Grupo EBC1-6: https://github.com/Moniica2505/EBC1-6

Mónica Romero Nájera Víctor Miguel Mora Alcázar

Implementación de un artefacto N-Cubo de Rubick

Tratamiento del fichero JSON

A partir de un fichero JSON proporcionado, el programa debe extraer la configuración inicial del cubo, asi como su dimensión.

Lo primero que se hace es parsear el fichero JSON, del que se obtienen los valores iniciales de cada cara.

El cubo tiene 6 caras: BACK, DOWN, FRONT, LEFT, RIGHT, UP.

```
JSONParser parser = new JSONParser();

	try (Reader reader = new FileReader(ubicacionJSON)) {

		JSONObject jsonObject = (JSONObject) parser.parse(reader);
		JSONArray back = (JSONArray) jsonObject.get("BACK");
```

Como el objetivo es transformar la información de el fichero en una String, creamos una función que “limpia” la información obtenida de caracteres no deseados. Después de “limpiar” cada parte obtenida, se juntan usando un sumatorio.

```
for(int i=0; i<back.size(); i++){
	sucia = (back.get(i)).toString();
	stringCubo = stringCubo + Limpiar(sucia);
}

public static String Limpiar(String str) {
      String limpia = ((str.replaceAll(",","") ).replaceAll("\\[","")).
      replaceAll("\\]", "");
      return limpia;
}
```

Para generar un ID del cubo, se pide que sea mediante hash MD5, para lo cual tenemos una función llamada MD5, que pasándole el String del cubo como argumento, devuelve el ID.

```
    public static String MD5(String input)
{
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] messageDigest = md.digest(input.getBytes());
        BigInteger no = new BigInteger(1, messageDigest);
        String hashtext = no.toString(16);
        while (hashtext.length() < 32) {
            hashtext = "0" + hashtext;
        }
        return hashtext;
    }
    catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
```

Fuente: https://www.geeksforgeeks.org/md5-hash-in-java/

Representación interna del cubo

Una vez hemos convertido el fichero JSON en un String, lo convertimos en vector, para poder recorrer sus valores. Cada valor representa el color que tiene cada "mini cubito" en el cubo. Además, ya podemos saber que dimensión tendrá el cubo.

```
dimension = (stringCubo.length())/18;
String array[] = stringCubo.split("");
```

Con este vector, podemos asignar los valores a una matriz tridimensional, que será en definitiva lo que representa el cubo, siendo 'a' las caras del cubo, 'b' la fila de esa cara y 'c' la columna en la fila. Para recorrer el array que contiene los valores, usamos un triple bucle *for* y una variable usada a modo de contador

```
cubo = new String[6][dimension][dimension];

for(a=0; a<6; a++) {
      for(b=0; b<dimension; b++) {
            for(c=0; c<dimension; c++) {

                  cubo[a][c][b]=array[contador];
                  contador = contador + 1;
                  System.out.print(cubo[a][c][b]);

            }
      }
}
```

Movimientos

En un principio, implementamos los movimientos pero de forma errónea, ya que presentaba problemas en caso de tener un cubo que no fuese de 3x3. Estamos en proceso de corregir e implementarlos para un cubo NxNxN