С какими языками дружат IDE?

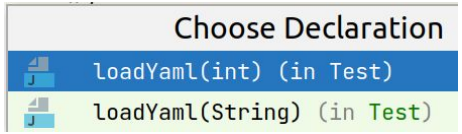
JPoint 2021

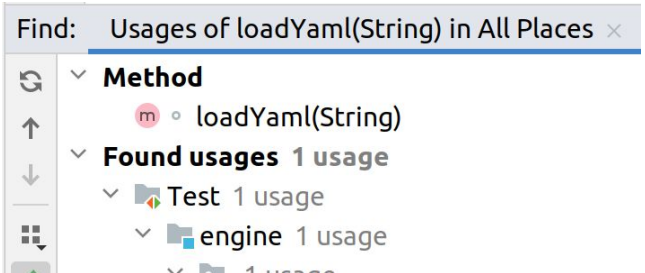
Пётр Громов
peter@jetbrains.com, @donnerpeter

Фичи IDE

Базовые:

- редактирование
- автодополнение
- форматирование
- Goto Declaration
- подсветка
- поиск/переименование





Продвинутые:

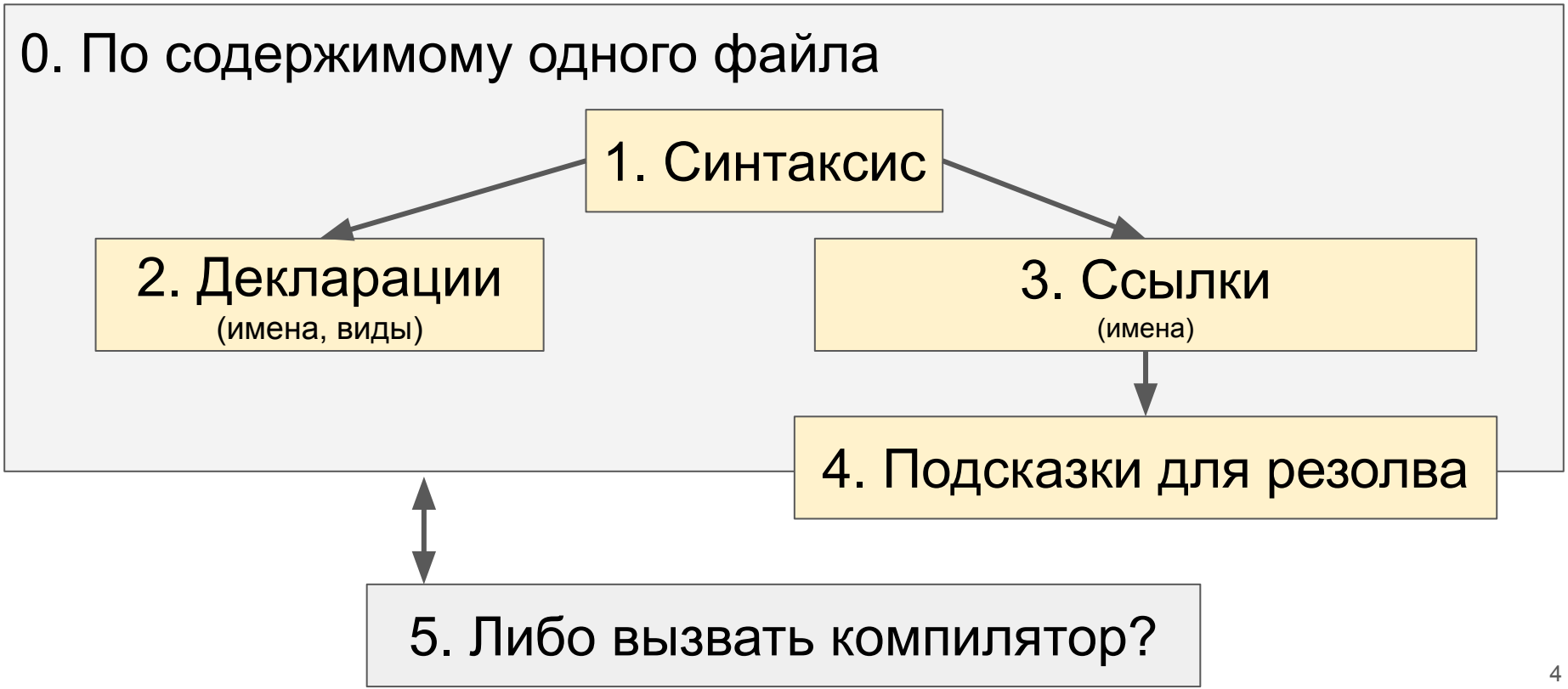
- ещё 30 рефакторингов
- сотни инспекций
  - с автофиксами
- анализ потока данных
- поиск дубликатов кода
- ...

Аудитория

- Авторы языков и фреймворков
- Все, кто выбирает, какие фичи (не) использовать
- Любопытные

Предыдущая версия:

IDE нужно определить...



IDE нужно определить:

1. Синтаксис
2. Декларации
3. Ссылки
4. Подсказки для резолва

```
package pkg

import anotherPkg.Util

data class Hello(val what: String) {
  fun hello() =
    Util.format(“Hello $what”)
}
```

IDE нужно определить:

0. По одному файлу

1. Синтаксис

2. Декларации

3. Ссылки

4. Подсказки для резолва

```
package pkg

import anotherPkg.Util

data class Hello(val what: String) {
  fun hello() =
    Util.format(“Hello $what”)
}
```

IDE нужно определить:

0. По одному файлу

1. Синтаксис

2. Декларации

3. Ссылки

4. Подсказки для резолва

5. Или скомпилировать?

```
package pkg

import anotherPkg.Util

data class Hello(val what: String) {
  fun hello() =
    Util.format(“Hello $what”)
}
```

IDE нужно определить:

0. По одному файлу

1. Синтаксис

2. Декларации

3. Ссылки

4. Подсказки для резолва

5. Или скомпилировать?

Зачем:

проиндексировать

правильно обновлять

0. По одному файлу: вопрос

Как дёшево для IDE ссылаться на другие файлы?

1. `#include “some.h”`
2. `import pkg.SomeClass`
3. неявный импорт (`java.lang.*, groovy.lang.*`)

0. По одному файлу: нарушения в Haskell

in *Main.hs*:

```
{-#LANGUAGE TemplateHaskell,TupleSections#-}
```

vs.

in *package.cabal*:

```
extensions: TemplateHaskell,TupleSections
```

0. По одному файлу: нарушения в Java

```
<properties>
   <maven.compiler.release>16</maven.compiler.release>
</properties>
```

0. По одному файлу: ответ

Как дёшево для IDE ссылаться на другие файлы?

❌ 1. `#include “some.h”`
✅ 2. `import pkg.SomeClass`
✅ 3. неявный импорт (`java.lang.*, groovy.lang.*`)

IDE нужно определить:

0. По одному файлу

1. Синтаксические конструкции

2. Декларации

3. Ссылки

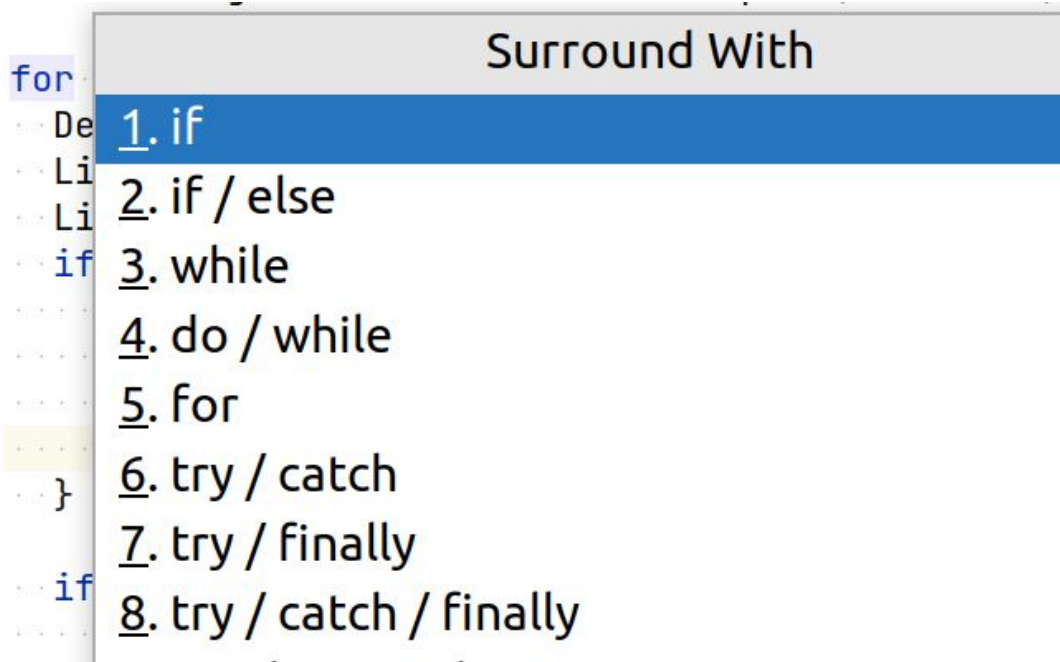
4. Подсказки для резолва

5. Или скомпилировать?

Зачем:

понимать код

помогать его писать



1. Синтаксис: вопрос про DRY-конструкции

С какими подходами синтаксис понятен?

1. Синтаксис: вопрос про DRY-конструкции

С какими подходами синтаксис понятен?

1. **текстовые макросы**

```
#define SQR(a) (a)*(a)

SQR(42)
```

1. Синтаксис: вопрос про DRY-конструкции

С какими подходами синтаксис понятен?

1. текстовые макросы
2. **синтаксические макросы**

```
(defmacro my-min [a b]
  (if (< a b) a b))

(println (my-min 42 39))
```

1. Синтаксис: вопрос про DRY-конструкции

С какими конструкциями синтаксис понятнее?

1. текстовые макросы
2. синтаксические макросы
3. **Java generics**

```
class Couple<T> {...}

new Couple<Integer>(42, 39)
```

1. Синтаксис: вопрос про DRY-конструкции

С какими подходами синтаксис понятен?

1. текстовые макросы
2. синтаксические макросы
3. Java generics
4. **шаблоны C++**

```
template <class T>
class Couple {
    T first, second;
    ...
};

Couple<int> p(42, 39);
```

1. Синтаксис: вопрос про DRY-конструкции

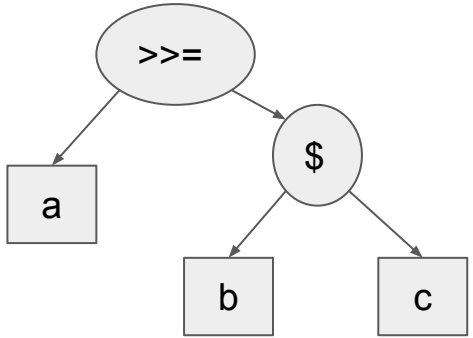
С какими подходами синтаксис понятен?

1. текстовые макросы
2. синтаксические макросы
3. Java generics
4. шаблоны C++
5. **AST-трансформации**

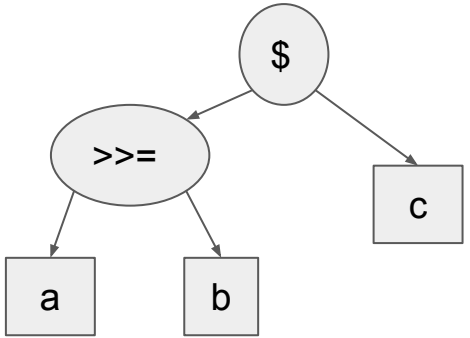
```
def m(int a, int b, int c) {
 expect:
  Math.max(a, b) == c

 where:
  a | b | c
  1 | 3 | 3
  7 | 4 | 4
  0 | 0 | 0
}
```

1. Синтаксис: нелокальная информация

```
a >>= b $ c
```



or




```
infixr 0 $

infixl 1 >>=
```

1. Синтаксис: неоднозначности

- C++: `a * b`
  - `width * height` *или* `string* ptr`
- приведение типа: `(String) o`
  - `(some.long.dotted.name) id`
  - `o as String` лучше
- декларация: `String s`
  - `some.long.dotted.name id` *или* `(`
  - `val s: String` лучше
- параметризация: `List<A>`
  - `(some.long.name<some.long.name >` *или* `)`
  - напечатали `X<`, вставлять `>`?

1. Синтаксис: сложный разбор

GHC с его 50+ языковыми расширениями

```
{-# LANGUAGE RecordWildCards, ScopedTypeVariables,
             BangPatterns, CPP, ... #-}
```

1. Синтаксис: медленный разбор

```
#define A1 1
#define A2 A1+A1+A1+A1
#define A3 A2+A2+A2+A2
#define A4 A3+A3+A3+A3
#define A5 A4+A4+A4+A4
#define A6 A5+A5+A5+A5
#define A7 A6+A6+A6+A6
#define A8 A7+A7+A7+A7
#define A9 A8+A8+A8+A8

int a = A9;
```

1. Синтаксис: шаблонизаторы

JSP:

```
<a href=${quotedUrl}${otherAttrs}>link${closeTag}text
```

```
<a href=  >link text
```

C++:

```
#define START(NAME) class NAME { void doit() {
#define END }}

START(foobar)
   printf("hello");
END
```

1. Синтаксис: ответ про DRY-конструкции

С какими подходами синтаксис понятен?

✖1. текстовые макросы (#define)
✖2. синтаксические макросы (defmacro)
✔3. Java generics
✔4. шаблоны C++
✖5. AST-трансформации (Groovy)

IDE нужно определить:

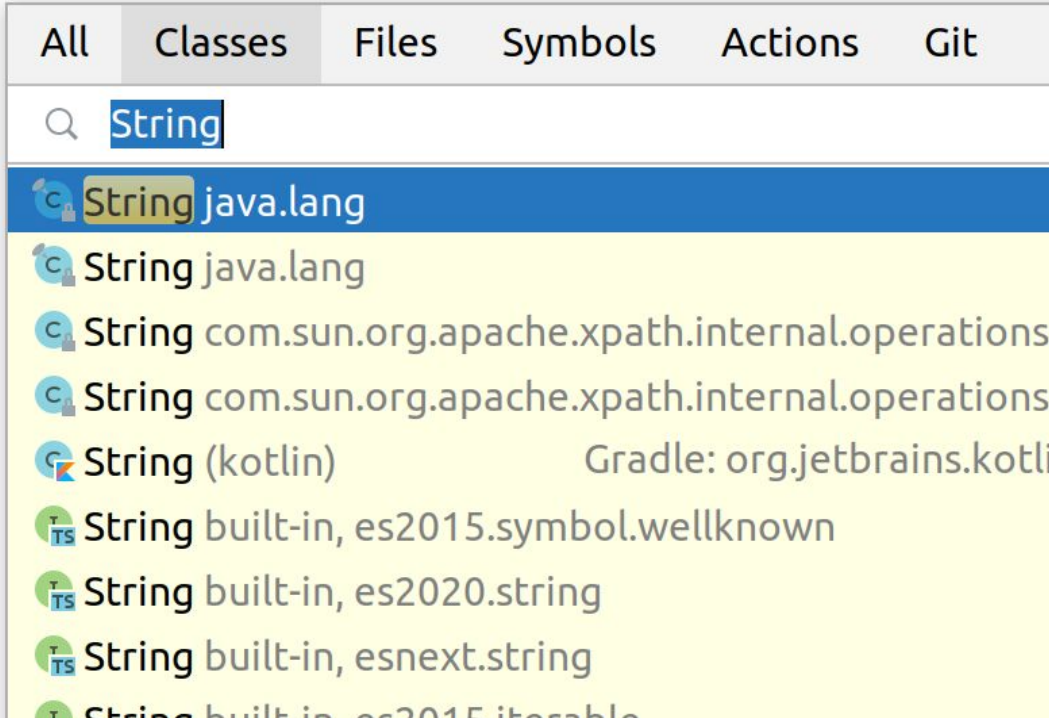
0. По одному файлу
1. Синтаксис
2. **Декларации**

   имена, виды

3. Ссылки
4. Подсказки для резолва
5. Или скомпилировать?

Зачем:

для быстрой навигации



2. Декларации: вопрос про Spring

Как легко понять, что тут задекларирован `someBean`?

2. Декларации: вопрос про Spring

Как легко понять, что тут задекларирован `someBean`?

1. **`@Bean/@Component`**

```
@Configuration
class Conf {

 @Bean
 SomeClass someBean() { … }

}

@Component
class SomeBean { … }
```

2. Декларации: вопрос про Spring

Как легко понять, что тут задекларирован `someBean`?

1. `@Bean/@Component`
2. **Custom annotation**

```
@Component
@Primary
@interface PrimaryComponent {}

@PrimaryComponent
public class SomeBean { … }
```

2. Декларации: вопрос про Spring

Как легко понять, что тут задекларирован `someBean`?

1. `@Bean/@Component`
2. Custom annotation
3. **Kotlin DSL**

```
beans {
  bean<SomeClass>(“someBean”)
}
```

2. Декларации: вопрос про Spring

Как легко понять, что тут задекларирован `someBean`?

1. `@Bean/@Component`
2. Custom annotation
3. Kotlin DSL
4. **XML-тег**

```
<bean name="someBean"
      class="SomeClass">
```

2. Декларации: кодогенерация при компиляции

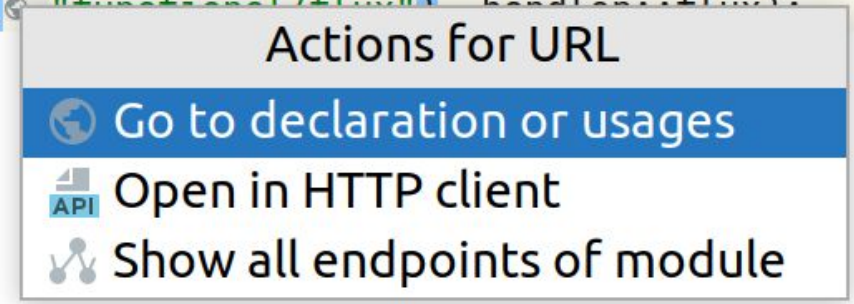
```
@lombok.Builder
class Person {
    String first;
    String last;
}

Person.builder().first('Peter').last('Gromov').build()
```

2. Непонятные декларации: в вызовах

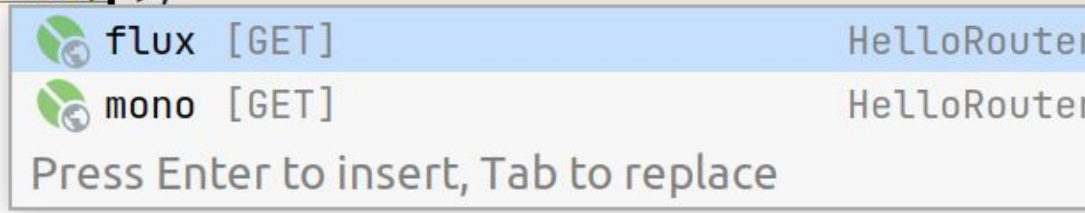
```
<?php
namespace N {
  class Foo {}
  class_alias('N\Foo', 'N\Bar');
  $o = new Bar;
}


JLoader::registerAlias(‘Alias’, ‘SomeClass’);
```

2. Непонятные декларации: в вызовах, Java

```
RouterFunctions
  .route(GET("/functional/mono"), handler::mono)
  .andRoute(GET(
```




```
new URL("http://localhost/functional/");
```



2. Декларации: ответ про Spring

Как легко понять, что тут задекларирован `someBean`?

1. ✔ `@Bean/@Component`
2. ✔ Custom annotation: `@PrimaryComponent`
3. ✖ Kotlin DSL: `bean<SomeClass>("someBean")`
4. ✔ XML-тег: `<bean name="someBean" class="SomeClass`

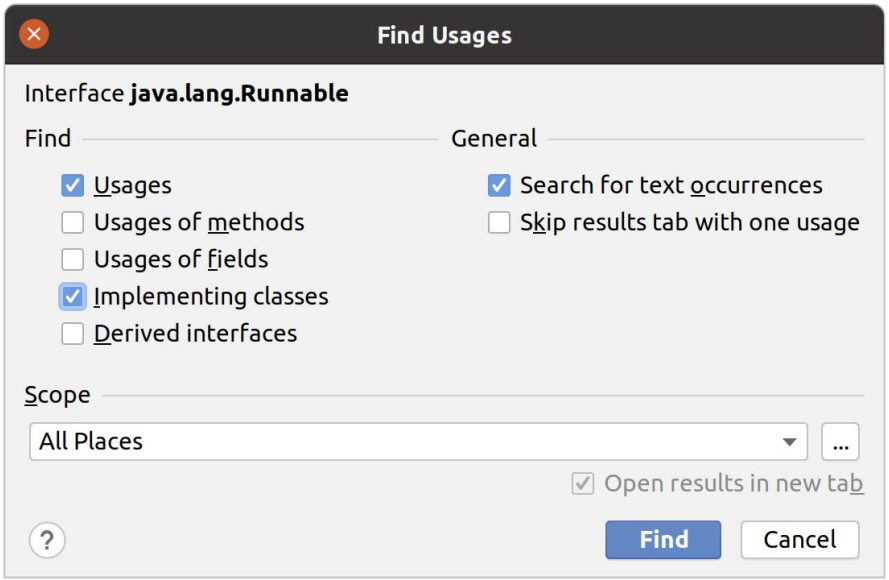
IDE нужно определить:

0. По одному файлу
1. Синтаксис
2. Декларации
3. **Ссылки (имена)**
4. Подсказки для резолва
5. Или скомпилировать?

Зачем:

поиск ссылок

рефакторинги



3. Ссылки: вопрос про Find Usages

Какие implicits не замедляют Find Usages?

1. **импорты** `java.lang, groovy.lang, etc`

3. Ссылки: вопрос про Find Usages

Какие implicits не замедляют Find Usages?

1. импорты
2. **преобразования**

```
val cm: Centimeters = Meters(2.5)

implicit def toCm(m: Meters) =
  Centimeters(m.value * 100)
```

3. Ссылки: вопрос про Find Usages

Какие implicits не замедляют Find Usages?

1. импорты
2. преобразования
3. **значения+параметры**

```
def write
  (s: String)(implicit c: Color)

write(“hello”)

implicit val red: Color =
  Color("red")
```

3. Ссылки: вопрос про Find Usages

Какие implicits не замедляют Find Usages?

1. импорты
2. преобразования
3. значения+параметры
4. **наследование**

```
invokeLater(() → doSomething())

invokeLater(this::doSomething)
```

3. Ссылки: вопрос про Find Usages

Какие implicits не замедляют Find Usages?

1. импорты
2. преобразования
3. значения+параметры
4. наследование
5. **типы переменных**

```
var x = “some string”;
```

3. Ссылки: как работает поиск

1. Текстовый поиск файлов с именем
2. Резолв найденных ссылок

`getName` или `add` искать долго

3. Неочевидные ссылки: в строках

- `Paths.get(“/home/peter”)`
- `IdeBundle.message(“dialog.title”)`
- `Class.forName(“java.lang.String”)`
- `new URL(“https://jpoint.ru/”)`
- `context.getBean(“parser”)`
- …
- Избегайте!

3. Неочевидные ссылки: неявные вызовы

`a + b`

в Java: `toString / String.valueOf`

в Groovy/Kotlin/Scala/etc: `plus`

3. Ссылки: ответ про Find Usages

Какие implicits не замедляют Find Usages?

✔️ 1. импорты: `java.lang.*, org.gradle.*`
✖️ 2. преобразования: `implicit def a2b(…) …`
✖️ 3. значения+параметры: `implicit val v = …`
✖️ 4. наследование: `new Intf() { … }` *vs* `() ⟶ {…}`
✔️ 5. типы переменных: `var i = 2`

[Как мы в IntelliJ IDEA ищем лямбда-выражения]

IDE нужно определить:

0. По одному файлу

1. Синтаксис

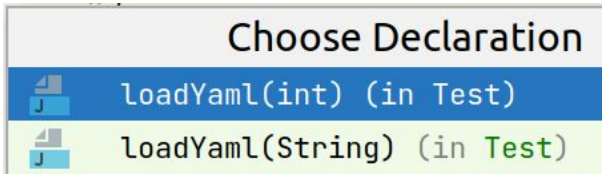
2. Декларации

3. Ссылки

4. Подсказки для резолва

5. Или скомпилировать?

Для всего: навигация, быстрая подсветка и автодополнение, поиск, рефакторинги



Меньше искать ⇨ быстрее IDE

Однозначный резолв ⇨ надёжные рефакторинги

Типизация имеет значение

4. Подсказки: вопрос про типы

Где можно убрать типы, сохранив быстрый резолв?

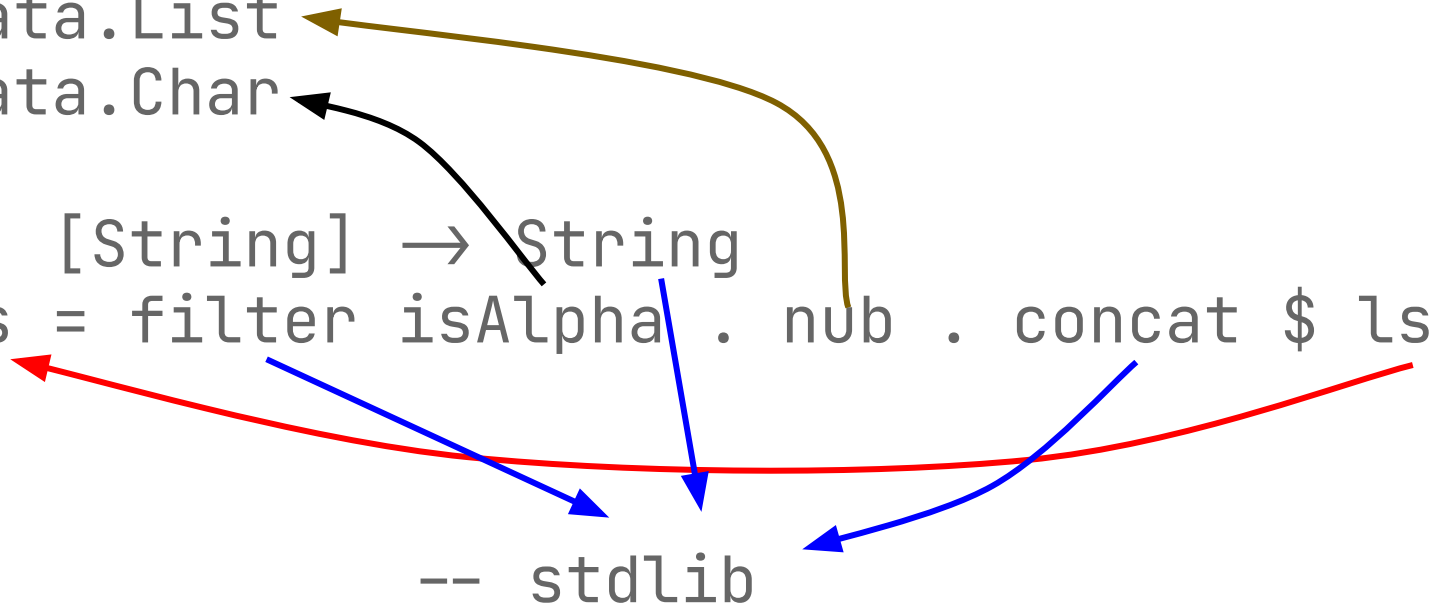
1. на локальных переменных: `var a = 42`
2. на методах/функциях: `fun foo() = “hello”`
3. на полях/свойствах: `val foo = “hello”`
4. на параметрах функций: `sum a b = a + b`
5. на лямбдах: `submit(o → process(o))`

4. Подсказки: макросы

```
template<typename T> void foo(T t) {
   t.bar();
}

class C { void bar(); };
C c;
foo(c);


template<typename T, typename =
  std::enable_if<std::is_base_of<C, T>::value>>
```

4. Подсказки: идеально (Haskell)

```
import Data.List
import Data.Char

myFunc :: [String] → String
myFunc ls = filter isAlpha . nub . concat $ ls

                    -- stdlib
```



4. Подсказки: Java

```
package pkg1;
import static pkg2.Class2.*;
class Outer extends Super1 {
  class Inner extends Super2 {
    Type t = qualifier(() → {}).method();
  }
}
```

4. Подсказки: Java

```
package pkg1;
import static pkg2.Class2.*;
class Outer extends Super1 {
  class Inner extends Super2 {
    Type t = qualifier(() → {}).method();
  }
}
```

4. Подсказки: Java

```
package pkg1;
import static pkg2.Class2.*;
class Outer extends Super1 {
  class Inner extends Super2 {
    Type t = qualifier(() → {}).method();
  }
}
...
QType1 qualifier(Runnable r) {...}
QType2 qualifier(BooleanSupplier b) {...}
```

4. Подсказки: ответ про типы

Где можно убрать типы, сохранив быстрый резолв?

1. на локальных переменных
2. на методах/функциях
3. на полях/свойствах
4. на параметрах функций
5. на лямбдах

4. Подсказки: ответ про типы

Где можно убрать типы, сохранив быстрый резолв?

✔️ 1. **на локальных переменных**
2. на методах/функциях
3. на полях/свойствах
4. на параметрах функций
5. на лямбдах

```
var v100 = new Type();
...
var v1 = v2.field;
var v = v1.anotherMethod();
v.someMethod();
```

4. Подсказки: ответ про типы

Где можно убрать типы, сохранив быстрый резолв?

1. на локальных переменных ✅
2. **на методах/функциях** ✅
3. на полях/свойствах
4. на параметрах функций
5. на лямбдах

```
fun method() = method2()

fun method2() =
    method3(42).method4()

fun <T> method3(a: T) =
    AnotherClass.mx(a)

fun method4(): Int = smth()
```

Бинарная совместимость

4. Подсказки: ответ про типы

Где можно убрать типы, сохранив быстрый резолв?

1. ✅ на локальных переменных
2. ✅ на методах/функциях
3. ✅ **на полях/свойствах**
4. на параметрах функций
5. на лямбдах

```
val p1 =
    AnotherObject.method().p2
```

4. Подсказки: ответ про типы

Где можно убрать типы, сохранив быстрый резолв?

1. ✅ на локальных переменных
2. ✅ на методах/функциях
3. ✅ на полях/свойствах
4. ❌ **на параметрах функций**
5. на лямбдах

```
showLen xs =
  show (length xs)
```

4. Подсказки: ответ про типы

Где можно убрать типы, сохранив быстрый резолв?

✅ 1. на локальных переменных
✅ 2. на методах/функциях
✅ 3. на полях/свойствах
❌ 4. на параметрах функций
✅ **5. на лямбдах**

```
call(() → { … });

void call(Runnable r) { … }
```

IDE нужно определить:

0. По одному файлу
1. Синтаксис
2. Декларации
3. Ссылки
4. Подсказки для резолва
5. **Или скомпилировать?**

Можно резолвить и искать по class-файлам

Ссылки разрешены, кодогенераторы отработали

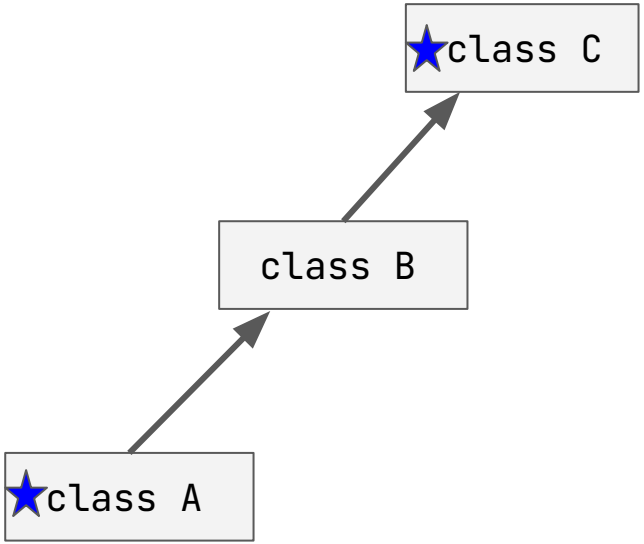
Eclipse vs IntelliJ IDEA

5. Компиляторы: проблемы использования

- рассинхронизация
- build-системы
- в класс-файлах есть не всё
  - фреймворки и ссылки в строках
  - unused import
  - константы: `if (DEBUG) doExpensiveLogging();`

5. Компиляторы: ДОЛГО!!!

- особенно с несколькими языками
- инкрементальность непроста
- ClassLoader



Так с чем же дружат IDE?

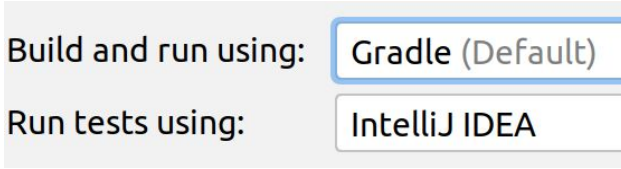
- со всеми языками и фреймворками, но по-разному
- лучше с ограничениями
  - по одному файлу определить:
  - синтаксис
  - декларации
  - ссылки
  - подсказки для резолва

О пользе ограничений

- людям легче понимать код
- но как же выразительность?
- нужна креативность!
- больше нарушений ⇨ медленнее IDE
- быстрый анализ ⇨ меньше ждать, меньше CPU usage

Как ускорить себе IntelliJ IDEA прямо сейчас

- статически типизировать
- писать типы на публичных методах/полях
- называть символы уникально
- поменьше `#include`, макросов, `implicit def/val`
- не сотни перегруженных методов
- не смешивать языки
- не делегировать компиляцию Gradle



Grazie!

peter@jetbrains.com
@donnerpeter