# ФИТ НГУ, курс Основы ООП, осенний семестр

**Задача №3. Интерфейсы. Наследование. Шаблон проектирования «фабричный метод»..**

Требуется реализовать приложение Workflow Executor.

Workflow – вычислительная схема, состоящая из набора вычислительных блоков и связей между ними. Программе подается workflow, описанный в файле.

В данной задаче рассматривается единственный тип workflow – линейный, т.е. конвейер.

Список блоков, используемых в схеме (набор блоков расширяем):

1. readfile <filename> – считывание текстового файла в память, целиком.   
   Вход – отсутствует, выход – текст.
2. writefile <filename> – запись текста в файл.  
   Вход – текст, выход – отсутствует.
3. grep <word> – выбор из входного текста строк, разделенных символами переноса строки, содержащих заданное слово <word>.  
   Вход – текст, выход – текст.
4. sort – лексикографическая сортировка входного набора строк.  
   Вход – текст, выход – текст.
5. replace <word1> <word2> – замена слова <word1> словом <word2> во входном тексте.  
   Вход – текст, выход – текст.
6. dump <filename> - сохранить пришедший текст в указанном файле и передать дальше.

Вход – текст, выход – текст.

Формат входного файла:

desc # описание блоков схемы

id1 = block1

id2 = block2

…

idN = blockN

csed

idA -> idB -> idC -> … -> idZ # описание структуры схемы

Где:

* desc, csed – ключевые слова, ограничивающие раздел описания блоков workflow.
* id1 … idN – целые, неотрицательные, неповторяющиеся числа.
* block1 … blockN – блоки из списка блоков, с обязательными параметрами.
* idA, idB … idZ – числа, принадлежащие множеству id1…idN. Могут повторяться, длина конвейера – неограничена.
* -> – ключевое слово, обозначающее связь вычислительных узлов.

Пример файла Workflow:

workflow.txt

*desc*

*1 = readfile in\_file.txt*

*2 = writefile out\_file.txt*

*3 = sort*

*10 = replace word1 word2*

*csed*

*1 -> 3 -> 10 -> 3 -> 2*

### Ограничения на вычислительную схему:

1. Блоки должны содержать нужное количество параметров.
2. Узлы схемы должны корректно соединяться, то есть тип данных на входе узла должен совпадать с типом данных предыдущего узла. Это означает, что чтение/запись файлов не может быть в середине схемы.

**Методические указания по реализации задачи:**

1. Разбор конфигурационного файла должен быть реализован в отдельном классе.
2. Создать интерфейс Worker. Классы представляющие блоки должны его реализовывать.
3. Создание блока нужно реализовать посредством шаблона проектирования «фабричный метод» ([Фабричный метод (шаблон проектирования) — Википедия](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B1%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))[)](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B1%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))).
4. Аргументы блоку нужно передавать в виде списка, блок сам должна уметь интерпретировать свои аргументы
5. Для обработки ошибок и исключительных ситуаций нужно использовать механизм исключений C++. Нужно разработать иерархию исключений, которые будут выбрасывать блоки при исполнении.
6. Текст блоку должен передаваться в виде строки или массива строк и возвращаться из блока в таком же виде.

**При сдаче задания продемонстрировать несколько рабочих и несколько нерабочих workflow.**