ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ АКАДЕМИЧЕСКИЙ УНИВЕРСИТЕТ РОССИЙСКОЙ АКАДЕМИИ НАУК

Предотвращение Null Pointer Exception с помощью @NotNull/@Nullable для языка Groovy в IntelliJ IDEA

студент: Д. О. Овчинников
руководитель: М. Ю. Медведев

Groovy

- Мультипарадигменный
- Динамически типизированный
- Статически типизированный (опционально)
- Компилируется в байткод JVM

Null Pointer Exception

- Типы Groovy - ссылочные, может возникать NPE
- Можно проверять не хранится ли в переменной null перед ее разыменовыванием
- Отказаться от использования null, использовать шаблон програмирования Null Object
- Статически анализировать код

@NotNull/@Nullable

- С помощью аннотаций можно сообщить дополнительную информацию о типах компилятору или среде разработки

- JSR-305 https://jcp.org/en/jsr/detail?id=305

- Аннотацией @NotNull помечаются поля, переменные и параметры, которые точно не хранят null, или методы, которые точно не возвращают null

- Аналогично аннотацией @Nullable помечается то, что может хранить или возвращать null

Цель

- Необходимо научить IntelliJ IDEA не только понимать аннотации @NotNull и @Nullable в Groovy коде, но и вычислять из контекста, может ли переменная принимать значение null

Задачи

1. Подготовка реализации для Groovy
2. Построение потока управления
3. Обработка инструкций

1. Подготовка реализации для Groovy

```
if (a == null) {
    a.method()
} else {
    a.method()
}
```

- В двух разных ветвях исполнения одна и та же переменная имеет разные состояния
- В первом случае должно быть подсвечено предупреждение, а во втором нет

Анализ потока данных

- Построение графа потока управления
  - ориентированный граф
  - в вершинах содержатся инструкции
- Собственно анализ потока данных
  - начальное состояние
  - каждая инструкция как-то меняет состояние
- Анализ результатов

1. Подготовка реализации для Groovy

- Существует реализация для Java
- Выделены интерфейсы, абстракции
- В результате была переиспользована часть инструкций, а так же реализация алгоритма анализа потока данных

2. Построение графа потока управления

- Практически все в Groovy - вызов метода
- Невозможно отследить все, что меняет метод
- Перегрузка операторов
- Безопасный вызов метода: `a?.foo(b,c,d)`

2. Построение графа потока управления

- Замыкания
  - В Groovy локальные переменные могут быть изменены изнутри замыкания
- Решения
  - Рекурсивный DFA - очень долгий, и все равно может не учитывать всех изменений состояния
  - Ветвление в месте объявления замыкания

3. Обработка инструкций

- Базовые инструкции манипулирования стеком
- Инструкции перехода
- Вызов методов
- Специфичные для Groovy инструкции

3. Обработка инструкций

- Groovy Truth

```
def unknownConditions(a) {
    if (a) {
        if (a) {}
        if (!a) {}
        if (a == null) {}
        if (a != null) {}
    } else {
        if (a != null) {}
        if (a) {}
        if (!a) {}
    }
}
```

Condition 'a != null' is always true more... (⌘F1)

- Чтобы понять может ли переменная быть null, необходимо отслеживать приводится ли она к true или к false в условии.

Результаты

- Рефакторинг реализации Java
- Реализация построения потока управления
- Реализация обработчика инструкций
- Инспекция в IntelliJ IDEA

Результаты

```
 9    @NotNull
10    def trinity(SomeClassWithMethods a) {
11        a == null ?
12                a.method()
```

Method invocation 'a.method()' may produce 'java.lang.NullPointerException' more... (⌘F1)

```
65    @NotNull
66    def safeNullableFieldReference(@Nullable ClassWithField a) {
67        a?.@field
```

Expression 'a?.@field' might evaluate to null but is returned by the method declared as @NotNull more... (⌘F1)

```
70    @NotNull
71    def safeNotNullFieldReference(@NotNull ClassWithField a) {
72        a?.@field
```

Qualifier 'a' is always not null more... (⌘F1)

Спасибо за внимание!