Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Отчет

По лабораторной работе №6

**«Наследование + полиморфизм»**

по дисциплине «Программирование – 3 семестр»

Студент группы ПИ-32Б Самохвалов Даниил Алексеевич

Преподаватель доцент, Старовойтов Никита Александрович - ассистент

Барнаул 2024

**Задание:**

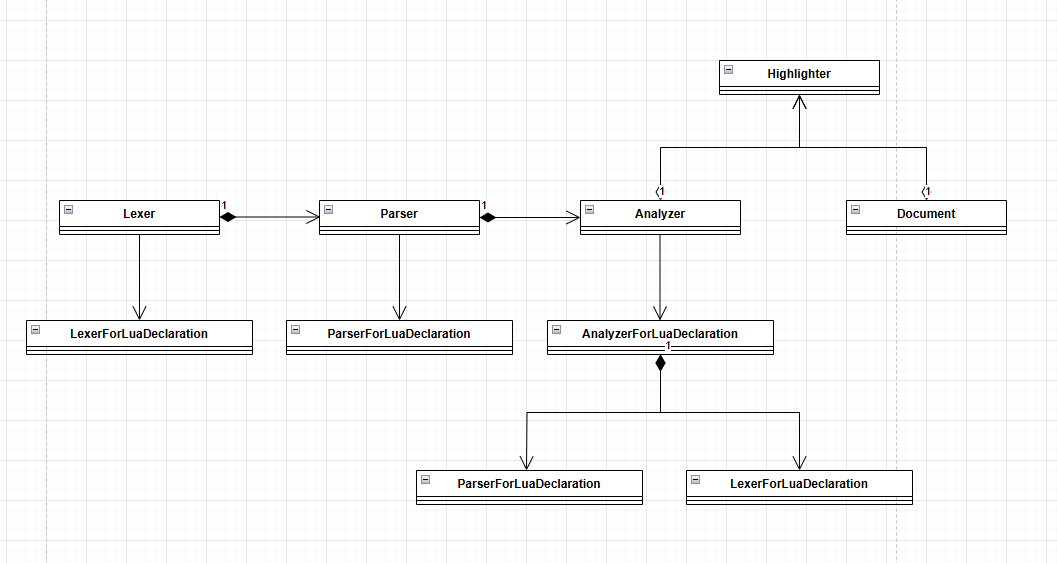
Модифицировать ваши проекты на С++ и Java

1. Придумать для чего в вашем проекте нужен производный класс и создать его (на С++ и Java);
2. Продемонстрировать применение модификатора protected (на С++ и Java);
3. Продемонстрировать перегрузку метода базового класса в производном классе (с вызовом метода базового класса и без такого вызова) (на С++ и Java);
4. Продемонстрировать в конструкторе производного класса с параметрами вызов конструктора базового класса (на С++ и Java);
5. Выполнить перегрузку оператора присваивания объекту производного класса объектов базового класса (С++).
6. Заменить методы Display используя операции << для C++ и метод toString для Java;
7. Придумать разумное использование виртуальных функций и создать их в вашем проекте. Вызов виртуальной функции продемонстрировать через ее вызов не виртуальной функцией базового класса и через динамические объекты базового и производного классов после присваивания указателя. Продемонстрировать изменение работы программы, если функция не виртуальная (С++).
8. В проектах на C++,  Java придумать разумное использование абстрактного класса и создать его. Продемонстрировать его использование;
9. В проектах Java придумать разумное использование интерфейсов. Продемонстрировать их использование;
10. В проектах на Java любой из классов сделать допускающий клонирование. Продемонстрировать варианты с мелким и глубоким клонированием.
11. В программах на C++ и Java придумать разумное использование и реализовать шаблон класса;

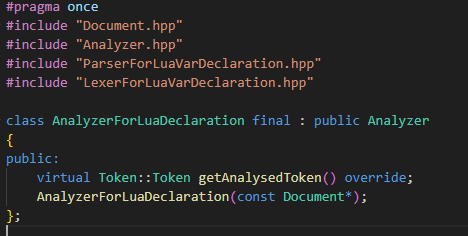
Часть C++

1. Придумать для чего в вашем проекте нужен производный класс и создать его (на С++ и Java);

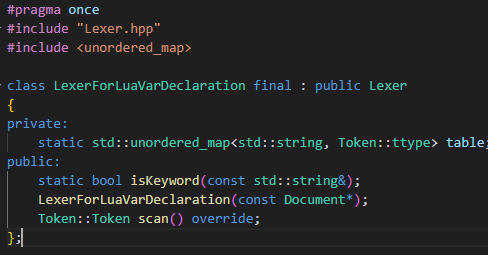
Для наглядности составил UML диаграмму ассоциаций.



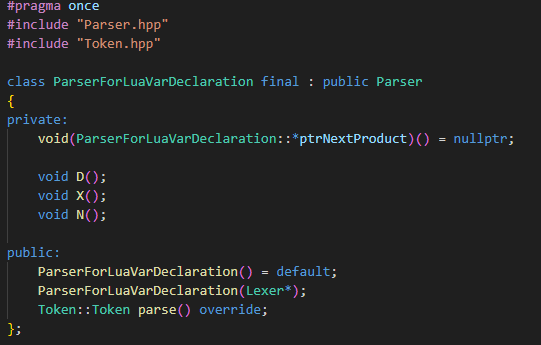
Производный класс AnalyzerForLuaDeclaration



Производный класс LexerForLuaDeclaration

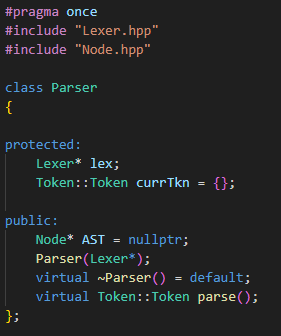


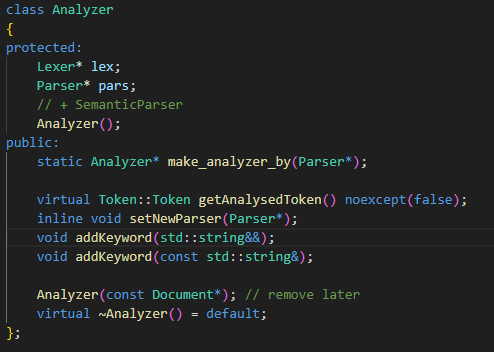
Производный класс ParserForLuaDeclaration

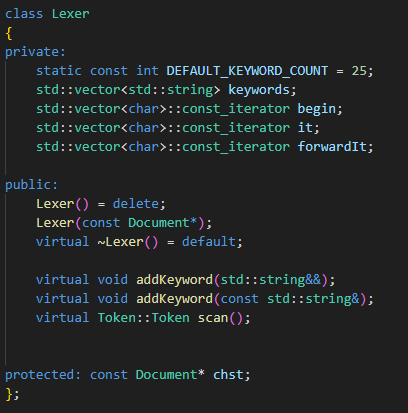


1. Продемонстрировать применение модификатора protected (на С++ и Java);

Использую protected в базовом классе.







*Благодаря protected я получаю доступ к данным в производных классах, но ограничиваю доступ остального вне.*

1. Продемонстрировать перегрузку метода базового класса в производном классе (с вызовом метода базового класса и без такого вызова) (на С++ и Java);

Class Base

Void print(){ cout << 1; }

Class Derived : public Base

Void print(){ cout << 2; }

Int main()

Derived d;

Base\* pb = &d

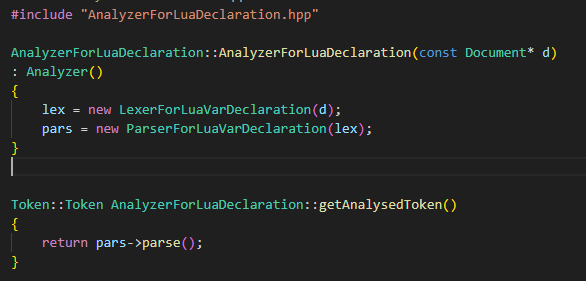
Pb->print() // 1

Derived\* pd = &d

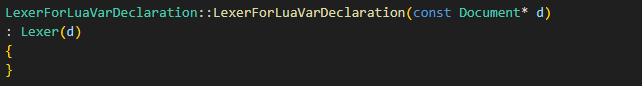
Pd->print() // 2

1. Продемонстрировать в конструкторе производного класса с параметрами вызов конструктора базового класса (на С++ и Java);

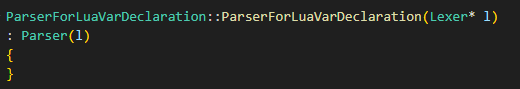
Вызов конструктора базового класса Analyzer в конструкторе производного.



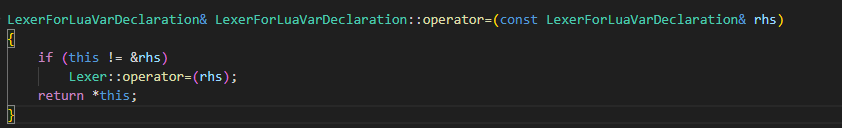
Вызов конструктора базового класса Lexer в конструкторе производного.



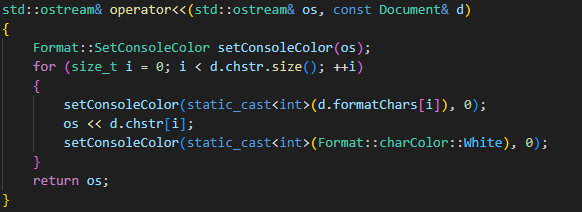
Вызов конструктора базового класса Parser в конструкторе производного.



1. Выполнить перегрузку оператора присваивания объекту производного класса объектов базового класса (С++).

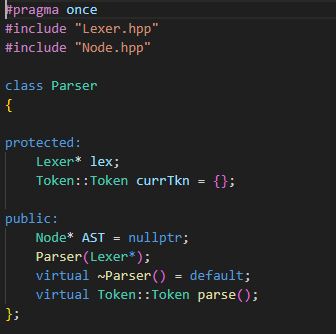


1. Заменить методы Display используя операции << для C++ и метод toString для Java;

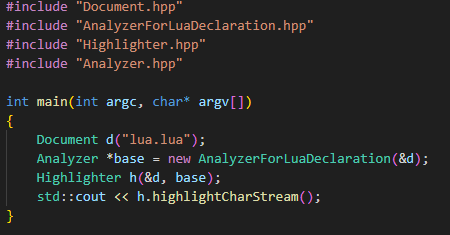


1. Придумать разумное использование виртуальных функций и создать их в вашем проекте. Вызов виртуальной функции продемонстрировать через ее вызов невиртуальной функцией базового класса и через динамические объекты базового и производного классов после присваивания указателя. Продемонстрировать изменение работы программы, если функция не виртуальная (С++).

Виртуальная функция в базовом классе Parser.



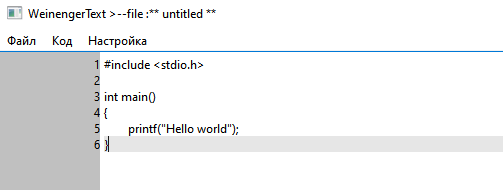
Динамическое связывание.



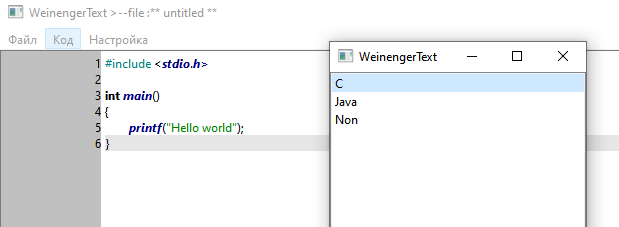
Примечание: Это позволяет удобно менять алгоритм парсинга. Меняя только ту часть, которая относится к парсингу, а не ту, в которой вызывается функция для парсинга.

Практический пример.

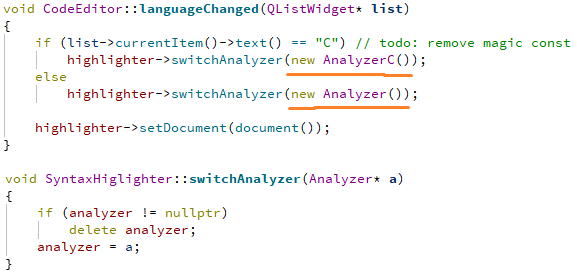
Ничего не подкрашивается.



В меню выбираем другой язык и подсветка появилась



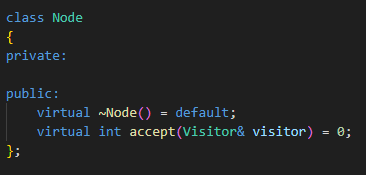
Реализуется это с помощью полиморфизма



*Для реализации анализатора для Java нужно создать новый класс, кардинально менять старые не надо. (в практическом примере достаточно добавить еще одно условие).*

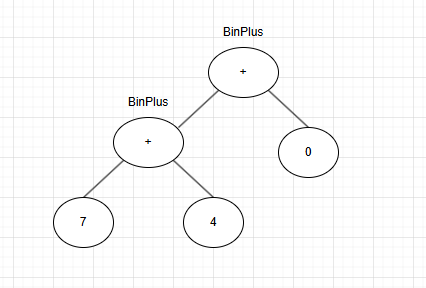
* 1. В проектах на C++,  Java придумать разумное использование абстрактного класса и создать его.

Абстрактный класс.



* 1. Продемонстрировать его использование;

К примеру, составим AST (Abstract Syntax Tree) для выражения 7+4+0.



*По блок схеме видно, что у узла* ***BinPlus*** *пара дочерних узлов, также видно, что они разных типов. Таким образом, чтобы программно реализовать возможность построения дерева наподобие этого необходим абстрактный класс* ***Node****, и реализующие его* ***BinPlus*** *и* ***IntValue****.*

Код демонстрации использования абстрактного класса:

[University\_projects/lab6\_7/cpp\_solution/NodeForLab6.cpp at main · eastpriSs/University\_projects](https://github.com/eastpriSs/University_projects/blob/main/lab6_7/cpp_solution/NodeForLab6.cpp) Вывод:

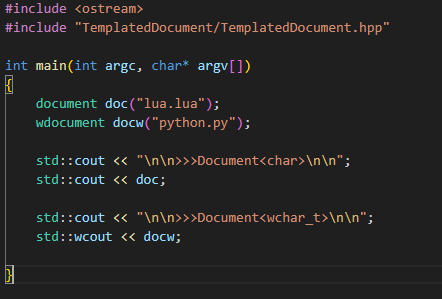


11) В программах на C++ и Java придумать разумное использование и реализовать шаблон класса;

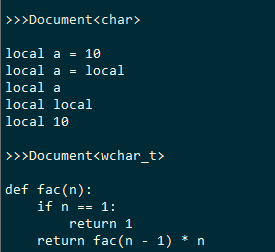
Я решил шаблонизировать класс Document.

Код: [University\_projects/lab6\_7/cpp\_solution/TemplatedDocument at main · eastpriSs/University\_projects](https://github.com/eastpriSs/University_projects/tree/main/lab6_7/cpp_solution/TemplatedDocument)

Использование:

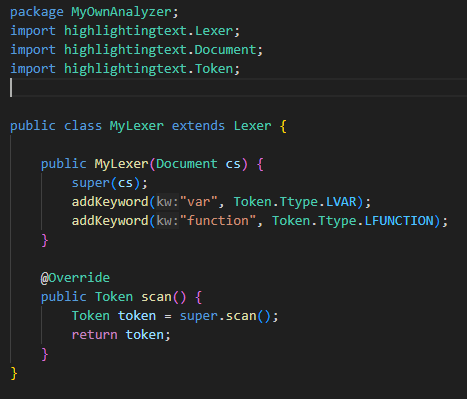


Вывод:

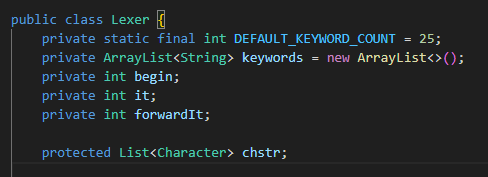


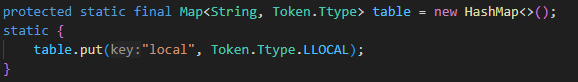
Часть Java

1. Придумать для чего в вашем проекте нужен производный класс и создать его (на С++ и Java);

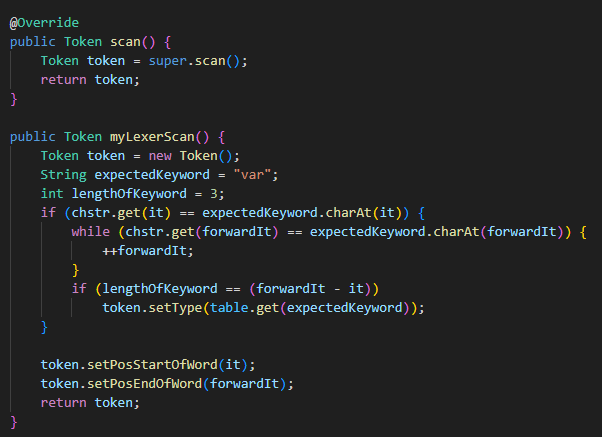


1. Продемонстрировать применение модификатора protected (на С++ и Java);

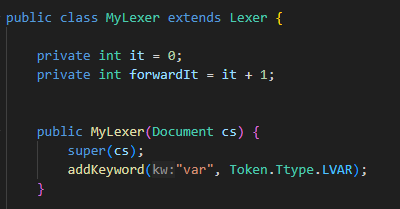




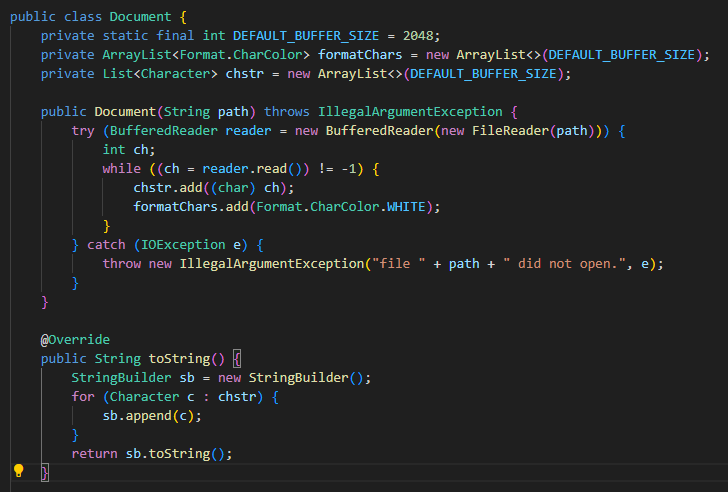
1. Продемонстрировать перегрузку метода базового класса в производном классе (с вызовом метода базового класса и без такого вызова) (на С++ и Java);



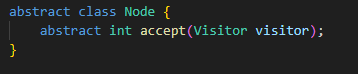
1. Продемонстрировать в конструкторе производного класса с параметрами вызов конструктора базового класса (на С++ и Java);

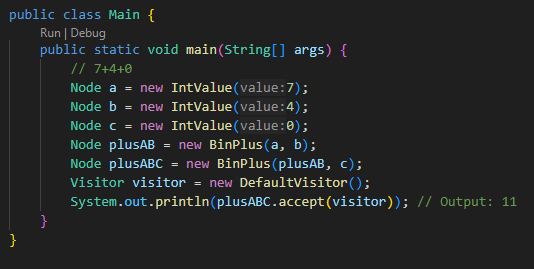


1. Заменить методы Display используя операции << для C++ и метод toString для Java;

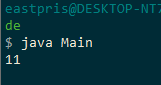


1. В проектах на C++,  Java придумать разумное использование абстрактного класса и создать его. Продемонстрировать его использование;

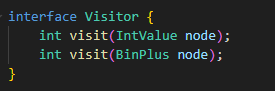




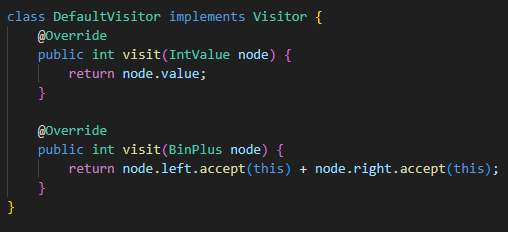
Вывод:



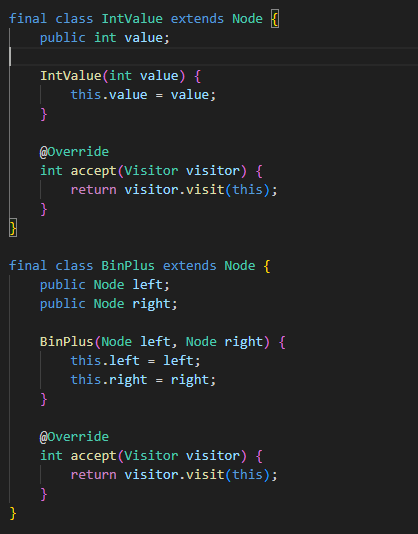
1. В проектах Java придумать разумное использование интерфейсов. Продемонстрировать их использование;



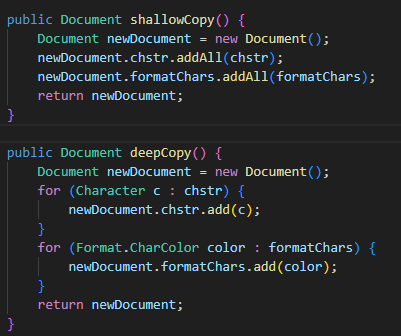
Реализация:



Использование:



1. В проектах на Java любой из классов сделать допускающий клонирование. Продемонстрировать варианты с мелким и глубоким клонированием.

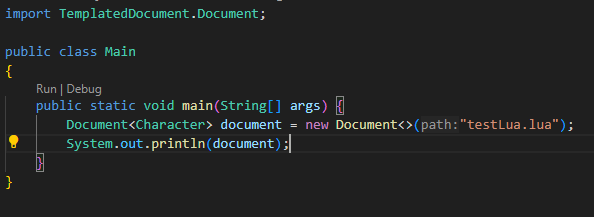


1. В программах на C++ и Java придумать разумное использование и реализовать шаблон класса;

Код:

[University\_projects/lab6\_7/java\_solution/TemplatedDocument at main · eastpriSs/University\_projects](https://github.com/eastpriSs/University_projects/tree/main/lab6_7/java_solution/TemplatedDocument)

Использование:



Вывод:

