Курсовая работа

РАЗРАБОТКА КОМПИЛЯТОРА ПОДМНОЖЕСТВА ПРОЦЕДУРНОГО ЯЗЫКА

**Программа и методика испытаний**

**643.02068048.00001-01 51 01**

**Листов 9**

2023

АННОТАЦИЯ

Документ "Программа и методика испытаний" описывает процесс испытаний разработанного компилятора процедурного языка, предназначенного для преобразования исходного кода в байт-код

СОДЕРЖАНИЕ

[1. Объект испытаний 4](#_Toc135314113)

[2. Цель испытаний и требования к программе 5](#_Toc135314114)

[3. Средства и порядок испытаний 6](#_Toc135314115)

[4. Пример проведения испытаний 7](#_Toc135314116)

1. Объект испытаний

Объект испытаний включает в себя компилятор процедурного языка, разработанный для преобразования исходного кода на процедурном языке в байт-код для JVM.

Компилятор процедурного языка состоит из следующих компонентов:

1. Лексический и синтаксический анализатор (Parser.java).
2. Описание класса (CompilationUnit.java).
3. Генератор байт-кода для JVM (BytecodeGenerator.java).
4. Цель испытаний и требования к программе

Цель испытаний заключается в проверке работоспособности компилятора процедурного языка и правильности преобразования исходного кода в байт-код для JVM. Основные аспекты, подлежащие проверке, включают:

1. Корректность лексического и синтаксического анализа.
2. Создание описания класса из дерева токенов.
3. Генерация бай-кода для JVM.
4. Обработка ошибок.
5. Средства и порядок испытаний

Для проведения испытаний компилятора процедурного языка в байт-код для JVM будут использованы следующие средства:

* Компьютер с установленной операционной системой, на которой установлена JVM версии 1.8 для исполнения скомпилированного кода.
* Редактор исходного кода: программное обеспечение, позволяющее создавать и редактировать исходный код на процедурном языке.
* Компилятор процедурного языка: разработанный компилятор, способный преобразовывать исходный код на процедурном языке в байт-код для JVM
* JVM — это виртуальная машина Java, позволяющая запускать файлы расширения .class

Испытания компилятора процедурного языка в байт-код для JVM будут проводиться в следующем порядке:

1. Подготовка тестовых программ
2. Запуск компилятора
3. Проверка результатов компиляции
4. Исполнение скомпилированного кода
5. Пример проведения испытаний

В качестве исходного кода (рис. 1) был взят код с функцией для вычисления суммы двух чисел.

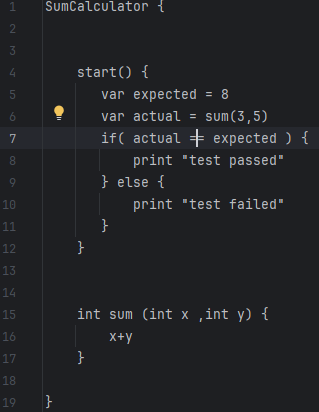


Рисунок 1 – Исходный код программы в файле «SumCalculator.w»

Результатом лексического и синтаксического анализа будет дерево токенов сгенерированное Antlr4 (рис. 2).

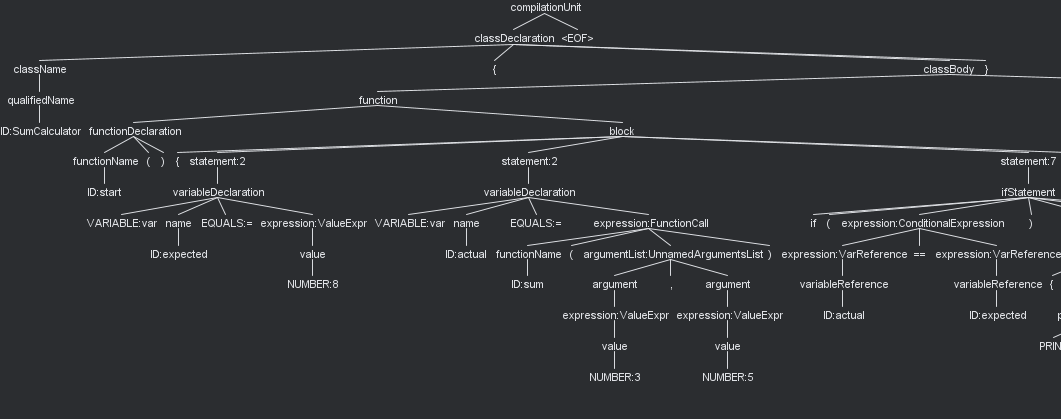


Рисунок 2.1

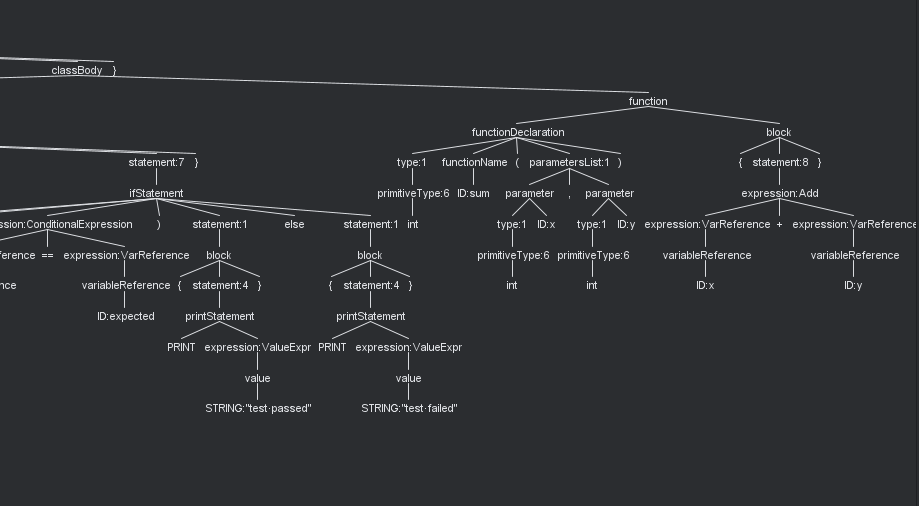


Рисунок 2.2

Результатом создания описания класса является экземпляр класса CompilationUnit в котором и хранится описания класса его полей, глобальных, локальных переменных, их типов и т.д. (рис 3.)

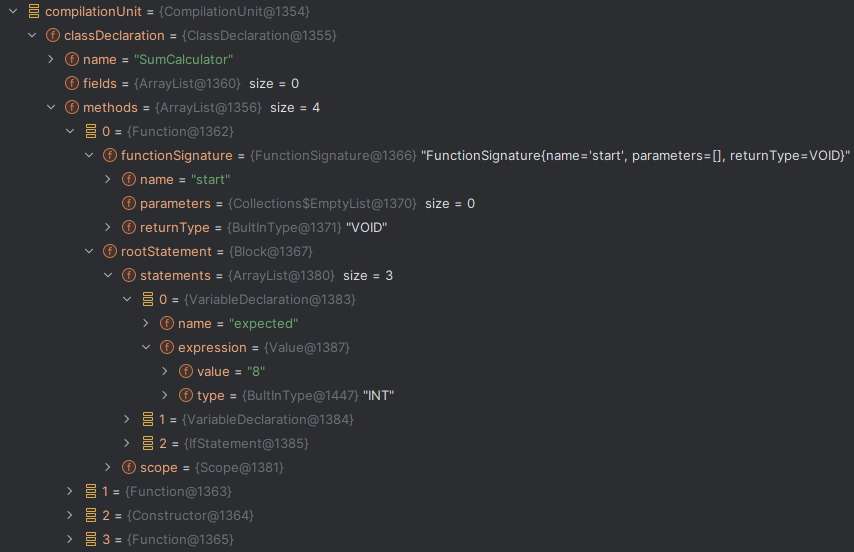


Рисунок 3

После завершения работы всей программы в файле «SumCalculator.class» будет находиться байт-код для JVM (рис. 3).

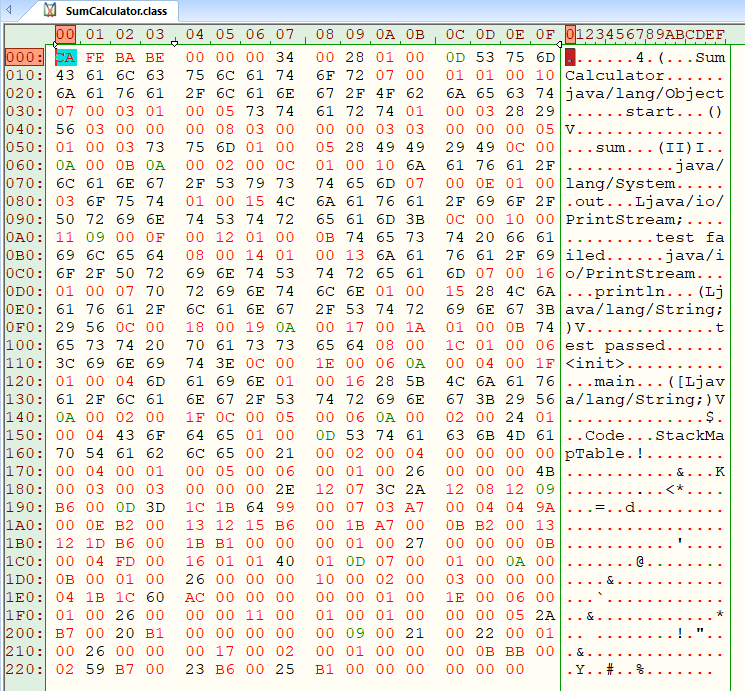


Рисунок 3

Если же в коде были допущены различные ошибки, то компилятор должен сообщить о них.

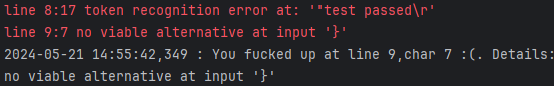


Рисунок 4– Пример сообщения об ошибке.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц) в докум. | № докумен-та | Входящий № сопроводит. докум. и дата | Подп. | Дата |
| Изм. | изменен-ных | заменен-ных | новых | аннули-рованных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |