Reader class

Реализовать класс Reader, читающий посимвольно текстовые файлы.

Интерфейс:
public static char[] readFile(fileName: String), где:
fileName - абсолютный или относительный путь до файла
возвращаемое значение - массив символов файла, включая переводы строк

Lexer class

Реализовать класс Lexer, осуществляющий лексический разбор текста.

Интерфейс:
public static Token[] makeTokens(char[] text), где:
text - массив символов текста
возвращаемое значение - массив токенов
в случае ошибка разбора выбрасывается исключение LexerException, содержащее информацию о сути ошибки и её позиции в тексте

Список лексем и правила их разбора взять из спецификации.

Класс токена зафиксирован, все необходимые изменения обсуждать с командой, реализующей Parser.

Parser class

Реализовать класс Parser, осуществляющий синтаксический разбор массива токенов.

Интерфейс:
public static IR parseProgram(Token[] tokens), где:
tokens - массив токенов
возвращаемое значение - внутреннее представление программы
в случае ошибка разбора выбрасывается исключение ParserException, содержащее информацию о сути ошибки и её позиции в тексте

Правила разбора взять из спецификации.

Класс токена зафиксирован, все необходимые изменения обсуждать с командой, реализующей Lexer.

Класс внутреннего представления зафиксирован, все необходимые изменения обсуждать с командой, реализующей Generator.

Generator class

Реализовать класс Generator, осуществляющий генерацию байткода по внутреннему представлению программы.

Интерфейс:
public static void generateCode(IR ir, String name), где:

ir - внутреннее представление программы
name - имя файла байткода

Класс внутреннего представления зафиксирован, все необходимые изменения обсуждать с командой, реализующей Parser.

Байткод сгенерировать в текстовом формате в соответствии со спецификацией. Все необходимые изменения обсуждать с командами, реализующими VM.

Bytecode reader class

Реализовать функцию, читающую байткод программы из файла.

Интерфейс:
Bytecode* readBytecode(const char* name), где:

name - абсолютный или относительный путь до файла байткода
возвращаемое значение - указатель на прочитанную структуру байткода

Текстовый формат байткода зафиксирован спецификацией. Все необходимые изменения обсуждать с командой, реализующей генератор байткода.

Структура байткода зафиксирована, все необходимые изменения обсуждать с командой, реализующей интерпретатор.

Interpreter class

Реализовать функцию, интерпретирующую байткод.

Интерфейс:
void interpret(Bytecode* bytecode), где:
bytecode - указатель на структуру байткода

Структура байткода зафиксирована, все необходимые изменения обсуждать с командой, реализующей чтение байткода.

Интерпретацию выполнять в соответствии с семантикой, заданной в спецификации:
исполнение начинается с функции main
аргументы функции помещаются в первые регистры
результаты берутся с первых регистров
и т.д.
Все необходимые изменения обсуждать с командой, реализующей генератор байткода в компиляторе.