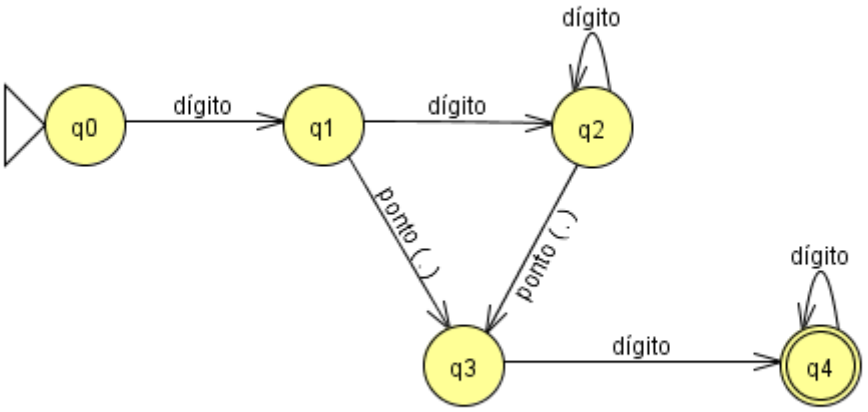
Integrantes: Maria Eduarda Wendel Maia, Marina Yamaguti, Paola Anacleto Lopes

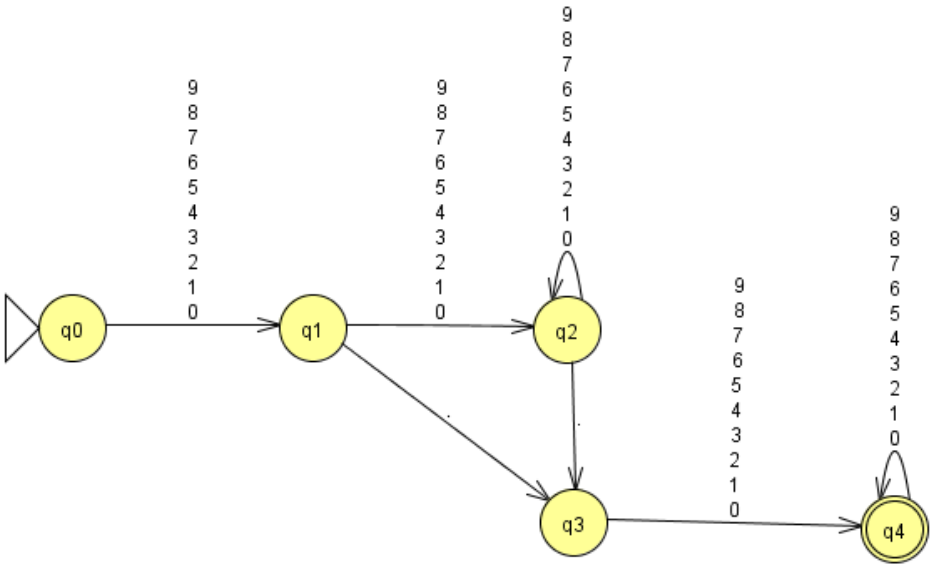
T1 - Linguagens de Programação

Parte 1 – Exercícios realizados na aula anterior (Projetamos diagramas de transições de estado que descreviam os padrões de tokens da linguagem e escrevemos um programa implementando os diagramas)

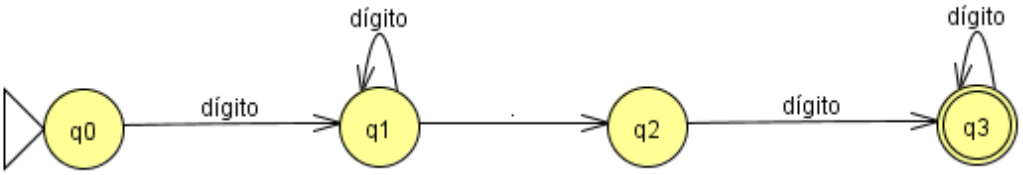
a) Tabela de transição de estados (sem alterações):



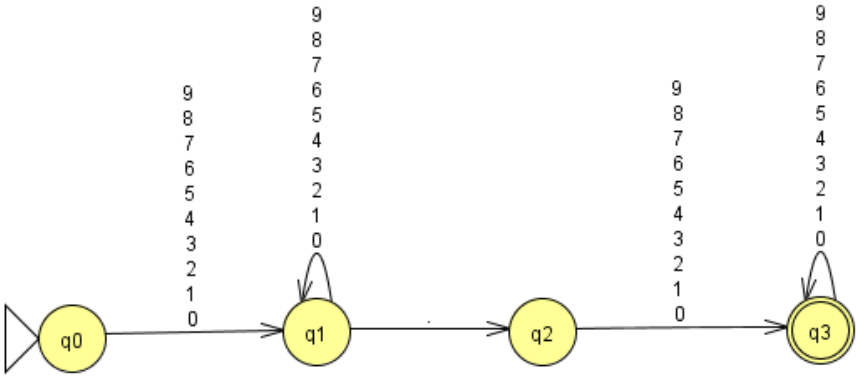
Com dígitos e pontos:



Com alterações:

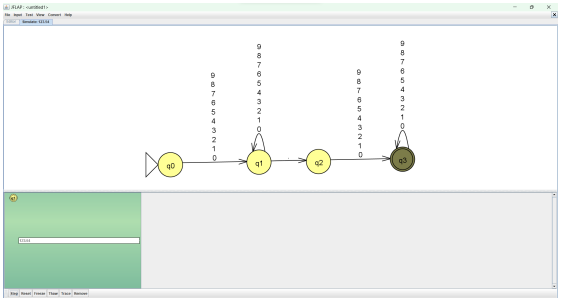


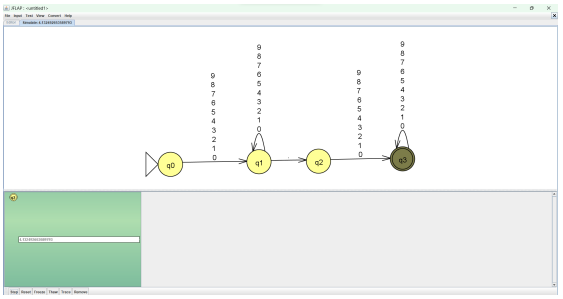
Com dígitos e pontos:



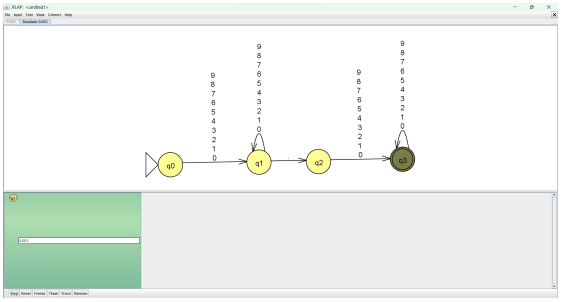
Testes:

123.54





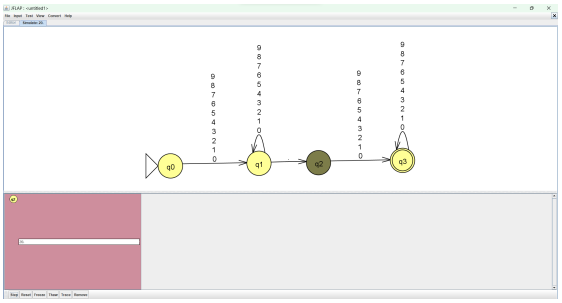
0.003



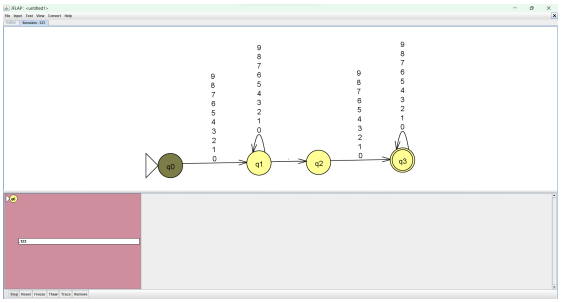
4.132492653589793

Testes que deram erro (pois precisa ser um double para funcionar):

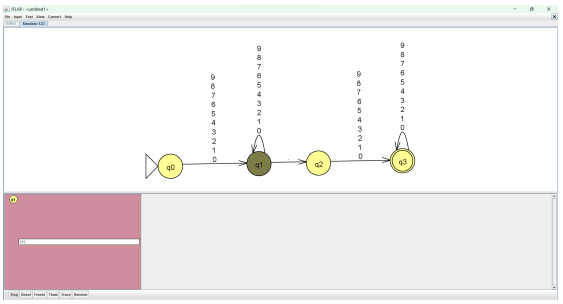
20.



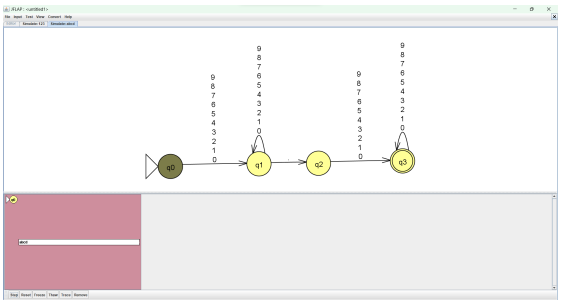
.123



123



abcd



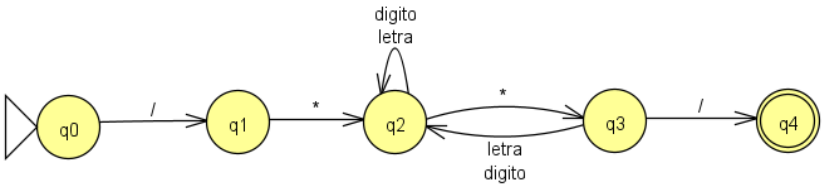
b) A modificação feita no código foi colocar 3 ao invés de 4 na linha 34, pois o nosso diagrama de estados otimizado possui 3 estados

```
28      private static boolean reconhecerLiteralPontoFlutuante(String entrada) {
29          estadoAtual = 0;
30          for (int i = 0; i < entrada.length(); i++) {
31              char caractere = entrada.charAt(i);
32              transicao(caractere);
33          }
34          return estadoAtual == 3; // Estado FINAL
35      }
```

- Resultado no console:

```
Testando: 123.54
Literal de ponto flutuante valido: 123.54
Testando: 0.003
Literal de ponto flutuante valido: 0.003
Testando: 4.132492653589793
Literal de ponto flutuante valido: 4.132492653589793
Testando: 20.
Literal de ponto flutuante invalido: 20.
Testando: .123
Literal de ponto flutuante invalido: .123
Testando: 123
Literal de ponto flutuante invalido: 123
Testando: abcd
Literal de ponto flutuante invalido: abcd
```

2- a) diagrama de estados no JFlap:



- Mudanças no código e anterior para conseguir realizar a leitura do comentário e verificar se é válido e explicação do código:

- ao invés do reconhecerPontoLiteralFlutuante colocamos reconhecerComentario

```java
    public static void main(String[] args) {

        try {
            in_fp = new BufferedReader(new FileReader(fileName:"entrada.txt"));
            String line;
            while ((line = in_fp.readLine()) != null) {
                System.out.println("Testando: " + line);
                if (reconhecerComentario(line)) {
                    System.out.println("Comentario valido: " + line);
                } else {
                    System.out.println("Comentario invalido: " + line);
                }
            }
        } catch (IOException e) {
            System.out.println(x:"Erro ao ler o arquivo.");
        }
    }
```

- Modificamos o estadoAtual para 4 pois o novo diagrama de estados possui 4 estados

```
private static boolean reconhecerComentario(String entrada) {
    estadoAtual = 0;
    for (int i = 0; i < entrada.length(); i++) {
        char caractere = entrada.charAt(i);
        transicao(caractere);
    }
    return estadoAtual == 4; // Estado FINAL
}
```

- Case 0: Este é o estado inicial. Se o caractere atual for uma barra "/", a máquina de estados muda para o estado 1 (caso contrário, o estado atual se torna -1, indicando um estado inválido).

```
switch (estadoAtual) {
    case 0:
        if (caractere == '/') {
            estadoAtual = 1;
        } else {
            estadoAtual = -1;
        }
        break;
```

- Case 1: Neste estado, a máquina de estados espera um asterisco, mudando para o estado 2 (caso contrário, o estado atual se torna -1).

```
case 1:
    if (caractere == '*') {
        estadoAtual = 2;

    } else {
        estadoAtual = -1; // Estado inválido
    }
    break;
```

- case 2: Serve para reconhecer os dígitos e caracteres e caso não seja dizer que o estado é inválido

```
case 2:
    if (Character.isLetterOrDigit(caractere) || caractere == ' ' || caractere == '.'|| caractere == '-' || caractere == '_'
        estadoAtual = 2;
    }
    else if(caractere == '*') {
        estadoAtual = 3;
    }
    else {
        estadoAtual = -1; // Estado inválido
    }
    break;
```

- Case 3: A primeira parte do bloco verifica se o caractere atual (armazenado na variável caractere) é uma letra, um dígito ou um dos caracteres especiais listados e se for algum desses caracteres, o estadoAtual é definido como 2. A segunda parte do bloco verifica se o caractere é uma barra ("/"). Se for, o estadoAtual é definido como 4. Se nenhuma das condições anteriores for atendida, o estadoAtual é definido como -1, indicando um estado inválido.

```
case 3:
    if (Character.isLetterOrDigit(caractere) || caractere == ' ' || caractere == '.'|| caractere == '-' || caractere == '_' |
        estadoAtual = 2;
    }
    else if (caractere == '/') {
        estadoAtual = 4;
    }
    else {
        estadoAtual = -1; // Estado inválido
    }
    break;
```

- Case 4: O código dentro deste bloco será executado se o valor da variável caractere for avaliado como 4. Aqui, o código verifica se o caractere atual é um dígito. Se for, o estadoAtual é definido como -1, o que representa comentário inválido

```
case 4:
if (Character.isDigit(caractere)) {
    estadoAtual = -1;
}
break;
```

- Resultado da implementação da forma de comentários de java:

```
Testando: /*123.54*/
Comentario valido: /*123.54*/
Testando: /*-*/
Comentario valido: /*-*/
Testando: /* abcd */
Comentario valido: /* abcd */
Testando: /* */
Comentario valido: /* */
Testando: /*__aaxx*/
Comentario valido: /*__aaxx*/
Testando: /*,=,%,@,#,(),&,*/
Comentario valido: /*,=,%,@,#,(),&,*/
Testando: jubs
Comentario invalido: jubs
Testando: 174
Comentario invalido: 174
```

Parte 2 – Gerador de Analisador Léxico com Flex – utilizando os mesmos exercícios da parte 1 (Escrevemos uma descrição formal dos padrões de token da linguagem, usando uma linguagem descritiva relacionada a expressões regulares. Descrições que serão usadas como entrada para uma ferramenta de software que gera um analisador léxico automaticamente (lex))

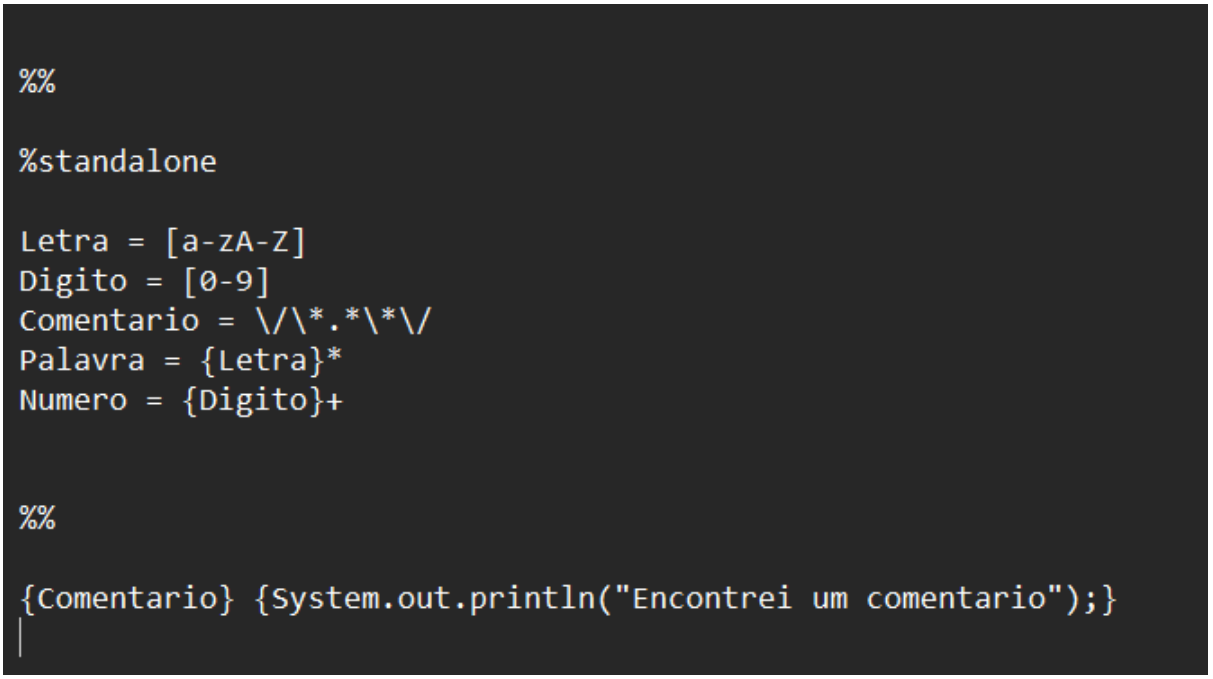
```
%%

%standalone

Letra = [a-zA-Z]
Digito = [0-9]
Comentario = \/\*.*\*\/
Palavra = {Letra}*
Numero = {Digito}+


%%

{Comentario} {System.out.println("Encontrei um comentario");}
```



arquivo de entrada:

```
/*123.54*/
/*-*/
/* abcd */
/* */
/*__aaxx*/
/*,=,%,@,#,(),&,*/
jubs
174
```

Resultado:

```
C:\Users\Duda Maia\Downloads>java Yylex entrada.txt
Encontrei um comentario

Encontrei um comentario

Encontrei um comentario

Encontrei um comentario

Encontrei um comentario

Encontrei um comentario

jubs
174
C:\Users\Duda Maia\Downloads>
```