Автоматическое извлечение шаблонов из кода

Плагин для Intellij IDEA

Байдин Дмирий

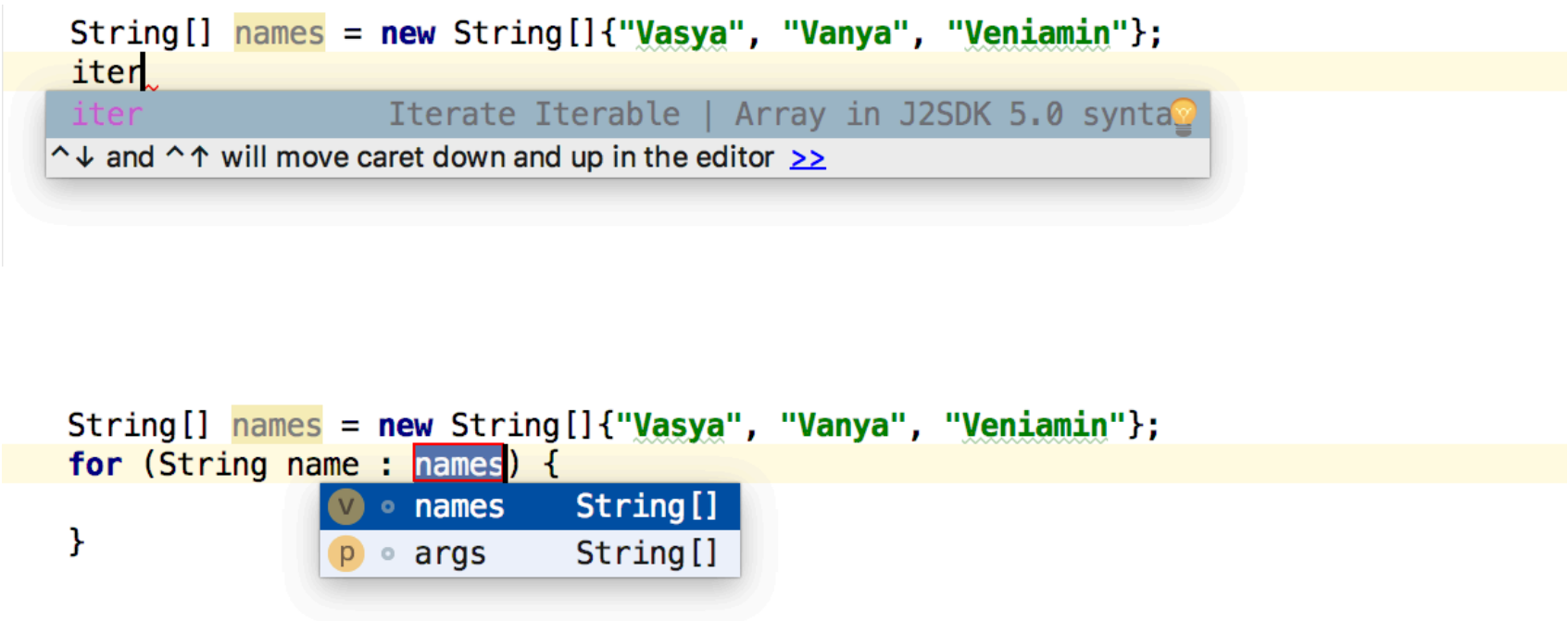
научный руководитель: Шеин Роман Евгеньевич

СПб АУ НОЦНТ РАН

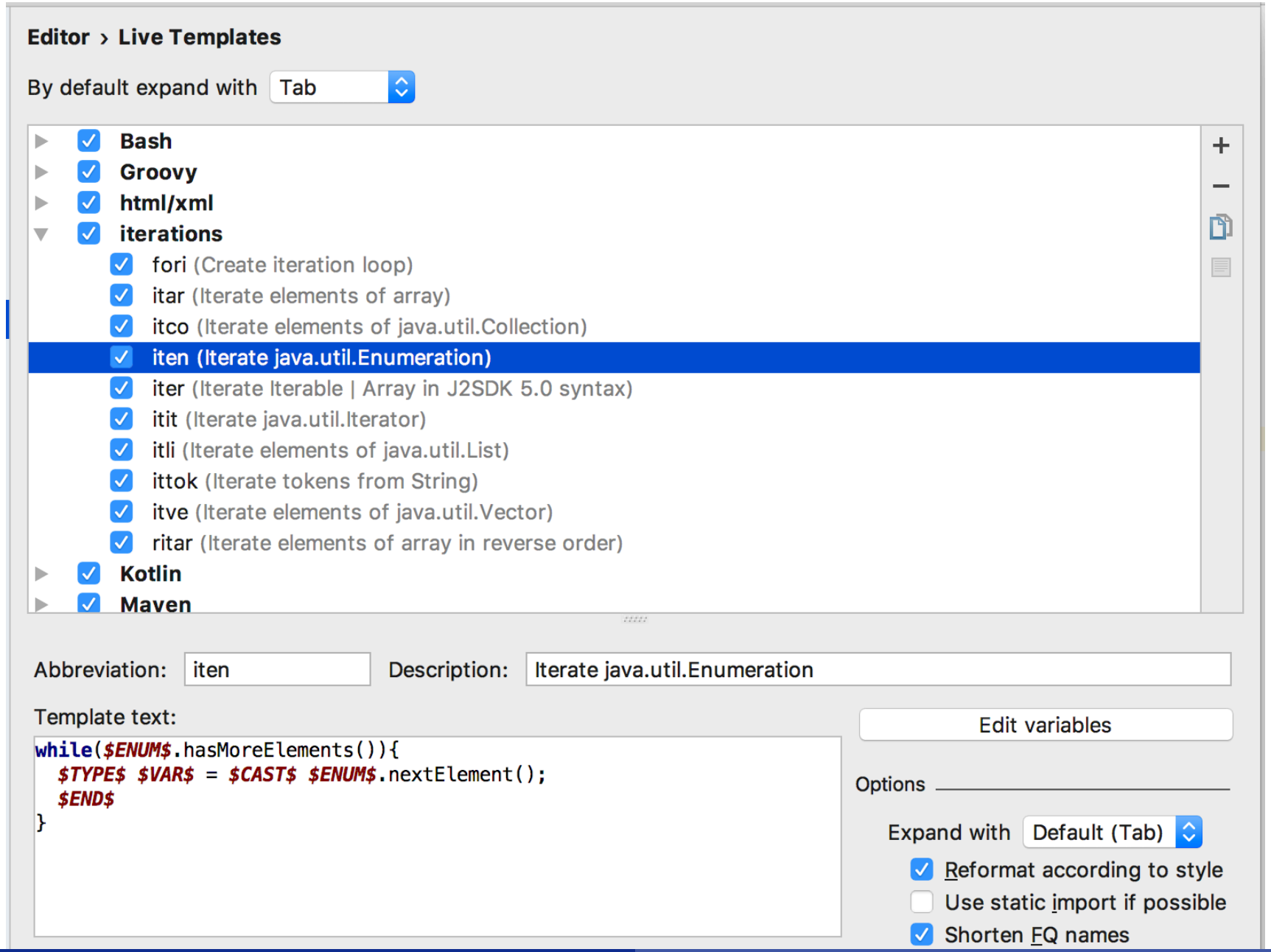
12 июня 2017 г.

Введение

Шаблон - часто используемая структура, имеющая общую часть и пропуски, которые заполняются пользователем. Пропуск может располагаться на месте имен переменных, литералов, типов, а также целых выражений.



Введение


Editor › Live Templates
By default expand with Tab
Bash
Groovy
html/xml
iterations
fori (Create iteration loop)
itar (Iterate elements of array)
itco (Iterate elements of java.util.Collection)
iten (Iterate java.util.Enumeration)
iter (Iterate Iterable | Array in J2SDK 5.0 syntax)
itit (Iterate java.util.Iterator)
itli (Iterate elements of java.util.List)
ittok (Iterate tokens from String)
itve (Iterate elements of java.util.Vector)
ritar (Iterate elements of array in reverse order)
Kotlin
Maven
Abbreviation: iten
Description: Iterate java.util.Enumeration
Template text:
while($ENUM$.hasMoreElements()){
$TYPE$ VAR = $CAST$ $ENUM$.nextElement();
END
}
Edit variables
Options
Expand with Default (Tab)
Reformat according to style
Use static import if possible
Shorten FQ names

Проблема

- Появление новых языков/библиотек для которых не существует заготовленных шаблонов.
- Необходимость проводить анализ, какие шаблоны будут подходящими.
- Текущее решение не удовлетворяет скоростью и качеством работы.

Цель и задачи

Ускорить поиск шаблонов, улучшить качество полученных шаблонов. Выложить плагин в репозиторий JetBrains.

- Изучить предыдущее решение и другие способы поиска часто встречающихся структур.
- Найти и реализовать алгоритм поиска часто встречающихся структур.
- Увеличить качество путем усовершенствования фильтрации и пост обработки шаблонов.

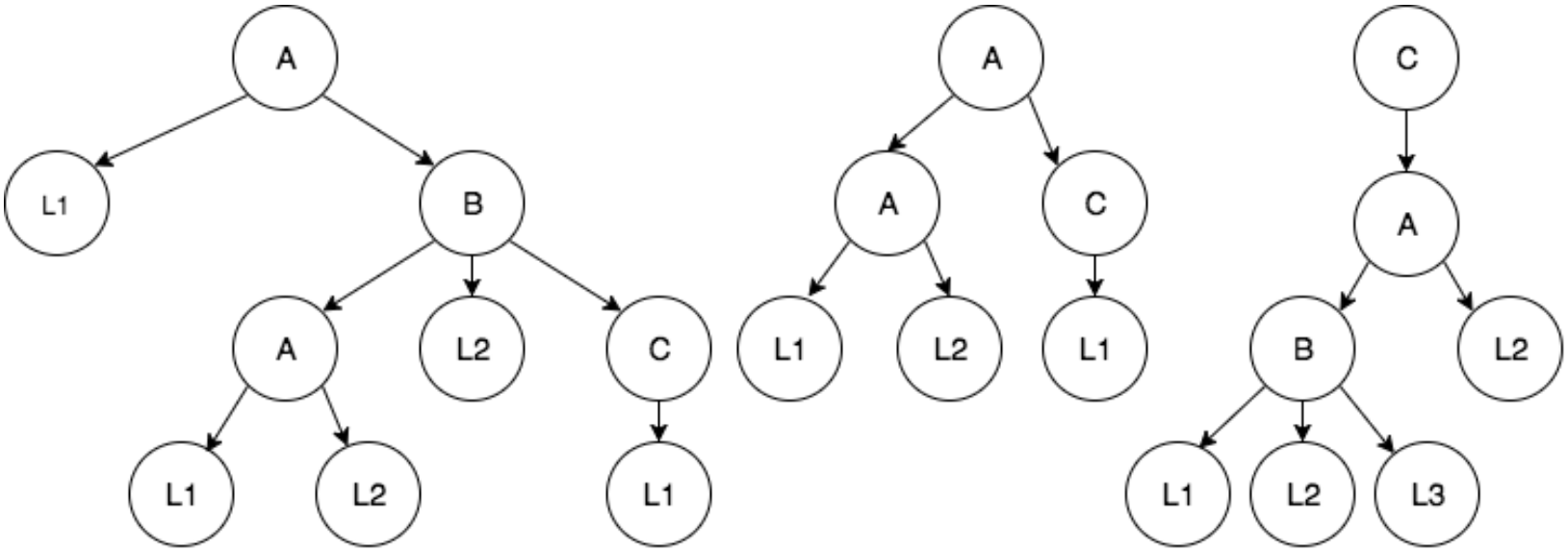
Алгоритм поиска шаблонов

- Поиск часто встречающихся структур.
- Пост-обработка.
- Выбор наиболее общих структур.
- Фильтрация.

IMB3 Поиск часто встречающихся структур

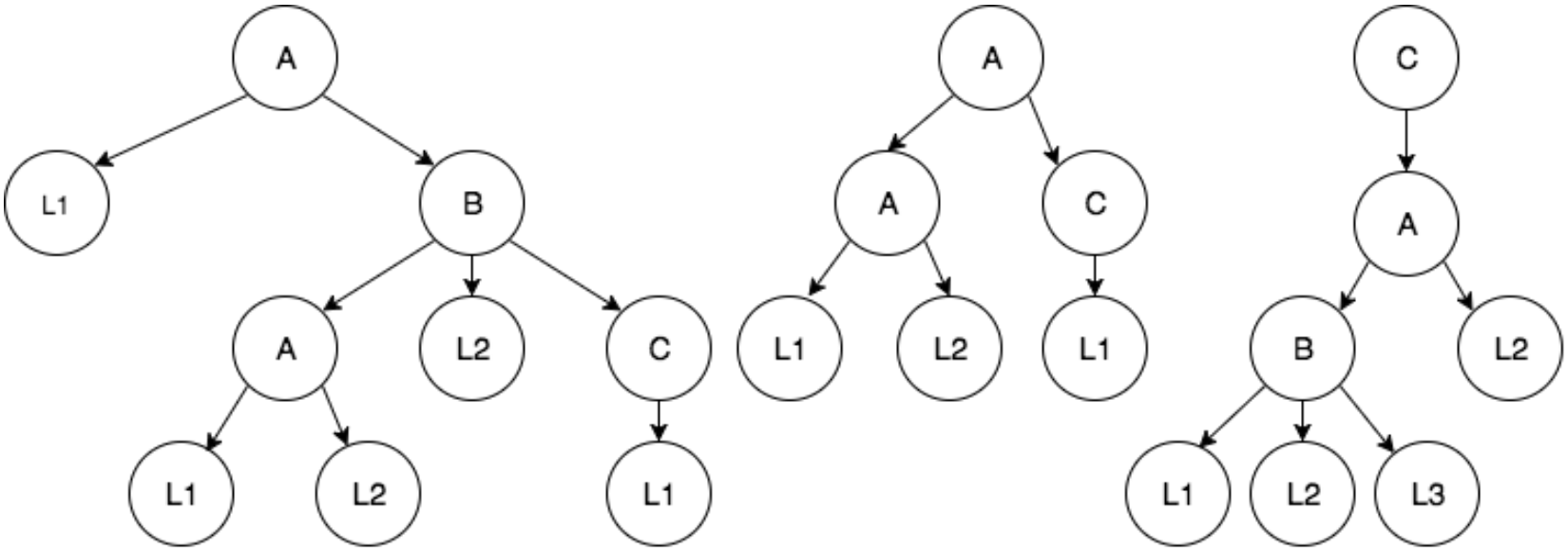
- Подсчет статистики леса, определение минимального количества раз (minSupport), которое должна встретиться структура, чтобы считаться часто встречающейся.
- Построение списка вершин. Вершины всех деревьев записанные в pre-order обходе. Вместе с каждой вершиной хранится положение самого правого ребенка (scope) и глубина.
- Расширения шаблонов. Использование вертикальных списков вхождений, в которых хранятся все вхождения шаблона в лес, и корзин.

IMB3 Список вершин



Num	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Name	A	L1	B	A	L1	L2	L2	C	L1	A	A	L1	L2	C	L1	C	A	B	L1	L2	L3	L2
Depth	0	1	1	2	3	3	2	2	3	0	1	2	2	1	2	0	1	2	3	3	3	2
Scope	8	-	8	5	-	-	-	8	-	14	12	-	-	14	-	21	21	20	-	-	-	-

IMB3 Вертикальный список вхождений

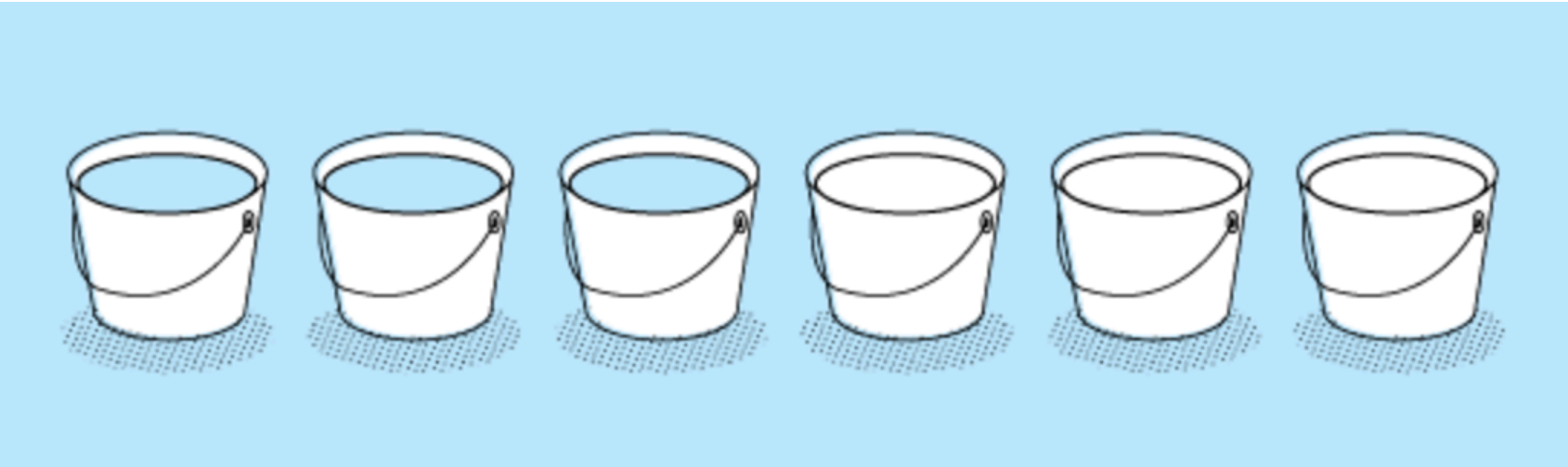


Num	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Name	A	L1	B	A	L1	L2	L2	C	L1	A	A	L1	L2	C	L1	C	A	B	L1	L2	L3	L2
Depth	0	1	1	2	3	3	2	2	3	0	1	2	2	1	2	0	1	2	3	3	3	2
Scope	8	-	8	5	-	-	-	8	-	14	12	-	-	14	-	21	21	20	-	-	-	-

A: (0, 0), (3, 3), (9, 9), (10, 10), (16, 16)
$A\ L1\ L2$: (3, 5), (10, 12)
$B\ \#\ L2\ \#$: (2, 8), (17, 21)

IMB3 Корзины

- Корзина может быть заполненной и незаполненной. Заполненная корзина содержит как минимум minSupport вхождений.
- Одно вхождение может лежать в нескольких незаполненных корзинах.
- Каждая корзина соответствует определенной вершине-кандидату на расширение.



Выбор наиболее общих структур

Используется обобщенное суффиксное дерево.

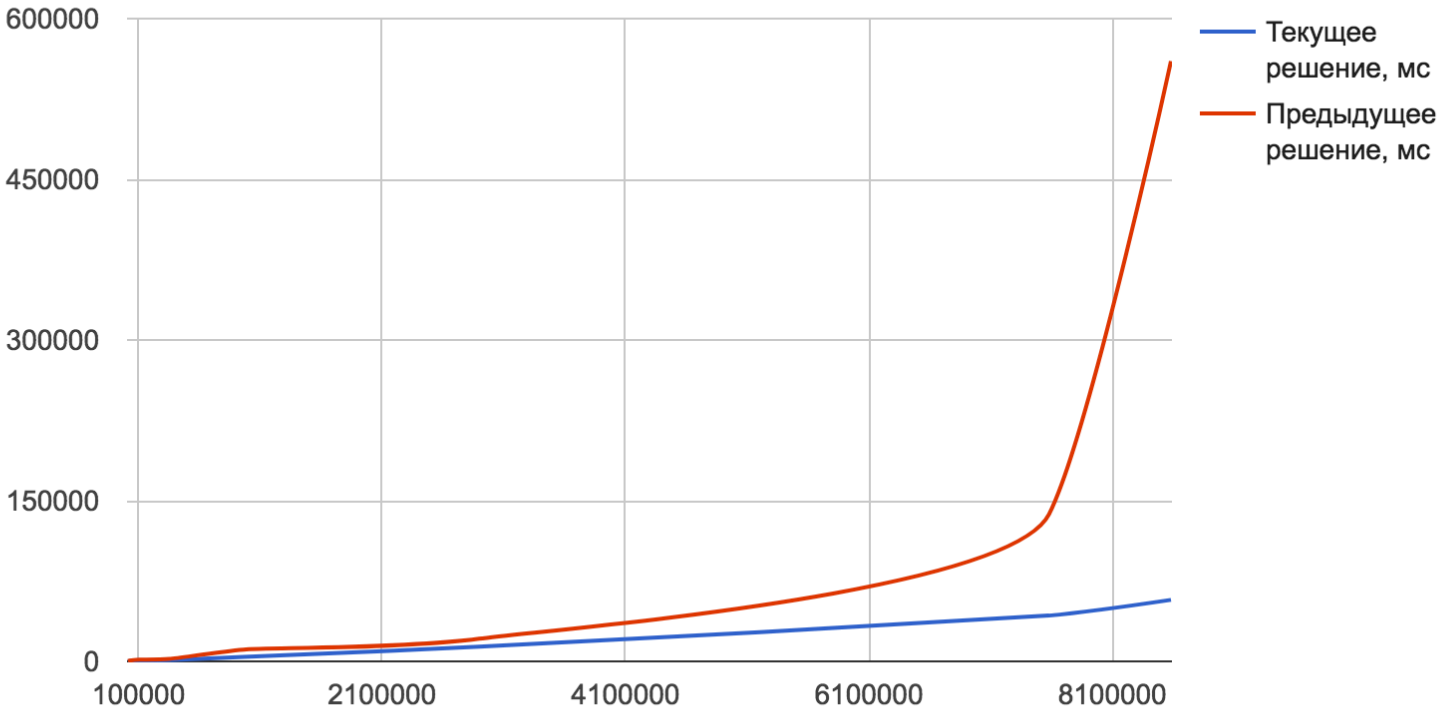
- Сортируем шаблоны по убыванию длины текста.
- Проверяем существует ли уже данный шаблон в обобщенном суффиксном дереве.
- Если его еще не существует, то добавляем его в ответ и в дерево.

Фильтрация шаблонов

Шаблоны фильтруются по следующим параметрам. В плагине существует режим, который позволяет оптимизировать значения параметров фильтрации.

- Минимальная и максимальная длина текста.
- Минимальное и максимальное количество вершин в составе шаблона.
- Максимальное количество placeholder.
- Максимальное соотношение количество placeholder к количеству вершин.

Сравнение скорости работы



Количество вершин в лесу AST деревьев

Примеры найденных шаблонов

```
1     @Test
2     @TestForIssue( jiraKey = # )
3     public void # () {
4         #
5     }
6
7     private static final Logger LOGGER = LoggerFactory.
          getLogger( # .class);
8
9     if ( #  == null) {
10        return # ;
11    }
12
13    return ( #  != null ? #  : # );
14
15    @NotNull private final #  = new # ();
```

Результаты

- Реализован алгоритм IMB3.
- Реализован выбор наиболее общих структур с помощью обобщенного суффиксного дерева.
- Оптимизация параметров при фильтрации шаблонов.
- Реализован плагин для поиска шаблонов в IDEA.

Спасибо за внимание!

`github.com/baydindima/AutomateLiveTemplates`

Сравнение скорости работы

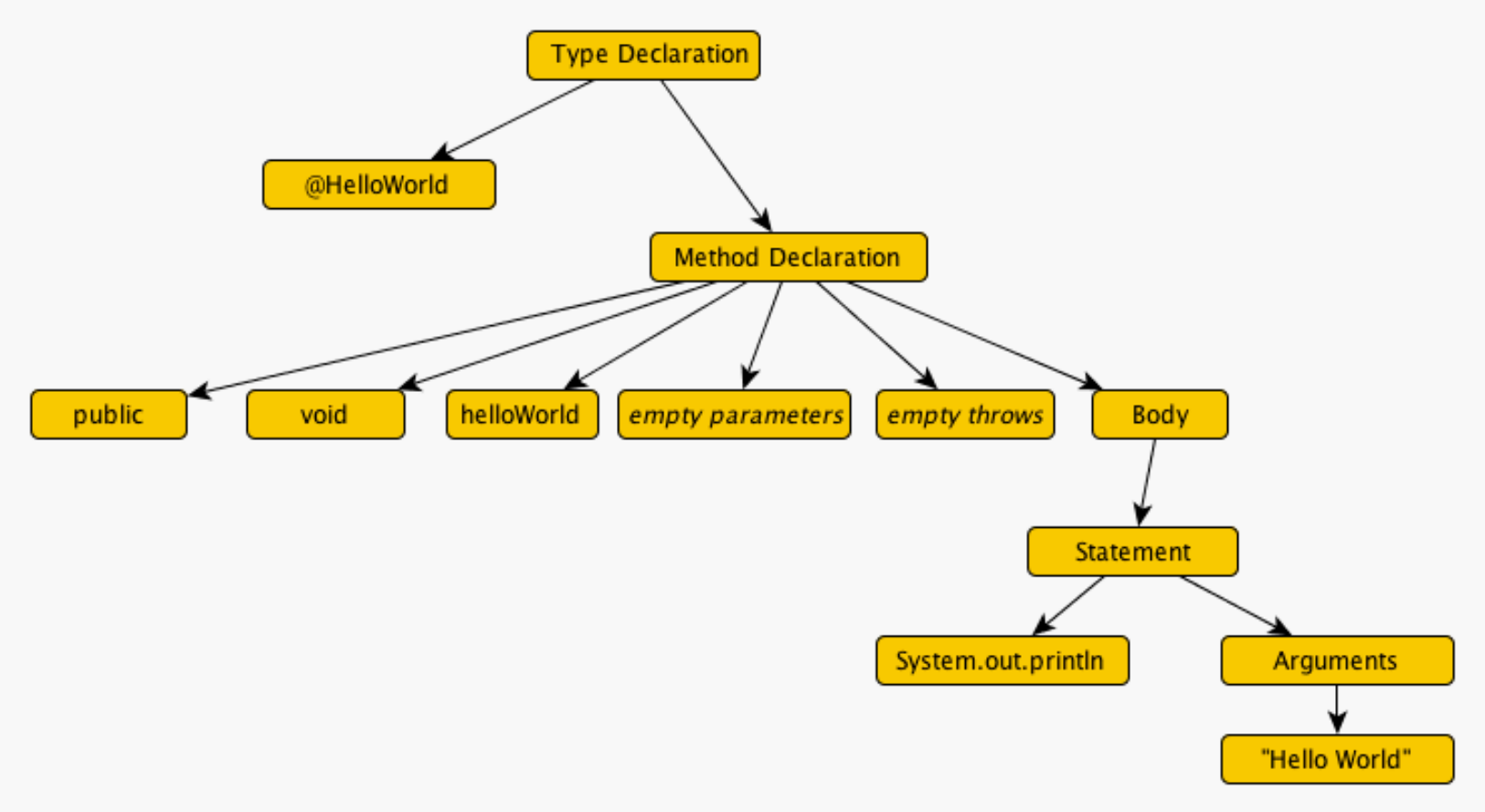
Название проекта	Количество вершин в лесу AST деревьев	Текущее решение, мс	Предыдущее решение, мс
Suffix tree	13784	43	231
Server benchmark	28885	363	681
Glide	104190	290	1589
Java Design Patterns	353308	1226	2364
OkHttp	963813	4147	11035
Spring Boot	2898363	13655	21106
Hibernate	7580079	42747	137300
Spring	8581835	57428	560286

Сложность алгоритма

- F- не листовые вершины
- $c(f)$ - количество потомков вершины
- На стадии расширения шаблонов в худшем случае мы сделаем $\mathcal{O}\left(\sum_{f \in F} c(f)^2\right)$

AST

```
1    @HelloWorld
2    public void helloWorld() {
3        System.out.println(”Hello World”);
4    }
```



Деревья похожести

- В каждой вершине храним либо количество листов с таким же текстом, либо ссылку на другую вершину с таким же типом ребенка.