Задание №2. (25 баллов) Шаблон проектирования «фабричный метод», журналирование (logging), модульное тестирование (unit testing).

Часть 1. (15 баллов).

Написать стековый калькулятор, который принимает в качестве аргумента командой строки имя файла, содержащего команды. Если аргумента нет, то использовать стандартный поток ввода для чтения команд. Использовать вещественные числа. Реализовать следующий набор команд:

- # - строка с комментарием.
- **POP**, **PUSH** — снять/положить число со/на стек(а).
- **+ , - , * , /, SQRT –** арифметические операции. Используют один или два верхних элемента стека, изымают их из стека, помещая результат назад
- **PRINT** — печать верхнего элемента стека (без удаления).
- **DEFINE** — задать значение параметра. В дальнейшем везде использовать вместо параметра это значение.

Пример (должно вывести 2):
DEFINE a 4
PUSH a
SQRT
PRINT

Часть 2. (10 баллов).

1. Реализовать набор модульных тестов, покрывающих функционал калькулятора.
2. Реализовать журналирование процесса работы калькулятора.

Методические указания:

- Создание команд рекомендуется реализовать посредством шаблона проектирования «фабричный метод» (http://ru.wikipedia.org/wiki/Фабричный_метод_(шаблон_проектирования)).
- Загрузку классов команд при создании в фабрике осуществлять по полному квалифицированному имени класса (включая имя пакета) посредством Class.forName() с последующим созданием объектов команд методом Class.newInstance(). Фабрика конфигурируется с помощью файла содержащего соответствия между именами команд и классами, реализующими эти команды. Зависимости фабрики от конкретных классов команд (кроме корневого-абстрактного) быть не должно. Файл конфигурации должен храниться рядом с файлом класса-фабрики, и загружаться в фабрику с помощью Class.getResourceAsStream()
- Аргументы команде (тем у которых есть аргументы) на исполнение можно передавать в виде массива либо списка объектов, команда сама должна уметь интерпретировать свои аргументы
- Содержимое стека и список (лучше ассоциативный контейнер Map<String, Double>) определенных именованных параметров передавать команде в виде специального объекта — контекста исполнения
- Разработать иерархию исключений, которые будут выбрасывать команды при исполнении. В случае возникновения исключения — выводить информацию об ошибке и продолжать исполнение программы (из файла или команд вводимых с консоли)
- Для реализации модульных тестов обычно используют готовые библиотеки:

  - TestNG (http://testng.org/).
  - JUnit (http://www.junit.org/).
- Для реализации журналирования обычно используют одну из библиотек:
  - Java Logging API (http://download.oracle.com/javase/1.4.2/docs/guide/util/logging/overview.html).
  - Log4j (http://logging.apache.org/log4j/1.2/).