Лекция 8.4: Новое в Java / Функциональные алгоритмы.

1. Избавляемся от циклов
2. Комбинаторные алгоритмы
3. Работаем с бесконечными структурами
Тесты по всей лекции
Видео
Лабораторные
... java8.functional.list_deep_copy
Литература

Ваша успеваемость

Java Core	14.34%
1. Основы Java	88.07%
2. Базовые алгоритмы	24.13%
3. Исключения	31.21%
4. Ввод/вывод	0%
5. Многопоточность	0%
6. Коллекции	0%
7. ООП: Синтаксис	0%
8. ООП: Шаблоны	0%
9. Продвинутые возможности	0%
10. Java 8	0%

1. Избавляемся от циклов

РАЗДЕЛ В РАЗРАБОТКЕ

2. Комбинаторные алгоритмы

РАЗДЕЛ В РАЗРАБОТКЕ

3. Работаем с бесконечными структурами

РАЗДЕЛ В РАЗРАБОТКЕ

Тесты по всей лекции

Тест, состоящий из случайных вопросов тестов этой лекции

РАЗДЕЛ В РАЗРАБОТКЕ

Видео

Набор октябрь 2013 (вторая половина часть 5 + часть 6) (functional pearls)

Лабораторные

java8.functional.list_deep_copy

java8.functional.list_deep_copy

Если у вас есть некоторый
List origin = ...
вы всегда можете сделать копию вызовом
List copy = new ArrayList<>(origin);

И, на первый взгляд, все хорошо

```
import java.util.ArrayList;
import java.util.List;
import static java.util.Arrays.asList;

public class App {
    public static void main(String[] args) {
        List> original = new ArrayList<>(asList(
                new ArrayList<>(asList(1, 2)),
                new ArrayList<>(asList(3, 4))
        ));
        List> copy = new ArrayList<>(original);

        System.out.println("original: " + original);
        System.out.println("copy: " + copy);

        original.add(asList(100500));

        System.out.println("original: " + original);
        System.out.println("copy: " + copy);
    }
}

>> original: [[1, 2], [3, 4]]
>> copy: [[1, 2], [3, 4]]
>> original: [[1, 2], [3, 4], [100500]]
>> copy: [[1, 2], [3, 4]]
```

Однако, вы получите ПОВЕРХНОСТНУЮ (shallow copy), то есть хотя оригинал и копия - это различные List-ы, но будут содержать ссылки на одни и те же элементы.

```
import java.util.ArrayList;
import java.util.List;
import static java.util.Arrays.asList;

public class App {
    public static void main(String[] args) {
        List> original = new ArrayList<>(asList(
                new ArrayList<>(asList(1, 2)),
                new ArrayList<>(asList(3, 4))
        ));
        List> copy = new ArrayList<>(original);

        System.out.println("original: " + original);
        System.out.println("copy: " + copy);

        original.get(0).add(100500);

        System.out.println("original: " + original);
        System.out.println("copy: " + copy);
    }
}

>> original: [[1, 2], [3, 4]]
>> copy: [[1, 2], [3, 4]]
>> original: [[1, 2, 100500], [3, 4]]
>> copy: [[1, 2, 100500], [3, 4]]
```

Однако, вы можете сделать глубокое копирование (deep copy) одной строкой

```
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import static java.util.Arrays.asList;

public class App {
    public static void main(String[] args) {
        List> original = new ArrayList<>(asList(
                new ArrayList<>(asList(1, 2)),
                new ArrayList<>(asList(3, 4))
        ));
        List> copy = original.stream().map(list -> new ArrayList<>(list)).collect(Co
llectors.toList());

        System.out.println("original: " + original);
        System.out.println("copy: " + copy);

        original.get(0).add(100500);

        System.out.println("original: " + original);
        System.out.println("copy: " + copy);
    }
}

>> original: [[1, 2], [3, 4]]
>> copy: [[1, 2], [3, 4]]
>> original: [[1, 2, 100500], [3, 4]]
>> copy: [[1, 2], [3, 4]]
```

Задание: реализовать подобным образом (через Stream API) однострочную функцию глубокого копирования но не для List>, а для List>>.

```
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class ListUtils {
    public static <T> List<List<List<T>>> deepCopy(List<List<List<T>>> list) {


    }
}
```

Литература

РАЗДЕЛ В РАЗРАБОТКЕ
Loopless Functional Algorithms