Маляр Иван Владиславович

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2**

**НА ТЕМУ: «ХИМИЧЕСКАЯ ЛАБОРАТОРИЯ»**

***Ссылка***

<https://github.com/malyariv/labs/tree/master/lab2>

***Реализованная функциональность:***

* стандартный консольный интерфейс ввода/вывода;
* программа генерирует задание как синтез какого-либо химического соединения;
* для выполнения задания предлагается на выбор 5 реагентов;
* также предлагаются две операции: мешать и нагреть;
* есть возможность очистить реактор — место, куда складываются реагенты и протекают реакции, посмотреть его содержимое, получить новое задание или выйти;
* список возможных реакций можно получить из текстового файла или базы данных (тут MySQL), список реагентов из файла, список заданий расположен в enum;

***Описание UML-схемы и реализованных классов:***

***Основные классы***

* The class *Main* is a console application simulating chemical laboratory where user can some operations with chemical reagents.
* The class *UserInteraction* provides interaction between user and application using console interface.
* The class *Reactor*, implementing interface *IObserverable*, is a container for objects implementing interface *IChemicalObserver*. This class is a model of chemical reactor where chemical interaction between chemical reagents occurs.
* The class *Substance*, implementing interface *IChemicalObserver*, is a model of a chemical reagent. It contains methods simulating response of the reagent upon different exposures.

***Вспомогательные классы:***

* The class *TaskGenerator* generates a chemical compound and a list of answers containing chemical reagents.
* The class *ChemicalGenerator* implements the factory method pattern. It generates *SolidsFactory* or *SolutionFactory* depending on number of arguments.
* The class *SolidsFactory* generates *ChemicalObserver* objects using data from file.
* The class *SolutionFactory* generates *ChemicalObserver* objects using data from file.
* The class *ReactionAdapterDB*, implementing interface *IReactionAdapter*, provides receiving information about chemical reaction from MySQL database.
* The class *ReactionAdapterFile*, implementing interface *IReactionAdapter*, provides receiving information about chemical reaction from text file.
* The class *Settings* allows changing of libraries of reagents and reactions using annotations.
* The enumeration *Tasks* is a set of tasks.
* The enumeration *Marker* is a set of chemical classes.
* The class *QuitException* extends *Exception*. Here, it allows to quit the program.

***Интерфейсы***

* The interface *IChemicable* allows receiving information about used chemical reagents
* The interface *IChemicalObserver* extends *IObserver* and *IChemicable* It allows operations with chemical reagents
* The interface *IObserver* allows receiving information from interface *Observerable*
* The interface *IObserverable* allows informing objects from extending interface *IChemicalObserver* and changing some fields
* The interface *IReactionAdapter* allows receiving information about chemical reaction
* The interface *IFactory* is a GoF patterns. Here, it allows generation of chemicals
* The abstract class *AState* is a GoF patterns. Here, it allows different behaviour of chemicals upon different exposures.

***Аннотации***

* The *ChemLib* allows changing of text library of chemical reagents
* The *ReactLibTxt* allows changing of text library of chemical reactions
* The *ReactLibDB* allows changing of database library of chemical reactions.

сама UML-диаграмма в файле UML.pdf

***Использованные шаблоны программирования:***

* Абстрактная фабрика (интерфейс IFactory и его реализации)
* Прототип (копирующий конструктор в классе Substance)
* Адаптер (интерфейс IReactionAdapter и его реализации)
* Состояние (абстрактный класс AState и классы-потомки, явлюящиеся внутренними для класса Substance)
* Стратегия — использование функционального интерфейса Consumer в методе notifyObservers() класса Reactor.
* Наблюдатель — объекты класса Substance получают сообщения от класса Reactor.