1. Разработка грамматики.

Разработаем контекстно-свободную грамматику для языка регулярных выражений с операциями конкатенации (простая последовательная запись строк), выбора (вертикальная черта), замыкания Клини. Приоритет операций стандартный. Скобки могут использоваться для изменения приоритета. Кроме операторов и скобок допускаются также только маленькие буквы латинского алфавита.

Построим грамматику:

R –> R R

R –> R | R

R –> R *

R –> (R)

R –> c

где **R** – регулярное выражение

В приведенной грамматике присутствует левая рекурсия и правое ветвление. Преобразуем грамматику, чтобы избавиться от левой рекурсии:

R –> Concat R

R –> Concat

Concat –> Or | Concat

Concat –> Or

Or –> Closure *

Or –> Closure

Closure –> (R)

Closure –> c

Нетерминал	Описание

R	Регулярное выражение
Concat	Конкатенация
Or	Операция выбора
Closure	Замыкание Клини

Теперь в грамматике отсутствует левая рекурсия, но все еще осталось правое ветвление. Избавимся от него, использую левую факторизацию:

R –> Concat R’

R’ –> R

R’ –> ε

Concat –> Or Concat’

Concat’ –> | Concat

Concat’ –> ε

Or –> Closure Or’

Or’ –> *

Or’ –> ε

Closure –> (R)

Closure –> c

Нетерминал	**Описание**
R’	Продолжение регулярного выражения
Concat’	Продолжение конкатенации
Or’	Продолжение операции выбора

Теперь упростим грамматику, раскрыв нетерминалы **R** и **Concat** в правилах для **R’** и **Concat’**:

R –> Concat R’

R’ –> Concat R’

R’ –> ε

Concat –> Or Concat’

Concat’ –> | Or Concat’

Concat’ –> ε

Or –> Closure Or’

Or’ –> *

Or’ –> ε

Closure –> (R)

Closure –> c

Получили итоговую LL(1) грамматику.

2. Построение лексического анализатора.

Приведем список терминалов нашей грамматики:

Терминал	Токен	Описание
|	**OR**	Оператор выбора
*****	**CLOSURE**	Оператор замыкания Клини
(	**LBRACKET**	Открывающая скобка
)	**RBRACKET**	Закрывающая скобка
c	**CHAR**	Маленькая латинская буква
$	**END**	Конец выражения

Класс лексического анализатора расположен в файле **LexicalAnalyzer.java**

3. Построение синтаксического анализатора.

Для построения синтаксического анализатора построим множества **FIRST** и **FOLLOW** для нетерминалов нашей грамматики.

Нетерминал	FIRST	FOLLOW
R	(, c	$,)

R’	ε, (, c	$,)
Concat	(, c	$, (, c,)
Concat’	|, ε	$, (, c,)
Or	(, c	|, $, (, c,)
Or’	*, ε	|, $, (, c,)
Closure	(, c	*, |, $, (, c,)

Класс для хранения дерева разбора находится в файле **Tree.java**

Класс синтаксического анализатора находится в файле **Parser.java**

4. Визуализация дерева разбора.

Для визуализации дерева разбора воспользуемся языком описания графов **DOT** и утилитой **dot**. Класс **TreeVisualiser**, расположенный в файле **TreeVisualiser.java**, получает дерево разбора (объект класса **Tree**), сгенерированное классом **Parser** и добавляет в файл **graph.gv** описание каждого ребра из этого дерева. Затем **TreeVisualiser** вызывает команду построения **.png** изображения по полученному файлу **graph.gv**. Итоговое изображение дерева разбора находится в файле **graph<i>.png**. **<i>** здесь означает номер изображения, т.к. тестирование производится на наборе из множества тестов.

5. Подготовка набора тестов.

Проверим правильность работы нашего синтаксического анализатора с помощью следующего набора тестов.

Тесты с верными регулярными выражениями:

Регулярное выражение	Описание
abcde	Выражение, состоящее только из конкатенаций
(((a)))(b)(c)((d))	Выражение с множественными скобками и конкатенациями
a*b*c*	Выражение с замыканиями Клини

(ab)*c*(abcd(efg*)*)*	Выражение замыканиями Клини внутри и снаружи скобок
(a|b)	Выражение с оператором выбора
(a|b|c(a|b|c)(ab|c|d))	Выражение с операторами выбора во вложенных скобках
a b c (a|b)* (z |x|v*)	Выражение с множественными пробелами
(((a*)(b*|c)*)*)*	Выражение с замыканиями Клини от замыканий Клини
(a(b*)|((n*|m)*)(p)ab)*	Выражение со всеми операторами
((((((a(b|c*))))))(abc)))	Выражение с глубокой вложенностью

Тесты с неправильными регулярными выражениями:

Регулярное выражение	**Описание**
a1bcde	Недопустимый цифровой символ
(abc	Незакрытая скобка
((a*b|abc)))	Лишняя закрывающая скобка
((((|))))(((*)*))()()(()*)	Отсутствие букв
(|b)	Пустой операнд оператора выбора
(a(b*|(cd)*|(a|b))')	Недопустимый символ: ’
((((a|b|c|d|))))	Пустой операнд оператора выбора в серии операторов выбора
ab*Cde	Недопустимый символ: заглавная буква
abcйdef	Недопустимый символ: й
((ab*(a))))))))))	Лишняя закрывающая скобка
(((((((abc)	Незакрытая скобка

Класс **Main** запускает все приведенные тесты. Для корректных регулярных выражений генерируется изображение дерева разбора и помещается в папку **trees**. Для некорректных регулярных выражений в файл **ErrorsReport** записываются описания ошибок в этих выражениях.