Практика 1: Command Line Interface

Задача на практику по архитектуре: спроектировать простой интерпретатор командной строки, поддерживающий следующие команды:

- cat [FILE] — вывести на экран содержимое файла;
- echo — вывести на экран свой аргумент (или аргументы);
- wc [FILE] — вывести количество строк, слов и байт в файле;
- pwd — распечатать текущую директорию;
- exit — выйти из интерпретатора.

Кроме того, должны поддерживаться одинарные и двойные кавычки (full and weak quoting, то есть одинарные кавычки передают текст как есть, двойные выполняют подстановки переменных окружения с оператором $), собственно окружение (команды вида “имя=значение”), оператор $, вызов внешней программы для любой команды, которую интерпретатор не знает. Должны ещё поддерживаться пайплайны (оператор “|”), то есть перенаправление ввода и вывода. Примеры:

```
> echo "Hello, world!"
Hello, world!
> FILE=example.txt
> cat $FILE
Some example text
> cat example.txt | wc
1 3 18
> echo 123 | wc
1 1 3
> x=exit
> $x
```

При этом должны быть учтены следующие архитектурные соображения (Architectural Drivers, ключевые соображения, определяющие архитектуру):

- возможность легко добавлять новые команды (расширяемость);
- наличие возможности реализовать что-то новое из того, что умеют другие шеллы (сопровождаемость) — поскольку другие шеллы умеют очень многое, заранее предсказать точки расширения функциональности сложно, поэтому тут ожидается аккуратная объектно-ориентированная декомпозиция и архитектура, позволяющая быстро разобраться и реализовать новую функциональность;

- Архитектурное описание, как умеете (сопровождаемость) — формальное архитектурное описание мы научимся писать потом, сейчас хочется картинку и словами рассказать, как оно работает.

Собственно, что делать сейчас — выполнить анализ и определить подходы к созданию CLI, выявить как можно больше потенциальных подводных камней и способов их преодоления. Даётся некоторое время подумать, после чего ожидается живое обсуждение, в ходе которого мы обсуждаем детали того, как бы вы стали писать такую задачу.

Документально результаты должны быть закреплены в виде диаграммы классов (в любом виде), выложенной как решение задания в Teams. Если вы не рассказывали на паре, то ожидается ещё текстовое описание того, как предполагается, что система будет работать.

При этом учтите опыт предыдущих поколений, решавших эту задачу.

- Проектируйте сверху вниз. Сначала определитесь с общей структурой системы — из каких компонентов она состоит, кто за что отвечает (вспомните принцип единственности ответственности), какие компоненты о каких знают, как направлены потоки данных между компонентами, как выглядят эти данные. Только после этого переходите к проектированию самих компонентов, рекурсивно повторяя процесс. Например, на первом уровне у нас есть парсер командной строки, далее он распадается на лексический анализатор и штуку, которая строит внутреннее представление, а лексический анализатор распадается на конечный автомат со своими состояниями, каждое из которых отдельный класс.

- Опасайтесь архитектурной жадности, надо вовремя остановиться. Не пытайтесь специфицировать каждый метод и даже каждый класс системы. Обычно ответственность архитектора заканчивается на уровне наброска диаграммы классов, конкретные параметры, поля и методы можно доверить программисту. Хотя ключевые методы. особенно у интерфейсов, я бы ожидал.

В процессе проектирования постарайтесь ответить на следующие вопросы.

- Как представляются команды и пайплайны?
- Как создаются команды?
- Как они исполняются? Как взаимодействуют потоки в пайплайне?
- Кто и как выполняет разбор входной строки?
  - Кто, как и когда выполняет подстановки?
- Как представляются переменные окружения?
- Что с многопоточностью?