

Проектирование программного обеспечения (практика)

Практика 1: Введение, задача про CLI

Юрий Литвинов
yurii.litvinov@gmail.com

12.02.2020г

Формальности

- ▶ Чтобы получить зачёт, надо:
 - ▶ Сдать все домашние работы
 - ▶ Успеть сдать каждую задачу до дедлайна
- ▶ Условия, материалы и сдача домашек через <http://hwproj.me/courses/53>
 - ▶ Надо записаться на курс
 - ▶ Ссылка на пуллреквест в собственный репозиторий на GitHub
 - ▶ Задач будет несколько, так что выкладывать лучше так, чтобы их все было можно смерджить

Краткое содержание курса по практике

- ▶ Снова лекции
 - ▶ Про архитектурную документацию
 - ▶ Про UML и другие языки проектирования
 - ▶ Про реализацию паттернов
 - ▶ Про антипаттерны
 - ▶ Различные примеры архитектуры
- ▶ Небольшие задачи прямо на паре
- ▶ Большие задачи на дом

Что ожидается от решений

- ▶ Работоспособность и соответствие требованиям условия
- ▶ Наличие архитектурной документации
 - ▶ Комментарии к каждому классу, интерфейсу и public-методу
 - ▶ Краткое описание деталей реализации в README
- ▶ Следование стайлгайдам и правилам здравого смысла
- ▶ Наличие юнит-тестов
- ▶ Применение индустриальных практик: логирование, Continuous Integration, обработка исключений

Ещё комментарии

- ▶ Овердизайн и активное использование знаний с лекций приветствуются
- ▶ Обоснованность принятых решений важнее, чем техника кодирования
- ▶ Некоторые требования могут показаться ненужными — это нормально
 - ▶ Мы учимся не написанию кода, а инструментам и техникам проектирования
- ▶ Комментарии вида “у вас неправильная архитектура” будут очень редки, как ни странно

Задача про CLI

Реализовать простой интерпретатор командной строки, поддерживающий 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

Задача про CLI (продолжение)

- ▶ Должны поддерживаться одинарные и двойные кавычки (full and weak quoting)
- ▶ Окружение (команды вида “имя=значение”), оператор \$
- ▶ Вызов внешней программы через Process (или его аналоги)
 - ▶ если введено что-то, чего интерпретатор не знает
- ▶ Пайплайны (оператор “|”)

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
```


Нефункциональные требования

- ▶ Легко добавлять новые команды (расширяемость)
- ▶ Наличие возможности реализовать что-то новое из того, что умеют другие шеллы (сопровождаемость)
- ▶ Адекватное покрытие юнит-тестами (качество, сопровождаемость)
- ▶ Комментарии в коде (сопровождаемость)
- ▶ Архитектурное описание, как умеете (сопровождаемость)
- ▶ Стайлгайд (сопровождаемость)

Как сдавать

- ▶ Сделать для этого курса репозиторий
- ▶ Выложить решение в отдельную ветку
- ▶ Сделать пуллреквест к себе в мастер
- ▶ Зарегистрироваться на <http://hwproj.me/> и записаться на курс
- ▶ Приложить там ссылку на пуллреквест к решению
- ▶ Смерджить пуллреквест, когда задача зачтена
- ▶ **Дедлайн 10:00 04.03.2020**

Что делать сейчас

Первые фазы жизненного цикла

- ▶ Выполнить анализ и определить подходы к решению
- ▶ Выявить подводные камни и способы их преодоления
- ▶ Декомпозировать задачу на подсистемы, классы и методы
- ▶ Нарисовать диаграмму классов
- ▶ Словами описать принцип работы и основные принятые решения
- ▶ К концу пары все должны понимать, что и как надо кодить