

## Предметная область.

Система управления задачами (органайзер) с модульной системой.

## Введение.

Проект разделен на главное приложение, систему управления модулями (плагинами) и систему управления задачами (задачи представлены в древовидном виде) . Обобщенная схема системы управления задачами представлена на первом рисунке (файл **.svg**). Главное приложение и систему управления модулями расположены на втором рисунке (следующий файл **.svg**)

Каждый модуль может решать определенную задачу, к примеру список книг к прочтению, список покупок или модуль для напоминания об отдыхе глаз через определенное время (именно этот модуль я и хочу сделать в первую очередь).

Приложение и модули будут иметь графический интерфейс, что требует использования промышленных библиотек для создания одного. В качестве такой библиотеки я использую **Qt (Qt5)**, по ряду причин:

- кроссплатформенность, программа понадобится как под Linux, так и под Windows).
- имеет богатый набор составных частей для создания графического интерфейса.

Каждый модуль может получить нужное дерево задач и выполнять его работу. Система управления задач может получить дерево задач из различных источников. Для реализации источника из файла в формате JSON я использовал библиотеку с открытым исходным кодом JSONCPP (<https://github.com/open-source-parsers/jsoncpp>), чтобы не писать этот разбор самому (обоснование банально :)

Данный проект разделен на следующие подпроекты:

- Исходники самого приложения (**./src**).
- **Interfaces** — интерфейсы основных модулей программы
- **TaskTreeManager** - система управления задачами
- **GUIAgent** - отдельный графический интерфейс адента (в частности реализация на Qt5).
- **PluginManager** - система управления модулями
- **Plugins** — сами модули (в частности модуль **PluginEyesRelax**)
- Документация

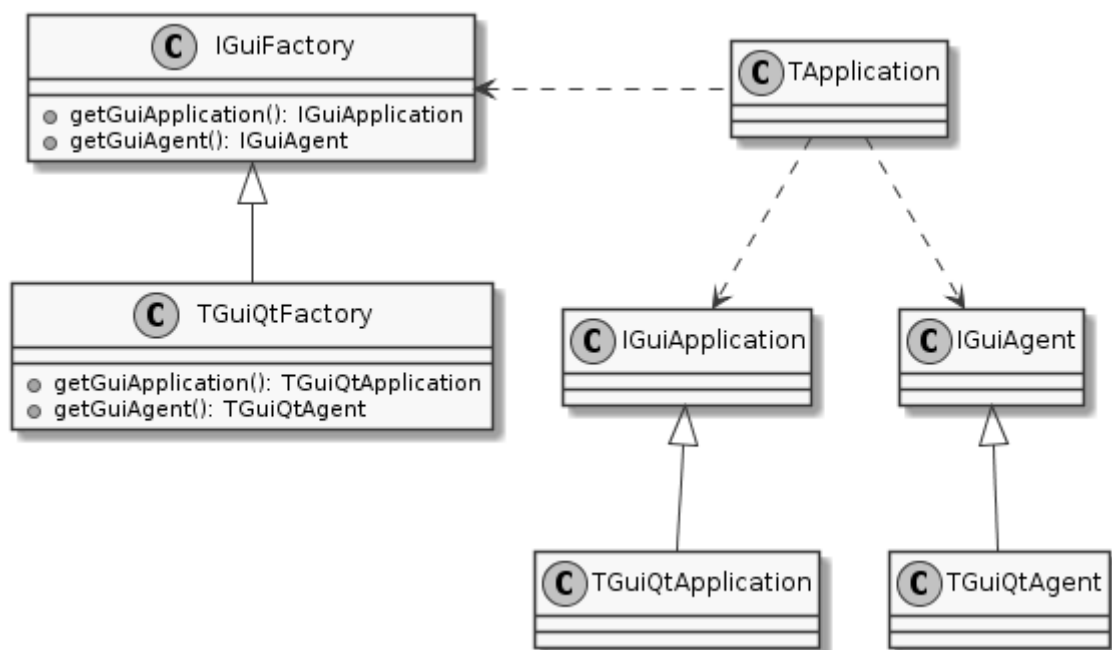
## Применение паттернов проектирования

В данном проекте применялись следующие паттерны:

### 1. Абстрактная фабрика

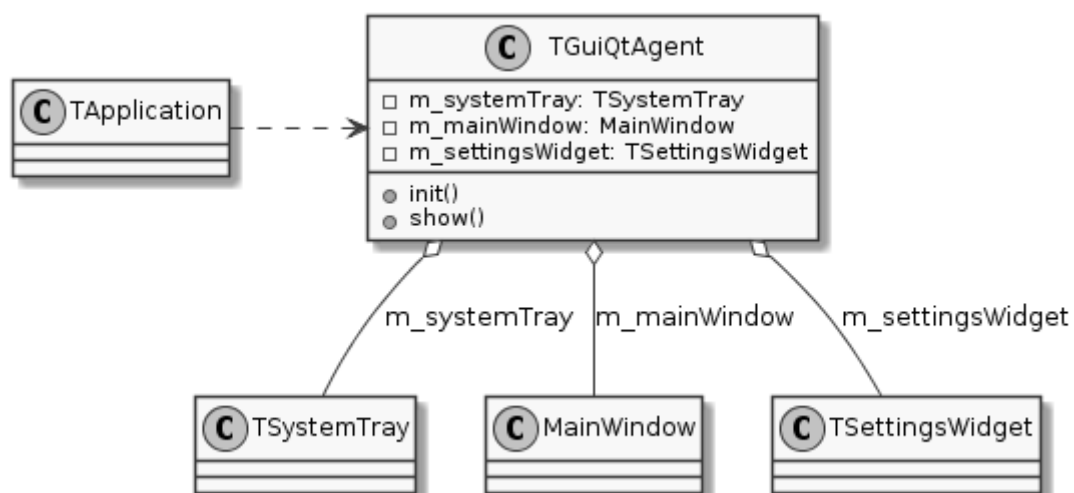
Представлена классами **IGuiFactory** (абстрактная фабрика), **TGuiQtFactory** (конкретная фабрика) и семейством классов-продуктов: **IGuiApplication**, **IGuiAgent** (абстрактные), **TGuiQtApplication**, **TGuiQtAgent** (конкретные).

Данная конкретная фабрика необходима для создания семейства классов, реализующих объекты графического интерфейса библиотеки **Qt**. Они должны использоваться совместно.



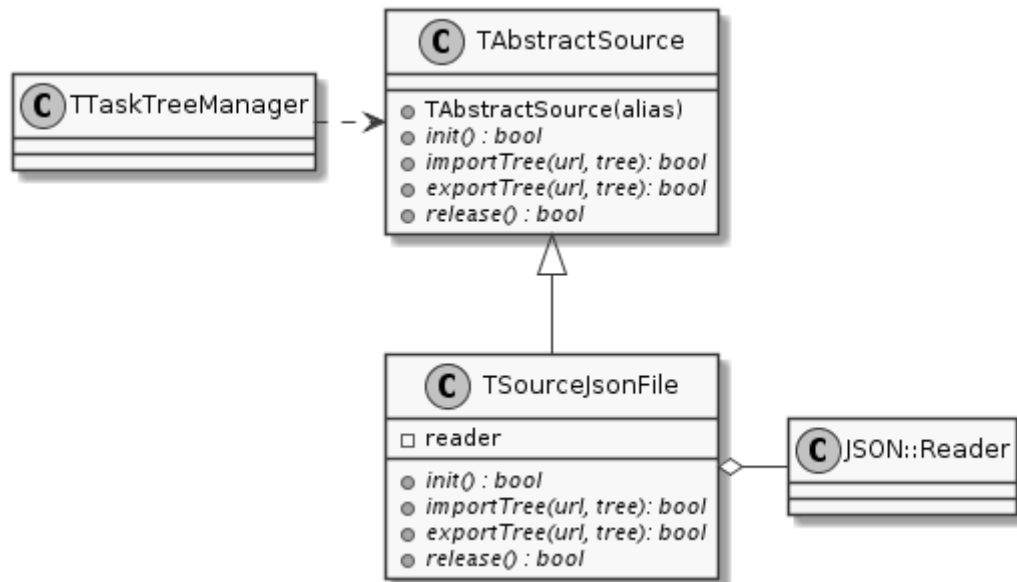
### 2. Фасад

Класс **TGuiQtAgent** в свою очередь выступает в качестве фасада, скрывая и изолируя в себе классы графического интерфейса и предоставляя упрощенный интерфейс для работы с ними.



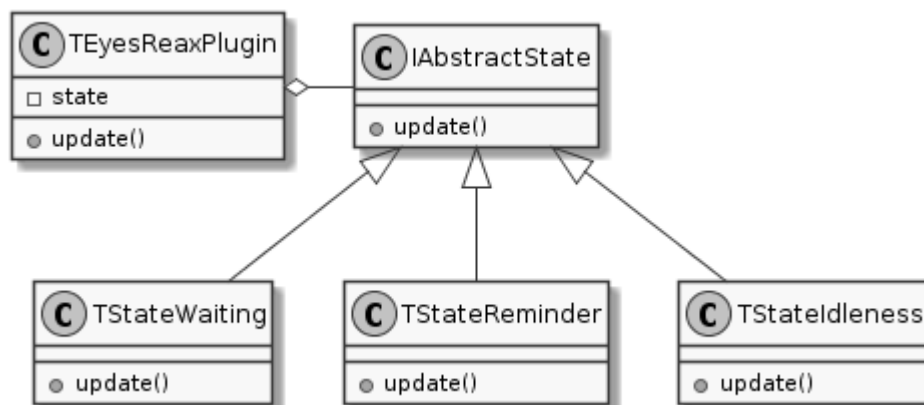
### 3. Адаптер

В качестве адаптера выступает класс **TSourceJsonFile**. Он адаптирует загрузку дерева задач из JSON файла к общему интерфейсу загрузки деревьев задач.



### 4. Состояние

Паттерн состояние используется в плагине для напоминания об отдыхе глаз. Плагин находится в следующих состояниях: в ожидании напоминания, в напоминании и в состоянии бездействия.



### Статический анализ кода

Для статического анализа кода использовал **Cppcheck** (<http://cppcheck.sourceforge.net/>). Нашел много чего интересного буду исправлять и дорабатывать свой проект.

Репозиторий с исходниками: <https://github.com/denproger/organizer>

**P.S.:** Диаграммы создавал с помощью **PlantUml**

**P.S.:** Проект будет еще дорабатываться :)