LoC Counting ARSW

Autor: Julian Eduardo Arias Barrera

1. INTRODUCCIÓN

Enseñare el funcionamiento, diseño general, pruebas y utilización de LoC Counting. Aplicativo que nos permite contar las líneas físicas y de código de un archivo java.

2. INSTALACIÓN Y EJECUCIÓN

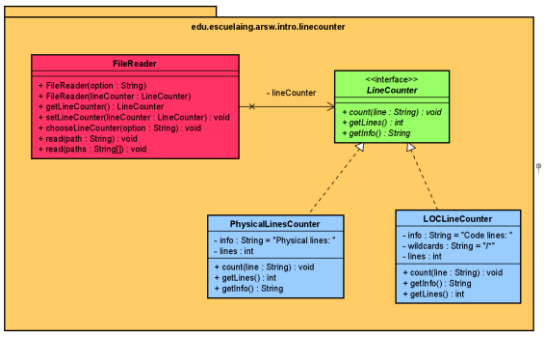
Para compilar LoC Counting se debe tener Git, Maven y Java 1.8, y seguir estos pasos

- Clonar el repositorio:
  - git clone https://github.com/AriasAEnima/LOC_Counting.git
- Ejecutar:
  - mvn package
- Y finalmente correr la acción deseada

java -cp target/LOC_Counting-1.0-SNAPSHOT.jar edu.escuelaing.arsw.intro.App <opt> <path>

Donde opt actualmente puede ser phy o loc y el path puede ser relativo a la carpeta de ejecución.

3. DISEÑO



1

Fig. 1. Modelo de LoC.

Utilice un patrón muy cercano a Strategy, con la excepción que "el contexto" toma las decisiones a través de un único (chooseLineCounter) y no varios, esto dado a que utilizamos un String para determinar cuál de las implementaciones concretas deseamos utilizar.

En términos generales este patrón nos permite reutilizar un objeto y/o configuración en particular, que en este caso seria FileReader (el contexto) y mantener/cambiar una implementación de LineCounter (estrategia) con un <set>. También nos permite agregar más implementaciones de LineCounter (estrategias) y realizar cosas como read([ ]) , que en pocas líneas se logra leer y cambiar el LineCounter de varios archivos ejemplo:

```
String[] patharchivos= [archivo1,archivo2, .. archivon];
FileReader fr=new FileReader(new LoCLineCounter());

fr.read(patharchivos);
fr.setLineCounter(new PhysicalLineCounter());
fr.read(patharchivos);
..
fr.setLineCounter(new Otraimplementacion());
fr.read(patharchivos);
..
Etc ..
```

Referencias de "contexto" y "estrategias" extraídas de Wikipedia. (Ver Bibliografía)

4. PRUEBAS

Se realizaron 5 pruebas, 4 probando cada uno de las dos configuraciones posibles y los dos archivos de prueba y uno que utilizamos ambos.

- Para el archivo prueba.java y opción phy

```
public void testPrimerArchivoPhy()
{
    try {
        FileReader FR = new FileReader("phy");
        FR.read("resources/prueba.java");
        int resultado=FR.getLineCounter().getLines();
        assertEquals(resultado,32);
    } catch (Exception ex) {
        ex.printStackTrace();
        fail("Algo Fallo ! ");
    }
```

- Para el archivo prueba.java y opción loc

```
public void testPrimerArchivoLoc()
{
    try {
        FileReader FR = new FileReader("loc");
        FR.read("resources/prueba.java");
        int resultado=FR.getLineCounter().getLines();
        assertEquals(12,resultado);
    } catch (Exception ex) {
        ex.printStackTrace();
        fail("Algo Fallo ! ");
    }

}
```

- De forma análoga con el archivo prueba2.java.

```
public void testSegundoArchivoPhy()
{
    try {
        FileReader FR = new FileReader("phy");
        FR.read("resources/prueba2.java");
        int resultado=FR.getLineCounter().getLines();
        assertEquals(96,resultado);
    } catch (Exception ex) {
        ex.printStackTrace();
        fail("Algo Fallo ! ");
    }

}

 public void testSegundoArchivoLoc()
{
    try {
        FileReader FR = new FileReader("loc");
        FR.read("resources/prueba2.java");
        int resultado=FR.getLineCounter().getLines();
        assertEquals(resultado,57);
    } catch (Exception ex) {
        ex.printStackTrace();
        fail("Algo Fallo ! ");
    }

}
```

- Para un array de paths.

```
public void testAmbosArchivosLoc()
{
    try {
        FileReader FR = new FileReader("loc");
        FR.read(new String[]{"resources/prueba2.java",
            "resources/prueba.java"});
        int resultado=FR.getLineCounter().getLines();
        assertEquals(resultado,57+12);
    } catch (Exception ex) {
        ex.printStackTrace();
        fail("Algo Fallo ! ");
    }

}
```

Los resultados de las 5 pruebas en orden aleatorio nos muestran que son correctos.

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running edu.escuelaing.arsw.intro.AppTest
Code lines: 57
Code lines: 57
Code lines: 69
Physical lines: 96
Physical lines: 32
Code lines: 12
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.034 sec

Results :

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
```

5. CONCLUSIONES

- Con la explicación del diseño y la prueba #5 se comprueba que es practica su utilización para varios archivos y diferentes implementaciones, y también para el programador que desee agregar más implementaciones.

6. BIBLIOGRAFIA

- colaboradores de Wikipedia. (2020, abril 17). Bridge (patrón de diseño) - Wikipedia, la enciclopedia libre. Recuperado 2 de junio de 2020, de https://es.wikipedia.org/wiki/Bridge_(patr%C3%B3n_de_dise%C3%B1o)