# **Fuzzing**

Подключили библиотеку JQF для проведения случайного тестирования. Подключение проводилось с помощью maven, где мы указали необходимые dependencies и plugins. Конечный maven file можно увидеть в проекте (pom.xml)

Для запуска fuzzing'a мы используем команду mvn clean verify, также помимо фаззинга у нас собирается информация с помощью јасосо о покрытии кода тестами.

Результаты покрытия мы можем наблюдать ниже:

## Lesson 5

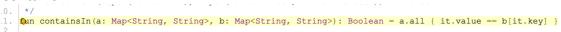
- <del></del>			
canBuildFrom(List, String)		100%	100%
<ul><li>containsIn(Map, Map)</li></ul>		100%	50%
Total	6/1 of 700	۵۰/- 5۵	of 62 /10/-

В целом фаззинг справился неплохо, покрытие инструкций на 100%, но в случае containsIn мы не зашли в одну из веток :-(

## CanbuildFrom

```
| 17. | */ | fun canBuildFrom(chars: List<Char>, word: String): Boolean = | word.lowercase().toSet().all { letter -> letter in chars.map { it.lowercaseChar() } } | 20. | | /** | * Средняя (4 балла)
```

#### ContainsIn



## Lesson 6



Здесь покрытие значительно хуже, методы, которые тестировались с помощью фаззинга выделены красной рамкой. Низкое покрытие обуславливается тем, что с помощью сгенерированного датасета не получилось пройтись по всем инструкциям и ветвям.

 ${\tt DateStrToDigit}$ 

```
80. | */
 81. fun dateStrToDigit(str: String): String {
           if (!str.matches(Regex("""\d+ ([a-я])+ \d+"""))) return ""
           val list = str.split(' ')
val day = list[0].toInt()
var month = list[1]
val year = list[2].toInt()
 83.
 84.
 85.
 86.
           val months = mapOf(
    "января" to "01",
    "февраля" to "02",
 87
 88.
                "февраля" to "02",
"марта" to "03",
"апреля" to "04",
"мая" to "05",
"июня" to "06",
"июля" to "07",
"августа" to "08",
"сентября" to "09",
 90.
 91.
 92.
 94.
 95.
 96.
                "октября" to "10",
                "ноября" to "11",
"декабря" to "12"
 98.
 99.
           if (month !in months) return "" else month = months[month].toString()
           return if (isDayCorrect(day, month.toInt(), year))
String.format("%02d.%s.%d", day, month, year) else ""
102.
104. }
105.
fun isDayCorrect(day: Int, month: Int, year: Int): Boolean {
    val limits = mutableMapOf<Int, Int>()

108.    for (i in listOf(1, 3, 5, 7, 8, 10, 12)) limits += i to 31

109.    for (i in listOf(4, 6, 9, 11)) limits += i to 30

110.    limits += if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)) 2 to 29 else 2 to 28

111.    return day in 1..limits[month]!
108. • 109. • 110. • 111. • 112. }
113.
114. /**
PlusMinus
            ^{\circ} при нарушении формата входнои строки оросить исключение ^{\circ} ^{\circ}
02.
103. fun plusMinus(expression: String): Int {

    if (expression.contains(Regex("""[+-] [+-]|\d \d|^[+-]""")))

05.
                      throw IllegalArgumentException()
06.
               val stripped = expression.filter { it != ' ' }
07.
               val numbers = Regex("[-+]").split(stripped)
               val move = stripped.filter { it in "-+" }.toList()
08.
09.
               var total = numbers[0].toInt()
10.
               for (m in move.indices) {
                      when (move[m]) {
11.
12.
                             '+' -> total += numbers[m + 1].toInt()
13.
                              '-' -> total -= numbers[m + 1].toInt()
14.
                      }
15.
               }
16.
               return total
17.
18.
Lesson7
                                                                                           88%
                                                                                                                           85%
```

Здесь покрытие кода лучше, чем в lesson 6, соответственно в данном случае fuzzing смог попасть данными в нужные инструкции и разветвления

ChooseLongestChaoticWord