Анализ и подходы к решению:

На каком языке писать?

C++

Минусы:

1) не все в команде хорошо знают плюсы
2) не удобно писать программы такого вида

Python

Плюсы:

1) простой язык

Минусы:

1) не все в команде знают python
2) проблемы с утечками памяти могут быть и их будет сложно обнаружить

Java

Плюсы:

1) все в команде знают данный язык
2) отсутствие утечек памяти

Поэтому мы решили выбрать Java.

Для реализации данного интерпретатора командной строки, мы можем разбить задачу на несколько подсистем:

1. **Парсинг ввода**: Необходимо разработать механизм разбора ввода пользователя для понимания, какую команду выполнять, какие аргументы использовать и какие операции с файлами или пайплайнами должны быть выполнены.
2. **Выполнение команд**: Разработать логику выполнения каждой из команд, включая встроенные команды (например, cat, echo, wc, pwd, exit) и обработку вызовов внешних программ.
3. **Работа с переменными окружения**: Реализация механизма для установки, получения и подстановки значений переменных окружения, а также обработка оператора $.
4. **Обработка кавычек**: Реализация логики для правильного разбора аргументов, включая учет двойных и одинарных кавычек.
5. **Обработка ошибок и неизвестных команд**: Необходимо предусмотреть обработку ситуаций, когда ввод пользователя не соответствует ожидаемому формату, а также обработку неизвестных команд.
6. **Поддержка пайплайнов**: Реализация механизма для передачи вывода одной команды на вход другой через оператор "|".

Подводные камни и способы их преодоления:

1. **Безопасность**: Необходимо быть осторожным при обработке пользовательского ввода, чтобы избежать уязвимостей типа инъекций команд.
2. **Эффективность**: При работе с файлами и внешними программами следует обеспечить оптимальное использование ресурсов.

3. **Управление ресурсами**: Необходимо грамотно управлять открытыми файловыми дескрипторами, процессами и памятью.
4. **Масштабируемость**: При разработке стоит учитывать возможность расширения функциональности в будущем, например, добавление новых команд или поддержки дополнительных операторов.

Декомпозиция задачи на классы и методы

1. **Класс Parser**: Отвечает за разбор ввода пользователя и выделение команд, аргументов и операторов.
2. **Класс CommandExecutor**: Выполняет команды, как встроенные, так и внешние, используя соответствующие методы для каждой команды.
3. **Класс Environment**: Управляет переменными окружения и их значениями.
4. **Класс QuoteHandler**: Обрабатывает кавычки и выполняет подстановку переменных окружения.
5. **Класс PipelineHandler**: Реализует логику для выполнения пайплайнов.
6. **Класс ErrorHandling**: Обрабатывает ошибки, возникающие при выполнении команд и разборе ввода.