Фаза 1

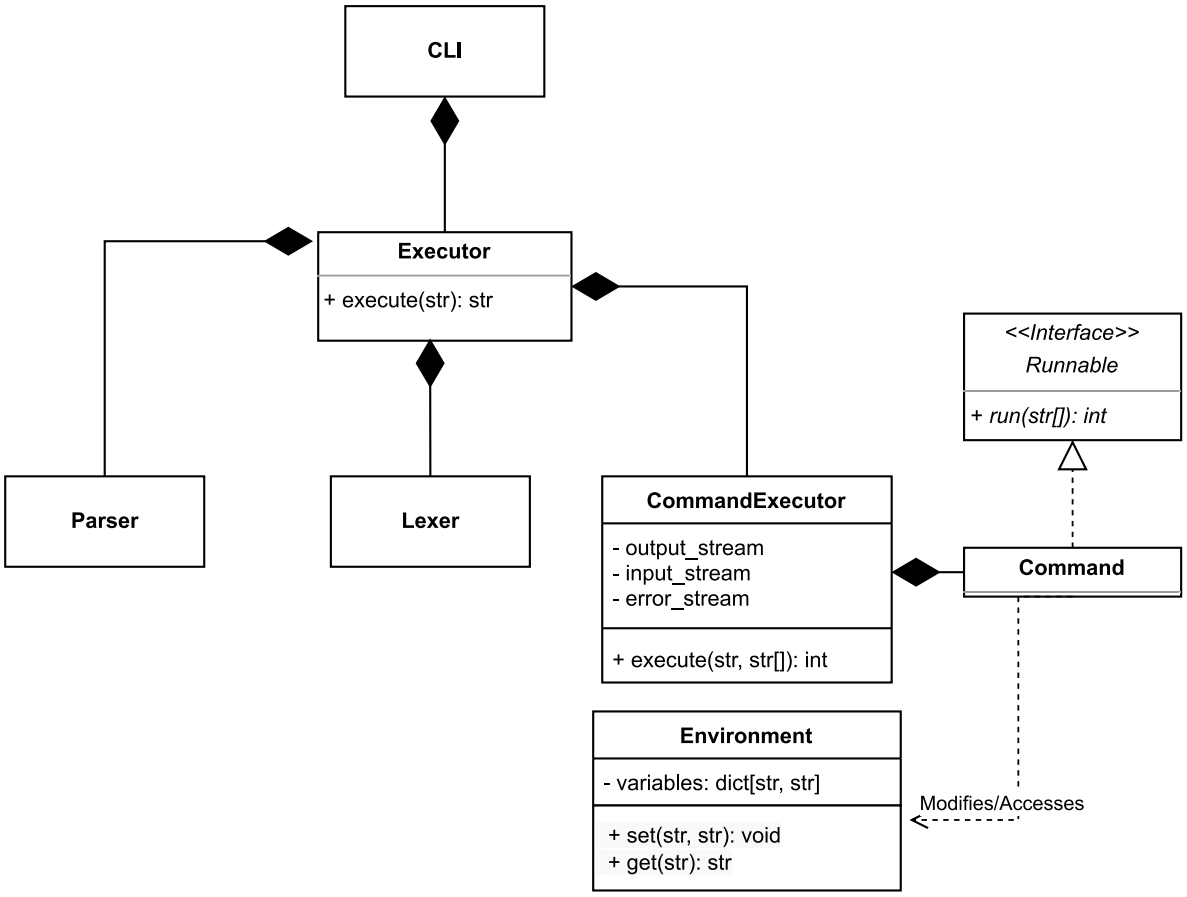

Рисунок 1. Структурная диаграмма первой фазы

CLI – класс, который будет ответственен за считывание пользовательского ввода в бесконечном цикле, передачи его компоненте Executor и последующего вывода результатов на экран.

Executor ответственен за координацию работы лексического анализатора (Lexer), парсера (Parser) и исполнителя команд (CommandExecutor). Ввод пользователя будет подаваться на вход лексическому анализатору. Полученные лексемы будут поданы на вход парсеру, который будет возвращать набор команд и их агрументы в виде той или иной структуры данных. Каждую команду будет исполнять элемент CommandExecutor. Результат исполнения команды (exit_code, output/error streams) будет обратно получать CLI.

Исполнитель команд (**CommandExecutor**) будет сам создавать команды и хранить их в словаре map[str, Command]. Комманд-экзекьютор будет иметь метод intepret(string, *args). Первый параметр будет именем команды, а *args – позиционные аргументы команды. Задание переменных окружения NEW_VAR=... также считается командой. Эта команда меняет окружение, поэтому у неё есть связь с Environment. Environment – объект, хранящий состояние. Если интерпретатор не нашел команду в словаре то будет вызвана специальная команда, ищущая требуемую команду по пути PATH.

Environment предоставляет интерфейс, позволяющий получать и записывать значение по ключу. Environment будет реализован в виде либо глобальной переменной, либо singleton-класса.

Фаза 2

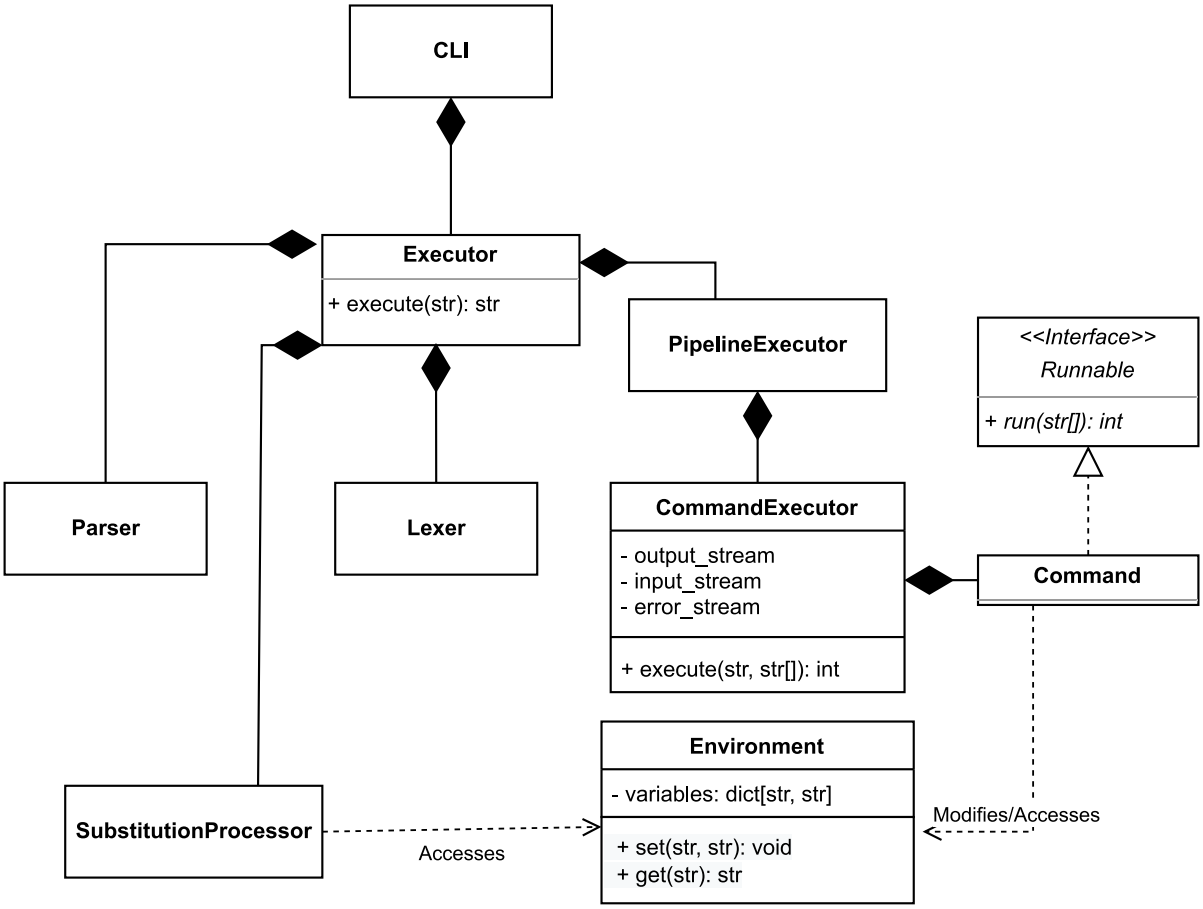

Рисунок 2. Структурная диаграмма второй фазы

В отличие от первой фазы, появляется возможность делать подстановки. Для их поддержки буден добавлен новый компонент **SubstitutionProcessor**, который будет ответственен за поддержку подстановок. Выход лексического анализатора (Lexer) будет подан на вход SubstitutionProcessor. Затем выход SubstitutionProcessor будет снова подан на вход лексеру, и только потом его выход будет подан парсеру (Parser). Чтобы делать подстановки SubstitutionProcessor будет читать переменные из Environment.

За поддержку пайпов будет отвечать **PipelineExecutor**. Parser выдаст команды и их аргументы в порядке исполнения. Исполнять их в правильном порядке, подставляя вывод предыдущей команды, будет PipelineExecutor.