string en lugar de imprimir directamente para que, en caso de que haya un error, no se expanda ningún archivo, sino que sólo se entregue el mensaje de error.

Es importante considerar que podría producirse un "loop" de referencias entre los archivos. Por ejemplo, si se tuvieran los siguientes:

1. **El archivo3 dice:**
   El archivo3 contiene al archivo4 aquí: <<<archivo4>>>

2. **El archivo4 dice:**
   El archivo4 contiene al archivo3 aquí: <<<archivo3>>>

En este caso, el programa debe detenerse para evitar un StackOverFlowException e indicar que se produjo un error. Para lograr esto, se creará una lista de *leidos*, donde se guardarán los archivos que vayan pasando por **lector()**. Antes de aplicar esta función recursivamente, se verificará si el archivo se encuentra en esta lista. No obstante esto no basta, ya que podría ocurrir que un archivo fuese referenciado dos veces, sin producirse un "loop". Por lo tanto, cuando cada archivo termine de ser leído, será quitado de la lista.

Se asumirá que el usuario entrega los archivos de manera correcta, según lo especificado en "Modo de uso" más adelante.

3. Solución del problema

3.1. Clases

3.1.1. FileReader y BufferedReader

Estas clases permiten abrir y leer un archivo respectivamente. Ambas se encuentran en la librería IO de Java.

3.1.2. Expand

Esta clase lleva a cabo el proceso descrito anteriormente para solucionar el problema. En ella se crea una lista de strings que contendrá los archivos leídos. Además, contiene cuatro métodos: **isArchivo**, **lector**, **lectorVarios** y **main**, que serán explicados en la siguiente sección.

3.2. Métodos

3.2.1. isArchivo

Este método recibe un string y verifica si este hace referencia a un archivo o no, entregando el boolean correspondiente. Para ello, recorre las primeras y últimas tres posiciones del string y verifica si corresponden a '<' y '>' respectivamente. El código de este método se encuentra en el anexo 6.1

Ejemplos de uso:

- isArchivo("archivo") entrega False.
- isArchivo("<<<archivo>>>") entrega True.

3.2.2. lector

Este método recibe un string que es el nombre de un archivo y entrega otro string con los contenidos de este archivo, expandiendo archivos a los que se haga referencia. Como se mencionó anteriormente, en la clase Expand se crea una lista con los archivos leídos. Dado esto, lo primero que hace lector() es incluir el archivo recibido a la lista. Luego se crea un string "resultado" donde se guardará lo que se debe imprimir.

Utilizando la clase FileReader y BufferedReader se lee cada línea del archivo mediante un *while*. A su vez, cada línea se separa en cada una de sus palabras y se itera sobre ellas con un *for*. Primero se utiliza la función **esArchivo()** para verificar si la palabra corresponde a un archivo. En caso de serlo, se revisa si se encuentra en la lista de leídos. Si lo está, el programa se detiene y se indica que hay un "loop", en caso contrario se lee de manera recursiva este nuevo archivo utilizando **lector()** y se agrega al string "resultado". Si la palabra no es un archivo, esta simplemente se agrega al string "resultado".

Esta revisión de casos se representa en la siguiente parte del código:

```
for (String palabra : linea.split(" ")) { //se separa la linea en palabras y se itera
    sobre ellas
            if (isArchivo(palabra)){ //se verifica si la palabra es un archivo
                String nuevoArchivo = palabra.substring(3,palabra.length()-3);
                if (leidos.contains(nuevoArchivo)){ //si lo es y ya se ha leido, hay
                    un loop y el programa se detiene
                    System.out.print("Error: Loop de referencias de archivos");
                    System.exit(0);
                }
                else{ //si es un archivo, y no se ha leido, se lee recursivamente y se
                    agrega al resultado
                    resultado += lector(palabra.substring(3,palabra.length()-3));
                }
            }
            else { //si no es un archivo, se agrega al resultado
                resultado += palabra + " ";
            }

        }
```

Finalmente se quita el nombre del archivo inicial de la lista de leídos para que este pueda ser leído nuevamente en un caso en que, por ejemplo, se haga referencia a un mismo archivo dos veces sin producirse un "loop".

Esta función entrega el string "resultado". El código completo de este método se encuentra en el anexo 6.2

Ejemplos de uso:

1. **El archivo1 dice:**
   Este es el primer archivo. \n
   En la siguiente línea se encuentra el archivo2: \n
   <<<archivo2>>> \n
   Fin del archivo1.

2. **El archivo2 dice:**
   "Este es el segundo archivo".

3. **El archivo3 dice:**
   El archivo3 contiene al archivo4 aquí: <<<archivo4>>>

4. **El archivo4 dice:**
   El archivo4 contiene al archivo3 aquí: <<<archivo3>>>

5. **El archivo5 dice:**
   Aquí va el archivo1: <<<archivo1>>> \n
   y aquí va el archivo 2: <<<archivo2>>>.

- lector(archivo1) entrega:
  "Este es el primer archivo. \n
  En la siguiente línea se encuentra el archivo2: \n
  "Este es el segundo archivo". \n
  \n
  Fin del archivo1. \n"

- lector(archivo3) entrega:
  "Error: Loop de referencias de archivos"

- lector(archivo5) entrega:
  "Aquí va el archivo1: Este es el primer archivo. \n
  En la siguiente línea se encuentra el archivo2: \n
  "Este es el segundo archivo". \n
  \n
  Fin del archivo1. \n
  \n
  y aquí va el archivo2: "Este es el segundo archivo". \n"

3.2.3. lectorVarios

Este método recibe un arreglo de strings (donde cada string corresponde a un archivo) e imprime el "resultado" de cada archivo utilizando **lector()** en cada uno de ellos. El código de este método se encuentra en el anexo 6.3

Ejemplos de uso

- lectorVarios({"archivo1", "archivo2"}) imprime:
  Este es el primer archivo. \n
  En la siguiente línea se encuentra el archivo2: \n
  "Este es el segundo archivo". \n
  \n
  Fin del archivo1. \n
  "Este es el segundo archivo". \n

- lectorVarios({"archivo3", "archivo5"}) imprime:
  "Error: Loop de referencias de archivos"

- lectorVarios({"archivo5", "archivo3"}) imprime:
  Aquí va el archivo1: Este es el primer archivo. \n
  En la siguiente línea se encuentra el archivo2: \n
  "Este es el segundo archivo". \n
  \n
  Fin del archivo1. \n
  \n
  y aquí va el archivo2: "Este es el segundo archivo" \n
  \n
  "Error: Loop de referencias de archivos".

3.2.4. main

Este método permite que la clase y métodos creados puedan ser implementados desde la línea de comando. En primer lugar se inicializa la lista de strings que contiene a los archivos leídos, y luego se entregan los argumentos a lectorVarios(). El código de este método se encuentra en el anexo 6.4.

Ejemplo de uso:

- *java Expand archivo1 archivo2* imprime lo mismo que lectorVarios({“archivo1”, “archivo2”})

4. Modo de uso

Para utilizar este programa, se deben guardar los archivos que se desean expandir junto a **Expand.java** en una misma carpeta. Al principio del código se debe indicar el nombre de esta carpeta, que actualmente es "src". A continuación, se debe compilar la clase **Expand.java**. Luego, para ejecutar el programa, en la linea de comando se deben ingresar las siguientes instrucciones:

```
1 java Expand archivo1 archivo2 ... archivoN
```

donde se indican los nombres de los archivos que se desean expandir en los lugares de los archivo1 hasta archivoN. Se puede ingresar una cantidad finita de archivos tan grande como se desee.

5. Resultados y análisis

Finalmente se creó la clase **Expand.java** que cumple los objetivos planteados. Es posible expandir un número finito de archivos, incluyendo las referencias a otros. Además, el programa detecta "loops" de referencias y se detiene, entregando el siguiente mensaje: *"Error: Loop de referencias de archivos"*

El uso de la recursión fue fundamental para llevar a cabo este programa y fue finalmente lo que permitió la solución del problema. A pesar de que este método puede ser ineficiente para ciertos casos, resultó ser la mejor opción para lograr expandir un archivo, debido a las referencias existentes.

Para un futuro programa como este, podría mejorarse la función **lectorVarios()**. A pesar de que **lector()** logra que no se imprima nada antes de un error, **lectorVarios()** no. Como se observa en el último ejemplo de uso de este método, en caso de entregar primero un archivo sin errores y luego uno con, se expande el primer archivo y luego se indica el error en el segundo.

A pesar de esto, se logró resolver el problema de manera satisfactoria.

6. Anexos

6.1. isArchivo

```
public static boolean isArchivo(String palabra){ //verifica si una palabra es un archivo
      for(int i=0; i < 3; i++){
          if(palabra.charAt(i) != '<' || palabra.charAt(palabra.length()-i-1) != '>' ){
              return false;
          }
      }
      return true;
   }
```

6.2. lector

```
   public static String lector(String fileName) throws IOException { //lee un archivo
       completo

       leidos.add(fileName); //se agrega el archivo a la lista de leidos
       String resultado = ""; //se crea un string que entregará el resultado
       String linea;

       // La clase FileReader permite abrir un archivo para lectura
       FileReader readableFile = new FileReader(fileName);
       // La clase BufferedReader permite leer del FileReader anterior
       BufferedReader reader = new BufferedReader(readableFile);

       linea = reader.readLine();

       while (linea != null) { //se lee el archivo
           for (String palabra : linea.split(" ")) { //se separa la linea en palabras y
               se itera sobre ellas
               if (isArchivo(palabra)){ //se verifica si la palabra es un archivo
                   String nuevoArchivo = palabra.substring(3,palabra.length()-3);
                   if (leidos.contains(nuevoArchivo)){ //si lo es y ya se ha leido, hay
                       un loop y el programa se detiene
                       System.out.print("Error: Loop de referencias de archivos");
                       System.exit(0);
                   }
                   else{ //si es un archivo, y no se ha leido, se lee recursivamente y se
                       agrega al resultado
                       resultado += lector(palabra.substring(3,palabra.length()-3));
                   }
               }
               else { //si no es un archivo, se agrega al resultado
                   resultado += palabra + " ";
               }

           }
           resultado += "\n";
           linea = reader.readLine();
```

```
        }

        leidos.remove(fileName); //una vez que se leyo un archivo este se quita de la
            lista para poder ser leido nuevamente
        // Luego de leer es necesario cerrar el archivo
        reader.close();
        readableFile.close();

        return resultado;
    }
```

6.3. lectorVarios

```
public static void lectorVarios(String[] archivos) throws IOException { //funcion que
    lee varios archivos
        for (String archivo : archivos) { //se itera sobre los archivos
            System.out.print(lector(archivo)); //se lee cada archivo y se imprime el
                "resultado" se cada uno
        }
    }
```

6.4. main

```
public static void main(String[] args) { //para que se pueda llamar desde el terminal
        leidos = new ArrayList<String>();
        try {
            lectorVarios(args); //toma los archivos ingresados y los "entrega" a
                lectorVarios()
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
```