Исследование и разработка аспектно-ориентированных расширений языка Kotlin

Скрипаль Б.А. гр. 63501/3
Руководитель: Ицыксон В.М.
Аттестация №3

Санкт-Петербургский политехнический университет Петра Великого

План работы

- ✓ Обзор существующих аспектно-ориентированных расширений для других языков;
- ✓ Формулирование требований к создаваемому расширению;
- ✓ Начальное описание грамматики аспектов для языка Kotlin;
- \+ Анализ PSI и построение срезов;
- \+ Применение советов к программе;
- \+ Доработка грамматики аспектов;
- – Реализация плагина для IntelliJ IDEA;
- – Тестирование;
- – Написание пояснительной записки.

Состояние дел

Сделано:

- Анализ промежуточного представления программы;
- Создание прототипа;
- Начато написание пояснительной записки.

Планируется сделать:

- Доработка способа внедрения советов;
- Доработка формирования срезов;
- Доработка прототипа;
- Продолжение написания пояснительной записки.

Оценка степени готовности:

- Практическая часть - 30%;
- Пояснительная записка - 10%.

Немного о том, что сделано

- Описание срезов:

```
pointcut fooPC(): execution(fun Foo.foo())
pointcut printPC(): call(fun kotlin.io.println(String))
```

- Описание советов:

```
after(): fooPC() && printPC() {
  println("Hello after!!")
}

before(): fooPC() && printPC() {
  println("Hello before!!")
}
```

- Разметка элементов PSI.

Немного о том, что сделано

При применении кода совета к точке включения, они оборачиваются в функцию «run»;

+ Легко вставлять совет, например, когда функции вызываются последовательно, например:

  ```
  val a = a.foo().bar()
  ```

  превращается в:

  ```
  val a = a.run{ val buf = foo(); advice_code; buf}.bar()
  ```

- При применении нескольких советов к одной точке включения возникают вложенные функции «run»;
- При внесении точки включения внутрь тела функции «run» необходимо обновлять метки.

Немного о том, что сделано

```
class Foo {
  fun foo() {
    run{
      val ____a = run{
        println("Hello before!!")
        println("Hello world")
      }
      println("Hello after!!")
      ____a
    }
    this.bar(1,this)
  }

  fun bar(a: Int, b: Foo) {
    println("Bar hello")
  }
}
```