Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления
Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №5
«Монады в языке Java»
по курсу: «Языки и методы программирования»

Выполнил:
Студент группы ИУ9-21Б
Старовойтов А. И.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

Приобретение навыков использования монад Optional и Stream в программах на языке Java.

Задачи

Во время выполнения лабораторной работы требуется разработать на языке Java один из классов, перечисленных в таблице, которая приведена ниже. В каждом классе нужно реализовать по крайней мере два метода: первый метод должен возвращать Stream, а второй – Optional. Операции, выполняемые каждым методом, указаны в вариантах задания. В методе main вспомогательного класса Test нужно продемонстрировать работоспособность разработанного класса, осуществив группировку содержимого потока, возвращаемого первым методом, с помощью группирующего коллектора. В исходном коде (включая класс Test) запрещено использовать циклы и рекурсию.

Решение

FileTree.java

```
import java.util.*;
import java.util.stream.Stream;

public class FileTree {
    private Directory root = new Directory("", 0);
    private int maxDepthContainsTwoFiles = 0;
    private int maxDepth = 0;

    private class Directory {
        private Set<String> files;
        private HashMap<String, Directory> next;
        private String name;
        private int depth;

        public Directory(String name, int depth) {
            this.name = name;
```

```
            this.files = new HashSet<String>();
            this.next = new HashMap<String, Directory>();
            this.depth = depth;
            maxDepth = java.lang.Math.max(depth, maxDepth);
        }

        public Directory go(String name) {
            if (!next.containsKey(name)) {
                Directory newDir = new Directory(name,
↪   this.depth+1);
                next.put(name, newDir);
                return newDir;
            }
            return next.get(name);
        }

        public void addFile(String name) {
            files.add(name);
            if (files.size() > 1 && this.depth >
             ↪  maxDepthContainsTwoFiles) {
                maxDepthContainsTwoFiles = this.depth;
            }
        }
    }

    public void add(String path) {
        int nameIndex = path.lastIndexOf("/");
        String name = path.substring(nameIndex+1);
        path = path.substring(0, nameIndex);
        Directory dir = findDir(path);
        dir.addFile(name);
    }

    public Directory findDir(String path) {
        Directory dir = root;
        for (String x : path.substring(1).split("/")) {
            dir = dir.go(x);
        }
        return dir;
    }
```

```
    public Stream<String> nameStream(String path) {
        Directory dir = findDir(path);
        return dir.files.stream();
    }

    public Optional<Integer> minDepthContainsOneFile() {
        if (maxDepthContainsTwoFiles == maxDepth) {
            return Optional.empty();
        }
        return Optional.of(maxDepthContainsTwoFiles+1);
    }
}
```

Test.java

```
import java.util.*;
import java.util.stream.Collectors;

public class Test {
    public static void main(String[] args) {
        FileTree test = new FileTree();
        test.add("/a/b/c/c.test");
        test.add("/a/b/c.aoao");
        test.add("/a/c.aoao");
        test.add("/a/d.aoao");
        test.add("/a/test.test.test12");
        test.add("/a/test");


 ↪      test.nameStream("/a/b/c").forEach(System.out::println);
        Optional<Integer> n =
 ↪      test.minDepthContainsOneFile();
        if (n.isPresent()) {
            System.out.println(n.get());
        } else {
            System.out.println("none");
        }
        Map<String, List<String>> files =
         ↪      test.nameStream("/a/")
```

```
            .collect(Collectors.groupingBy(x ->
             ↪  x.contains(".") ?
             ↪  (x.substring(x.lastIndexOf(".")+1)) : "No
             ↪  extension"));
        System.out.println(files);
    }
}
```

Пример вывода

```
c.test
2
{aoao=[c.aoao, d.aoao], test12=[test.test.test12], No extension=[test]}
```