P. Thiemann, L. Fennell Sommersemester 2017

Programmieren in Java

`http://proglang.informatik.uni-freiburg.de/teaching/java/2017/`

filesystem

Einträge eines Dateisystems

Woche 12 Aufgabe 3/3

Herausgabe: 2017-07-20
Abgabe: 2017-08-08

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.

Project	`filesystem`
Package	`filesystem`
Klassen	

Main
`public static void main(String[])`

In dieser Aufgabe wird eine Baumstruktur wie sie in Dateisystemen vorkommt modelliert. Ein Dateisystemobjekt ist entweder ein *Verzeichnis* (eng. directory) oder eine *Datei* (eng. file). Eine Datei besteht aus ihrem Inhalt, einem String. Die Größe einer Datei ist die Länge ihres Inhalts. Ein Verzeichnis bildet Dateinamen (Strings) auf Dateisystemobjekte ab. Die Größe eines Verzeichnisses ist die Summe der Größe der Dateien, die es enthält.

Das Interface `FSTree` repräsentiert ein Dateisystemobjekt:

```
1  package filesystem;
2
3
4  import java.util.List;
5  import java.util.regex.Pattern;
6
7  /**
8   * Interface for an immutable filesystem tree.
9   */
10 public interface FSTree {
11
12     /**
13      * Return the total size of the tree, i.e. the sum of the sizes of all its
14      * files.
15      */
16     int getSize();
17
18     /**
```

```
19      * Return a list of TextFiles (i.e. name and content) contained in this
20      * FSTree that match the Regexp "matchName".
21      */
22     List<Document> find(Pattern matchName);
23
24     /**
25      * Return a new FSTree from this FSTree with the name and content of "file"
26      * inserted at "path". This FSTree is not modified. Missing "path" elements
27      * are created as Directories. If the "path" cannot be created, an
28      * IllegalArgumentException is thrown.
29      *
30      * @param path The path as a list of Strings where the file should be
31      * created.
32      * @param document The name and content of the file to be created, as a
33      * Document.
34      * @return The new FSTree.
35      */
36     FSTree createFile(List<String> path, Document document);
37
38 }
```

Ihre Aufgabe ist es, die Baumstruktur von `FSTree` durch rekursive Klassen zu implementieren. Die Methode `getSize` gibt dabei die Größe des Dateisystemobjekts zurück. Die Methode `find` gibt die Liste von Dateien zurück, die unter einem Dateinamen abgespeichert sind, welcher von einem gegebenen regulären Ausdruck erkannt wird. Die Dateien werden von `find` als `Document` Objekte zurückgegeben, die Dateinamen und Dateiinhalt zusammenfassen. Die Klasse `Document` finden Sie im Skelett.

Die Methode `createFile` erstellt einen neuen `FSTree`, der den ursprünglichen `FSTree` um eine Datei erweitert, die als `Document` übergeben wird. Die Datei soll unter dem Pfad `path` gespeichert sein, einer Liste an Verzeichnissen, durch die man zu der neuen Datei gelangt. Existiert die Datei bereits, wird sie von der neuen Datei ersetzt. Existieren einige dieser Verzeichnisse nicht, sollen sie erzeugt werden. Kann ein Verzeichnis nicht erzeugt werden, da es schon eine entsprechend benannte Datei gibt, soll einen `IllegalArgumentException` geworfen werden.

Zusätzlich zu den Klassen für `FSTree` sollen Sie noch die Fabrikmethode `FSTrees.empty()` implementieren, die ein leeres Verzeichnis als `FSTree` zurückgibt.

Achtung: Das Verwenden von `null` ist in dieser Aufgabe strikt verboten; Selbst wenn die Tests auf Jenkins durchlaufen, können Sie keine Punkte erlangen, wenn Sie `null` in Ihrem Code verwenden.

Beispieltests

```
1 package filesystem;
2
3 import org.junit.Test;
4
5 import java.util.Arrays;
```

```
6  import java.util.Collection;
7  import java.util.Collections;
8  import java.util.HashSet;
9  import java.util.regex.Pattern;
10
11 import static org.junit.Assert.*;
12
13 public class ExampleTests {
14
15     @Test
16     public void test1() {
17         FSTree tree = FSTrees.empty()
18                             .createFile(Arrays.asList("home", "fennell", "Desktop"),
19                                         new Document("Greeting", "Hello World"))
20                             .createFile(Arrays.asList("home", "fennell"),
21                                         new Document("Greeting2", "Hello again, World"))
22                             .createFile(Arrays.asList("home", "fennell", "Desktop"),
23                                         new Document("PhoneBook", "Lu: 2038053"));
24
25         assertEquals(40, tree.getSize());
26         assertSetEquals(Arrays.asList(
27                 new Document("Greeting", "Hello World"),
28                 new Document("Greeting2", "Hello again, World")),
29                 tree.find(Pattern.compile("Greeting[0-9]?")));
30     }
31
32     @Test(expected = IllegalArgumentException.class)
33     public void testFail() {
34         FSTree tree = FSTrees.empty()
35                                 .createFile(Arrays.asList("home", "fennell", "Desktop"),
36                                             new Document("Greeting", "Hello World"))
37                                 .createFile(Arrays.asList("home", "fennell"),
38                                             new Document("Greeting2", "Hello again, World"))
39                                 .createFile(Arrays.asList("home", "fennell", "Desktop"),
40                                             new Document("PhoneBook", "Lu: 2038053"));
41
42         tree.createFile(Arrays.asList("home", "fennell", "Desktop", "Greeting", "subdir"),
43                         new Document("File", "content"));
44     }
45
46     private <T> void assertSetEquals(Collection<T> expected, Collection<T> actual) {
47         assertEquals(new HashSet<>(expected), new HashSet<>(actual));
48     }
49
50
51 }
```