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

```
<program> ::= <expression>
<expression> ::= <constant> | <variable> | <procedure-call> |
    <conditional> | <lambda> | <definition>
<constant> ::= <number> | <boolean> | <string> | <char> | <symbol>
<number> ::= [0-9]+
<boolean> ::= #t | #f
<string> ::= ".*"
<char> ::= #\\.
<symbol> ::= [a-zA-Z]+
<variable> ::= [a-zA-Z]+
<procedure-call> ::= (<procedure> <expression>*)
<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
<conditional> ::= (if <predicate> <expression> <expression>)
        | (cond (<predicate> <expression>)+ [else <expression>])
<predicate> ::= <expression>
<lambda> ::= (lambda (<variable>*) <expression>)
<definition> ::= (define <variable> <expression>)
```

Выбор языка для написания интерпретатора

Kotlin

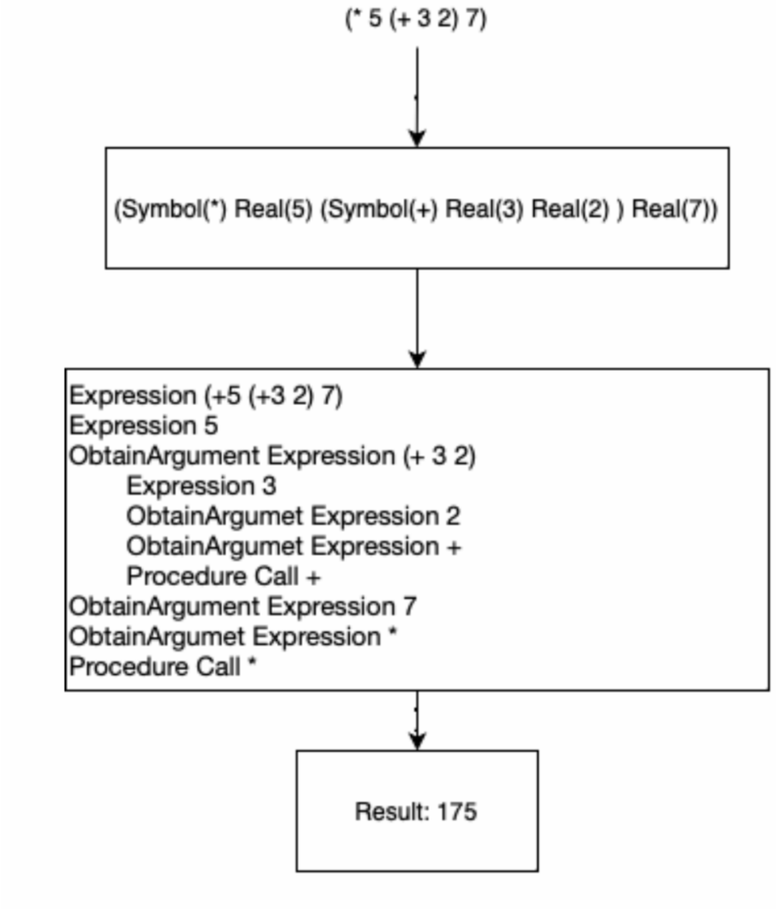
- Объектно-ориентированный язык программирования
- Используется для мобильной, серверной и десктоп разработки
- Имеет JVM, LLVM и JS бэкенды компиляторов

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

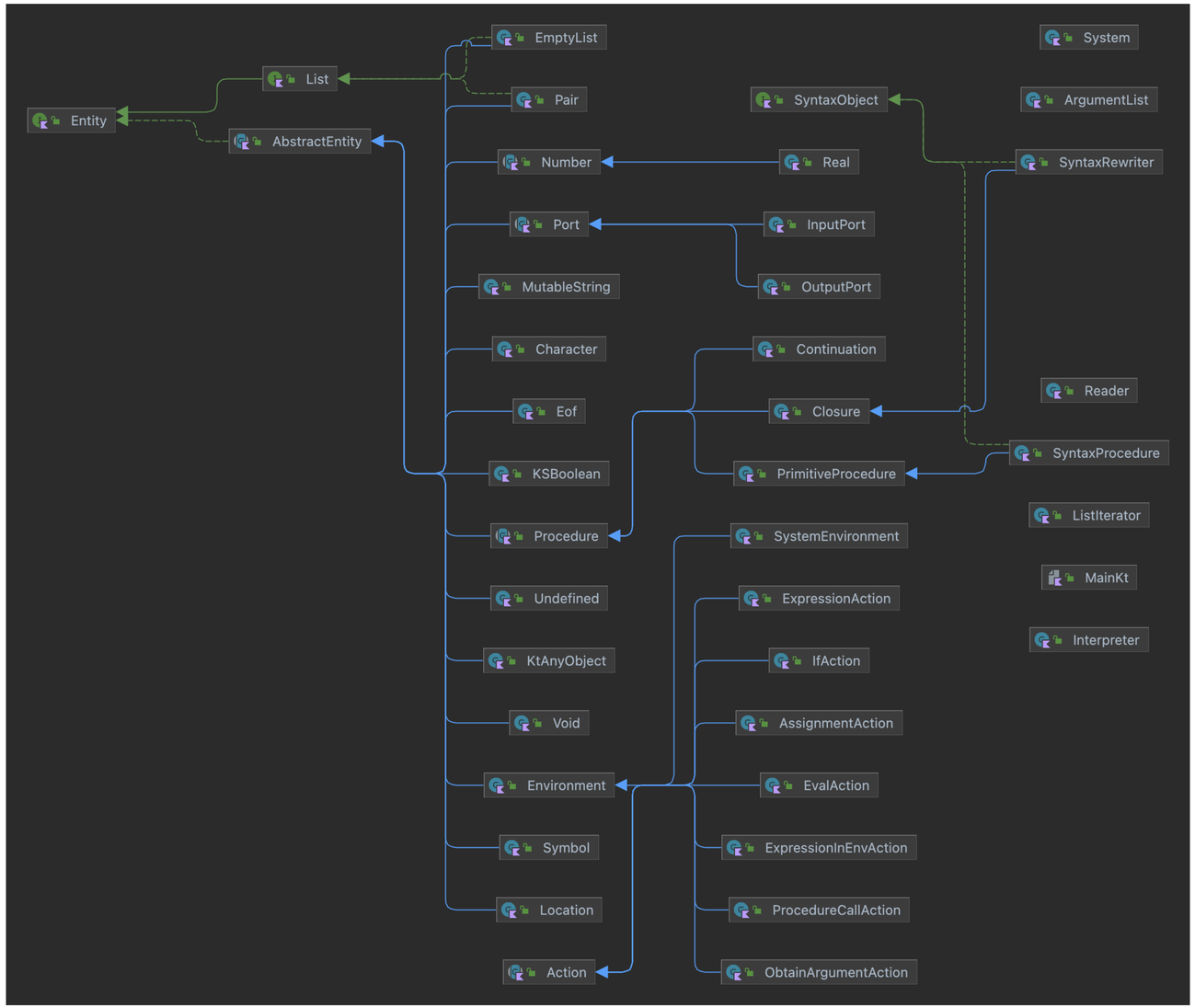
- Entity
- Action

```kotlin
interface Entity : Serializable {

    fun eval(env: Environment, continuation: Continuation): Entity?

    fun analyze(env: Environment): Entity

    fun optimize(env: Environment): Entity

    fun write(out: PrintWriter)

    fun display(out: PrintWriter)

    fun toWriteFormat(): String

    override fun toString(): String
}
```

```kotlin
abstract class Action(
    val env: Environment?,
    var next: Action?,
) : Serializable {

    fun andThen(action: Action): Action {
        action.next = this.next
        this.next = action
        return action
    }

    abstract fun invoke(arg: Entity, cont: Continuation): Entity?

    protected fun interface Printer {
        fun print(port: OutputPort)
    }

    protected fun trace(doo: Printer, env: Environment?) {
        if (env?.interpreter?.traceEnabled == true) {
            val cout = env.out ?: return
            val actionName = this.javaClass.simpleName.replace("Action",
    "")
            cout.printf("%s ", actionName)
            doo.print(cout)
        }
    }
}
```

Разработка 2/3

Диаграмма классов



Разработка 3/3
Примитивные процедуры

- name — имя, как в программе;
- definitionEnv — окружение, в котором используется примитив;
- keyword— флаг,показывающий,являетсяпримитивчастьюсинтаксиса;
- 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 катеригии
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

Исходный код

https://github.com/AndVl1/KScheme/