Транслятор из Java в Kotlin

Дипломная работа

Кириллов Илья Олегович,
444 группа

Научный руководитель: к.т.н., доцент Литвинов Ю. В.
Консультант: Подхалюзин А. В.

24 мая 2019

Постановка задачи

Создание транслятора из Java в Kotlin для среды IntelliJ IDEA, который по возможностям и расширяемости превосходил бы старый

Задачи:

- Проанализировать недостатки старой версии транслятора
- Реализовать транслятор из языка Java в язык Kotlin
- Произвести сравнение старого и нового трансляторов

Старый транслятор

Принцип работы:

- Рекурсивно обходится представленный в виде дерева Java код и сразу же печатается код на Kotlin
- Ввиду чего отсутствует возможность многоэтапных преобразований
- И, как следствие, невозможность написания некоторых видов преобразований (например, вывод nullability)
- Генерирует большое количество кода с ошибками

Nullability в языке Kotlin

Nullability в языке Kotlin: возможность переменной принимать значение **null**

- `val x: String?` — nullable переменная
- `val y: String` — not null переменная
- `x.size` — некорректно (`x` может быть `null`)
- `y.size` — корректно (`y` точно не `null`)

Реализация

Новый транслятор:

- Написан на языке Kotlin внутри Kotlin плагина к среде IntelliJ IDEA
- Позволяет конвертировать как файлы целиком, так и фрагменты Java кода
- Многопроходный

Фазы трансляции

- Построение синтаксического дерева по Java коду
- Выполнение преобразований над ним
- Превращение синтаксического дерева в код на языке Kotlin
- Вывод nullability для типов
- Пост-обработка кода

Структура дерева

- Изменяемое дерево
- Отношение “родитель-ребёнок” поддерживается с помощью делегатов языка Kotlin

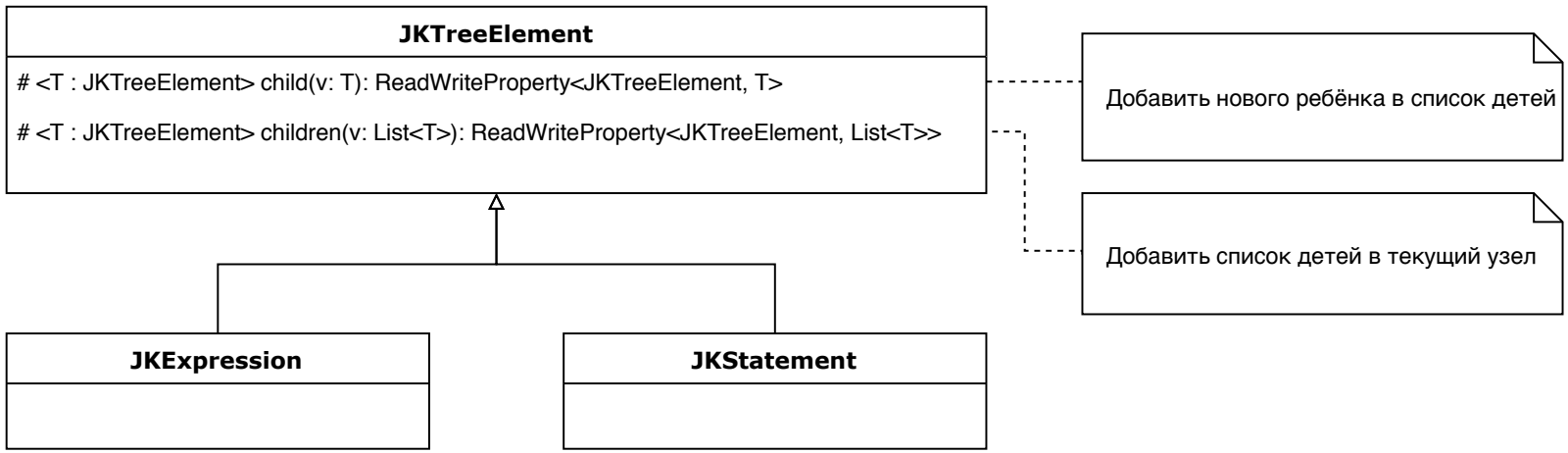

Рис.: Структура дерева

Выполнение преобразований

- Около 70 независимых преобразований
- Каждое преобразование работает со своим типом элементов
  - Рекурсивно в дереве ищется элемент, к которому преобразование может быть применено
  - Выполняет преобразование
  - Запускается на детях преобразованного узла
- Пример преобразований:
  - Конвертирующие типы
  - Объединяющие методы с аргументами по умолчанию

Вывод Nullability

- Для каждого объявления типа в программе создается типовая переменная, отражающая nullability этого типа, с тремя возможными значениями:
  - `nullable`
  - `not_null`
  - `unknown`
- Строится система для nullability из ограничений двух видов:
  - `A <: B`
  - `A := B`
- И тремя правилами (где `N` — множество всех возможных nullability):
  - `not_null <: nullable`
  - $\forall$ `S` $\in$ `N` $\Rightarrow$ `S <: S`
  - $\forall$ `S` $\in$ `N` $\Rightarrow$ `S <: unknown`

Вывод Nullability

- Каждое ограничение основывается на анализе кода, например:
  - `val a = b` $\Longrightarrow$ `nullability(b) <: nullability(a)`
  - `a.foo()` $\Longrightarrow$ `nullability(a) := not_null`
- Система последовательно решается
- Если для некой типовой переменной не найден тип, то он считается nullable

Пост-обработка

Над сгенерированным кодом на Kotlin выполняются:

- Инспекции, встроенные в IntelliJ IDEA
- Различные упрощения и оптимизации кода, например:
  - Генерация data классов
  - Объединение геттеров и сеттеров в свойства класса
  - Избавление от квалифицированных имён

Апробация

Проект	Java файлов	Строчек кода	Строчек кода на файл	Новый транслятор			Старый транслятор		
				Ошибок на файл	Всего ошибок	Время работы (сек)	Ошибок на файл	Всего ошибок	Время работы (сек)
Atlassian Jenkins v2.164.2	1557	214282	138	7.1	11057	2109	24.6	38412	977
Badlogicgames LibGDX v1.9.9	580	104841	180	8.1	4673	1373	31.6	18304	529
Google Guava v21.7	572	86944	152	4	2288	1285	35	20020	467
Apache Commons Lang v3.9	154	27889	181	7.9	1217	212	37	5695	118

Таблица: Запуск трансляторов на проектах с открытым исходным кодом

Выводы:

- Количеством ошибок компиляции меньше в 5 раз
- Время работы больше в 2.5 раз

Заключение

В рамках данной работы получены следующие результаты:

- Проанализированы недостатки старой версии транслятора
- Реализован транслятор из языка Java в язык Kotlin, превосходящий старый по качеству генерируемого кода и расширяемости
- Произведено сравнение старого транслятора с новым