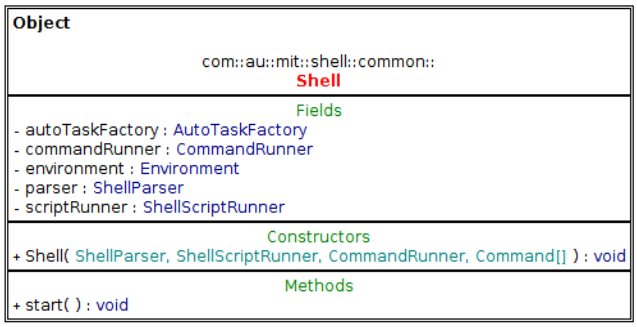
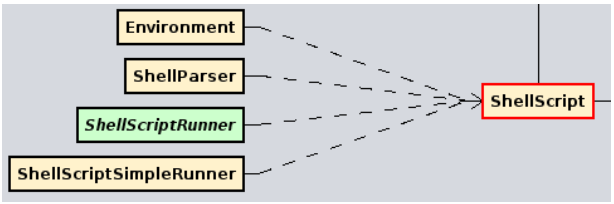
Описание структуры классов

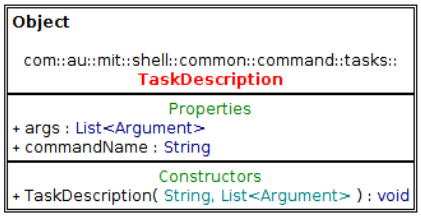
Начнем с главного класса, Shell



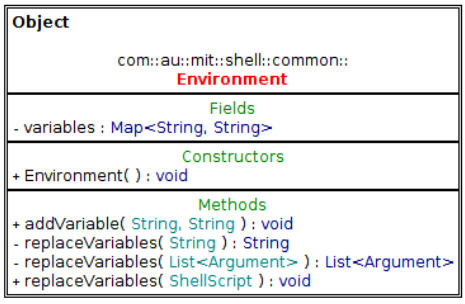
Его мы создаем, с указанием нужных нам параметров (можно было сделать через builder, но и так слишком много классов уже) и запускаем через start(). В нем у нас хранятся основные объекты для парсинга и исполнения комманд.

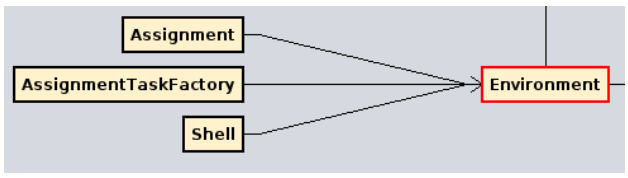


В процессе парсинга мы получаем ShellScript – по сути список TaskDescription, для комманд, объединенных через pipe. А он, в свою очередь, хранит название комманды и список ее аргументов.



После того как парсер отдал нам ShellScript, мы в нем заменяем переменные с помощью Environment.

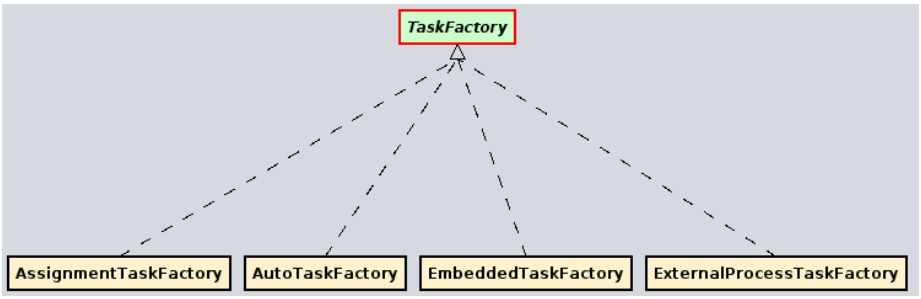




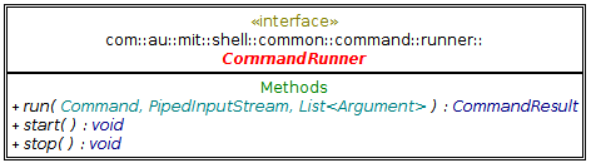
Далее, мы вызываем сохраненный в Shell'e ShellScriptRunner (интерфейс как всегда для гибкости, пока что реализует его только один класс)

Сам runner должен с помощью TaskFactory из TaskDescription сделать непосредственно различные таски

При этом есть специальный класс AutoTaskFactory, которых хранит список фабрик, и пробует поочередно создавать с их помощью таски, а именно вызывает у них метод tryCreate, который возвращает либо null, либо не null.



Далее, эти таски отдаются в CommandRunner, и у этого интерфейса есть 2 реализации: SimpleCommandRunner и MultiThreadCommandRunner.



Разница у них в том, что первый запускает команды разделенные pipe'ом поочередно, а второй запускает их в N потоков и они передают друг другу результаты выполенения через PipedInputStream и PipedOutputStream. Это позволяет делать такие вещи как:
cat /dev/urandom | echo
(Пример бесполезный, но если расширять список комманд до filter и takeFirst, то будет осмысленней)
Не слишком премудрый объект Command: