Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Вычислительная техника

кафедра

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1, 3**

Преподаватель Васильев В.С.

подпись, дата инициалы, фамилия

Студент КИ17-08Б, 031723830 Кулаев С.Ю.

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

**Задание**

Таблица 1 – Условие задания

|  |  |  |
| --- | --- | --- |
| Обобщенный артефакт, используемый в задании | Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив) | Параметры, общие для всех альтернатив |
| 7. Тексты, состоящие из цифр и латинских букв, зашифрованные различными способами. | Шифрование заменой символов (указатель на массив пар: [текущий символ, замещающий символ]; зашифрованный текст – строка символов)  Шифрование циклическим сдвигом  кода каждого символа на n (целое число, определяющее сдвиг; зашифрованный текст – строка символов) | Незашифрованный  текст – строка символов |

Тип контейнера:

1. Двунаправленный линейный список.

**Зависимости между артефактами и модулями в формате UML**

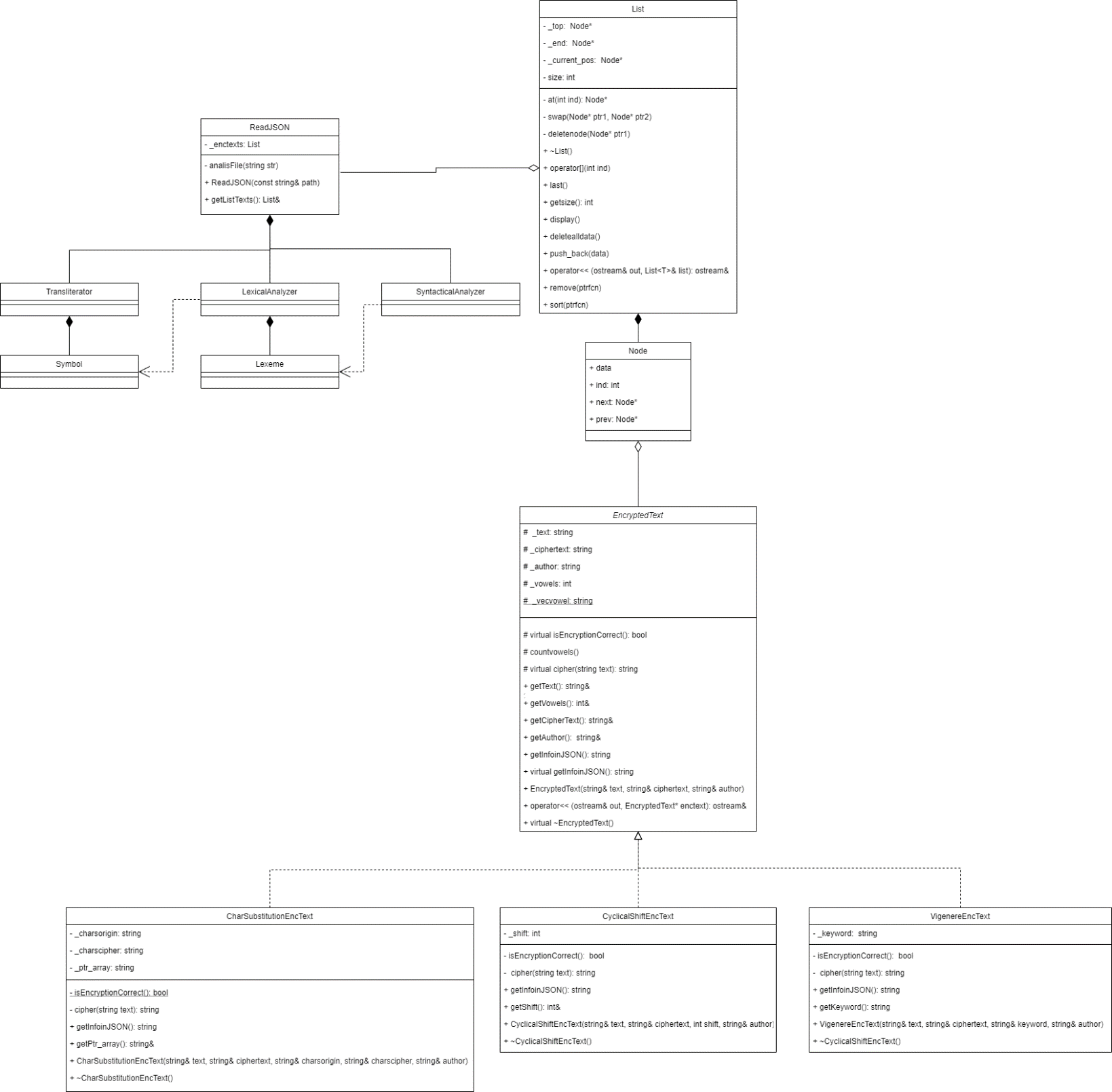
****

Рисунок 1 – Зависимости между артефактами и модулями в формате UML

**Ход работы**

На первом этапе был создан класс транслятора ReadJSON для распознавания JSON и его вспомогательные классы для распознавания:

* Symbol – класс символа;
* Transliterator – класс транcлитератора;
* Lexeme – класс лексемы;
* LexicalAnalyzer – класс лексического анализа;
* Syntactical\_Analyzer – класс синтаксического анализа;

Далее был добавлен абстрактный класс «EncryptedText», который и стал родителем ко всем типам шифрования. В него были добавлены:

* Свойства для исходного текста, зашифрованного текста и геттеры для них;
* Конструктор с параметрами для инициализации исходного и зашифрованного текстов;
* Виртуальный деструктор;
* Виртуальный метод для получения данных объекта в формате json;
* Дружественная перегрузка оператора «<<» для вывода данных в json;
* Виртуальный метод шифрования и метод проверки корректности объекта текста шифрования.

Далее был реализован шаблонный двунаправленный список «List» для хранения текстов и его вложенный класс «Node» для хранения содержимого узла. В классе «Node» был добавлен указатель абстрактного класса «EncryptedText» для хранения текстов разных типов. В «List» были реализованы основные методы, такие как:

* Добавление узла;
* Метод получения узла по индексу;
* Метод получения текста узла по индексу с помощью оператора «[]»;
* Метод для вывода списка на экран, перегрузка оператора «<<» для вывода текстов в файл, а также метод удаления узла;
* Декструктор.

Далее были созданы классы для циклического типа шифрования (CyclicalShiftEncText) и для шифрования заменой символов (CharSubstitutionEncText). В класс «CyclicalShiftEncText» было добавлено свойство сдвига, а в класс «CharSubstitutionEncText» свойство для хранения оригинальных символов и их замены. После этого были добавлены геттеры для этих полей и реализованы все виртуальные функции базового класса.

В класс «ReadJSON» был добавлен контейнер «List<EncryptedText\*> \_enctexts», в который после распознавания были занесены текста.

На втором этапе был добавлен класс «VigenereEncText», унаследованный от «EncryptedText», для типа шифрования Виженера. В него были добавлены поле для ключевого слова и геттер для него, реализованы все виртуальные функции базового класса.

На третьем этапе в базовый класс шифрования было добавлено свойство автора «\_author» и геттер для него. Изменены методы вывода.

На четвертом этапе в базовый класс было добавлено свойство «\_vowels», содержащее количество гласных и реализован метод для их подсчета, вызов которого был добавлен в конструктор.

На пятом этапе в класс «List» был добавлен метод «swap» для перестановки узлов и метод «sort» для сортировки, который в качестве параметра принимает компаратор. Были написаны компараторы для сортировки по убыванию и возрастанию количества гласных.

На шестом этапе в класс «List» был добавлен метод «remove», принимающий в качестве параметра компаратор, для удаления узлов. Были написаны компараторы для удаления текстов в зависимости от типа их шифрования, который определяется с помощью динамического приведения типов.

На каждом этапе разработки проводилась проверка на утечки памяти с помощью «Visual Leak Detector».

**Инструкция по сборке**

Сборка осуществляется с помощью “Meson”.

В папке с исходным кодом в терминале необходимо ввести три команды:

1. meson setup builddir
2. cd builddir
3. meson compile

Для запуска программы из папки builddir введите команду с 2мя дополнительными параметрами(1 – путь к файлу с входными данными, 2 – путь к файлу для вывода).

1. demo.exe <path-input-data> <path-output-data>

**Тестирование**

Для запуска тестов сначала необходимо произвести сборку программы. Из каталога с исходными файлами перейдите в каталог «tests» и запустите на исполнение bat-файл «start\_tests.bat». В каталоге «tests» есть следующие каталоги:

1. Каталог «input\_data» содержит текстовые файлы с входными данными для тестирования. Подробное описание входных данных, а также ожидаемые результаты описаны в файле «Описание.docx». Файл «log\_cmd.txt» содержит весь вывод командной строки после запуска теста;
2. Каталог «expected\_results» содержит ожидаемые результаты выполнения программы;
3. Каталог «actual\_result» содержит файлы с результатами полученными после тестирования с помощью bat-файла;