

Интерпретатор языка “Scheme”

**Курсовая работа по дисциплине “Конструирование
компиляторов”**

**Выполнил студент: Владиславов А.Г., ИУ9-72Б
Научный руководитель: Синявин А.В.**

План презентации

- Краткий обзор технологий
- Описание структуры интерпретатора
- Описание написанного исходного кода
- Тестирование
- Итоги

Scheme

- Функциональный язык программирования
- Диалект Lisp
- Динамически типизирован
- Использует префиксную нотацию для вычислений
- Изучается на кафедре ИУ9 на 1 курсе



Описание Scheme в БНФ

```
1 <program> ::= <expression>
2 <expression> ::= <constant> | <variable> | <procedure-call> |
   <conditional> | <lambda> | <definition>
3 <constant> ::= <number> | <boolean> | <string> | <char> | <symbol>
4 <number> ::= [0-9]+
5 <boolean> ::= #t | #f
6 <string> ::= ".*"
7 <char> ::= #\|.
8 <symbol> ::= [a-zA-Z]+
9 <variable> ::= [a-zA-Z]+
10 <procedure-call> ::= (<procedure> <expression>*)
11 <procedure> ::= + | - | * | / | number? | not | boolean? | char? |
   string? | symbol? | eq? | eqv? | > | < | car | cdr | cons | list |
   pair? | null? | set-car! | set-cdr! | quote | quasiquote | define |
   lambda | if | set! | begin | cond | else | load | eof-object? | read
   | display | write | newline | flush | port? | input-port? |
   output-port? | textual-port? | binary-port? | current-input-port |
   current-output-port | eval | null-environment |
   scheme-report-environment | interaction-environment |
   current-environment | in-environment | make-environment |
   environment? | call-with-current-continuation | procedure? | apply
12 <conditional> ::= (if <predicate> <expression> <expression>)
13   | (cond (<predicate> <expression>)+ [else <expression>])
14 <predicate> ::= <expression>
15 <lambda> ::= (lambda (<variable>*) <expression>)
16 <definition> ::= (define <variable> <expression>)
```

Выбор языка для написания интерпретатора Kotlin

- Объектно-ориентированный язык программирования
- Используется для мобильной, серверной и десктоп разработки
- Имеет JVM, LLVM и JS бэкенды компиляторов



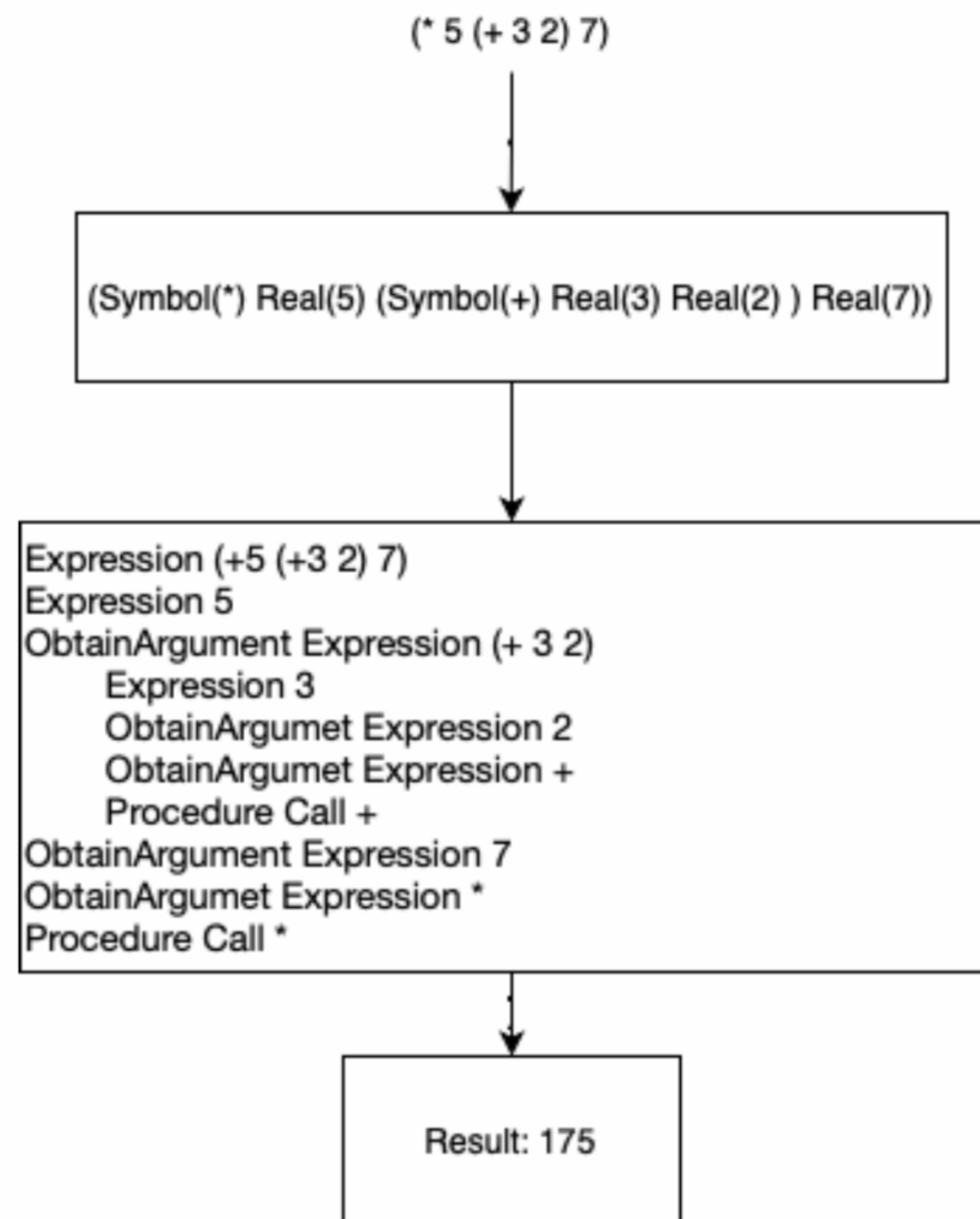
Проектирование

1/2

- Интерфейс командной строки
- На вход — программный код на языке Scheme
- На выход — результат выполнения кода

Проектирование

2/2



Разработка 1/3

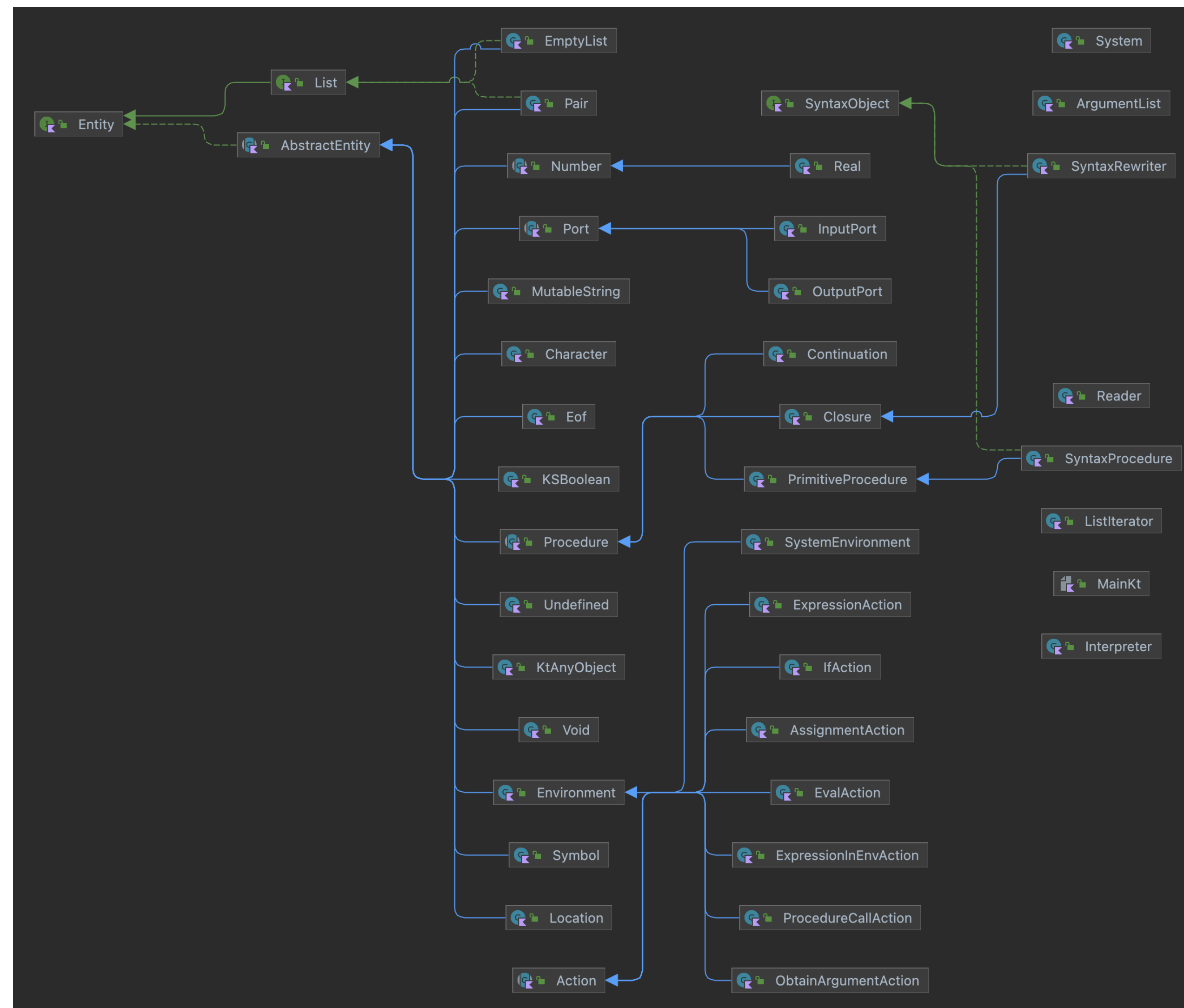
Базовые сущности, набор методов

- Entity
- Action

```
1 interface Entity : Serializable {
2
3     fun eval(env: Environment, continuation: Continuation): Entity?
4
5     fun analyze(env: Environment): Entity
6
7     fun optimize(env: Environment): Entity
8
9     fun write(out: PrintWriter)
10
11    fun display(out: PrintWriter)
12
13    fun toWriteFormat(): String
14
15    override fun toString(): String
16 }
```

```
1 abstract class Action(
2     val env: Environment?,
3     var next: Action?,
4 ) : Serializable {
5
6     fun andThen(action: Action): Action {
7         action.next = this.next
8         this.next = action
9         return action
10    }
11
12    abstract fun invoke(arg: Entity, cont: Continuation): Entity?
13
14    protected fun interface Printer {
15        fun print(port: OutputPort)
16    }
17
18    protected fun trace(doo: Printer, env: Environment?) {
19        if (env?.interpreter?.traceEnabled == true) {
20            val cout = env.out ?: return
21            val actionName = this.javaClass.simpleName.replace("Action",
22                "")
23            cout.printf("%s ", actionName)
24            doo.print(cout)
25        }
26    }
27 }
```


Диаграмма классов



Разработка 3/3

Примитивные процедуры

- name — имя, как в программе;
- definitionEnv — окружение, в котором используется примитив;
- keyword — флаг, показывающий, является примитив частью синтаксиса;
- minArgs — минимальное число аргументов для использования примитива;
- maxArgs — максимальное число аргументов для использования примитива;
- comment — короткая заметка о примитиве;
- documentation — опциональное поле, более долгое объяснение, возможно с примером использования;

Категория	Функции, относящиеся к категории
Операции с числами	+, -, *, /, number?
Операции с логическими значениями	not, boolean?
Операции с символами	char?
Операции со строками	string?
Операции с символами	symbol?
Равенства и неравенства	eq?, eqv?, >, <
Операции с парами и списками	car, cdr, cons, list, pair?, null?, set-car!, set-cdr!,
Синтаксические операции	quote, quasiquote, define, lambda, if, set!, begin, cond, else
Системный интерфейс	load
Ввод	eof-object?, read
Вывод	display, write, newline, flush,
Операции с портами ввода/вывода	port?, input-port?, output-port?, textual-port?, binary-port?, current-input-port, current-output-port
Вычисление	eval, null-environment, scheme-report-environment, interaction-environment, current-environment, in-environment, make-environment, environment?
Функции управления	call-with-current-continuation, procedure?, apply

Тестирование

- Unit-тесты на основной функционал языка
- Тестирование запуском простого интерпретатора моим интерпретатором
- Запуск тестов на каждом Pull Request



Итог

- Написан интерпретатор
- Есть тестирование
- Возможны значительные модификации
 - Уход от JVM
 - Написание графического пользовательского интерфейса