

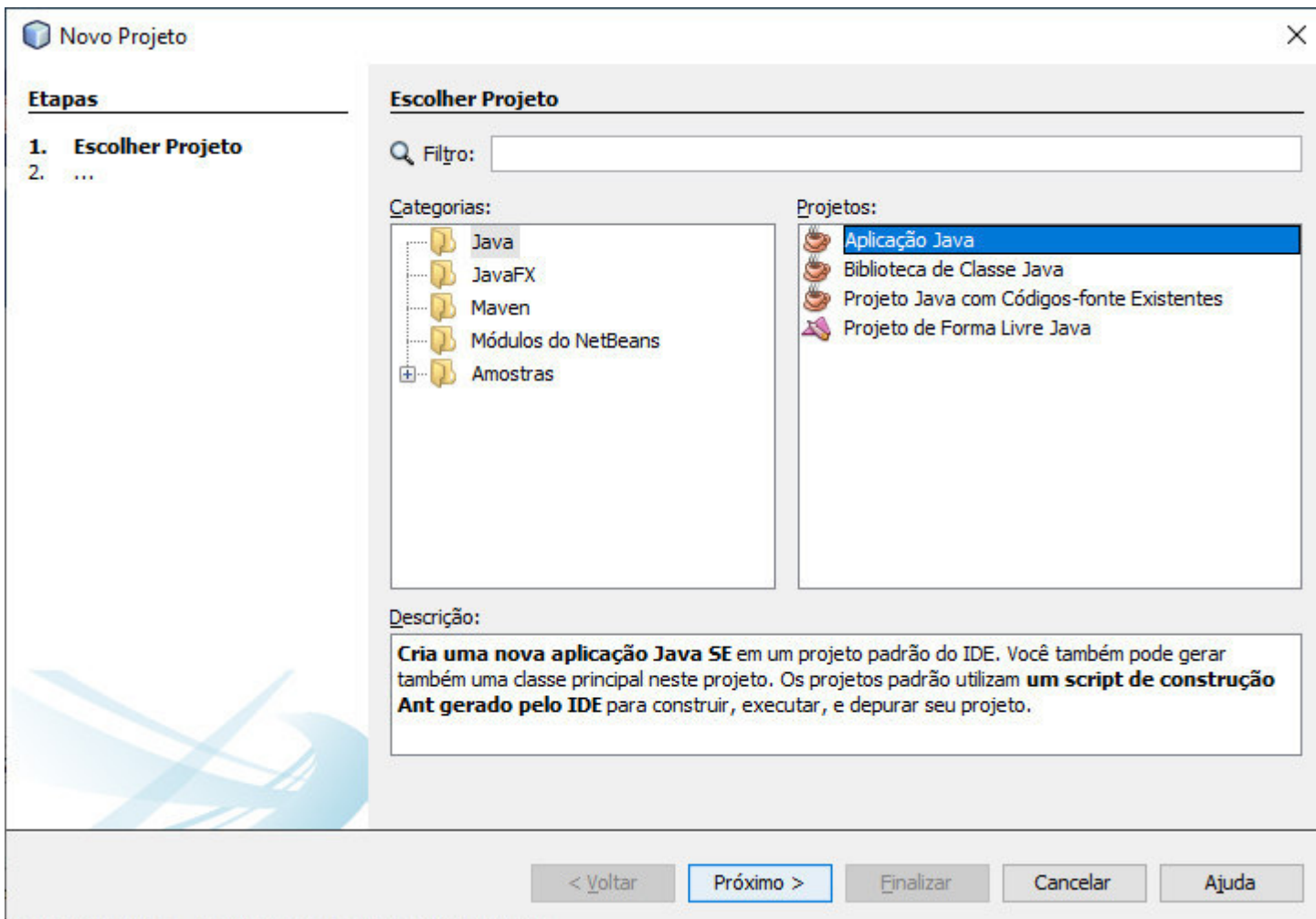
Desenvolvimento do analisador léxico

1. Criar novo projeto;
2. Associar JFlex na biblioteca;
3. No projeto criar pacote “Compilador”;
4. Criar arquivo de especificação do Flex;
 - a) Compreender e implementar as especificações;
 - b) Realizar melhorias para refinar a análise dos Tokens e Lexemas;

Novo projeto

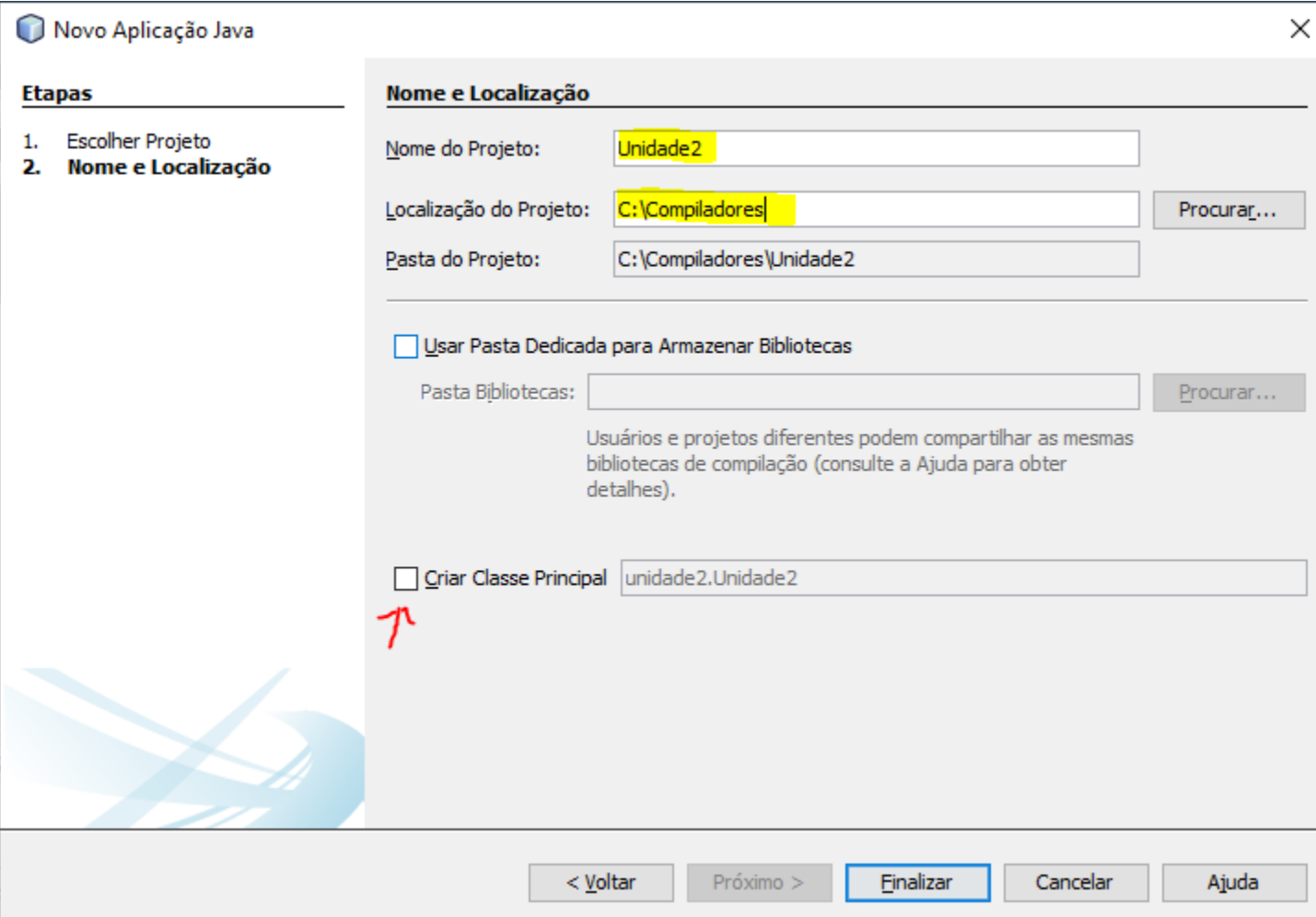
Criar um novo projeto

- Novo Projeto > Aplicação Java



Criar um novo projeto

- Nome: Unidade2
- Localização: C:\Compiladores
- Desmarque a criação da classe principal



Novo Aplicação Java

Etapas

1. Escolher Projeto
2. **Nome e Localização**

Nome e Localização

Nome do Projeto:

Localização do Projeto: Procurar...

Pasta do Projeto:

☐ Usar Pasta Dedicada para Armazenar Bibliotecas

Pasta Bibliotecas: Procurar...

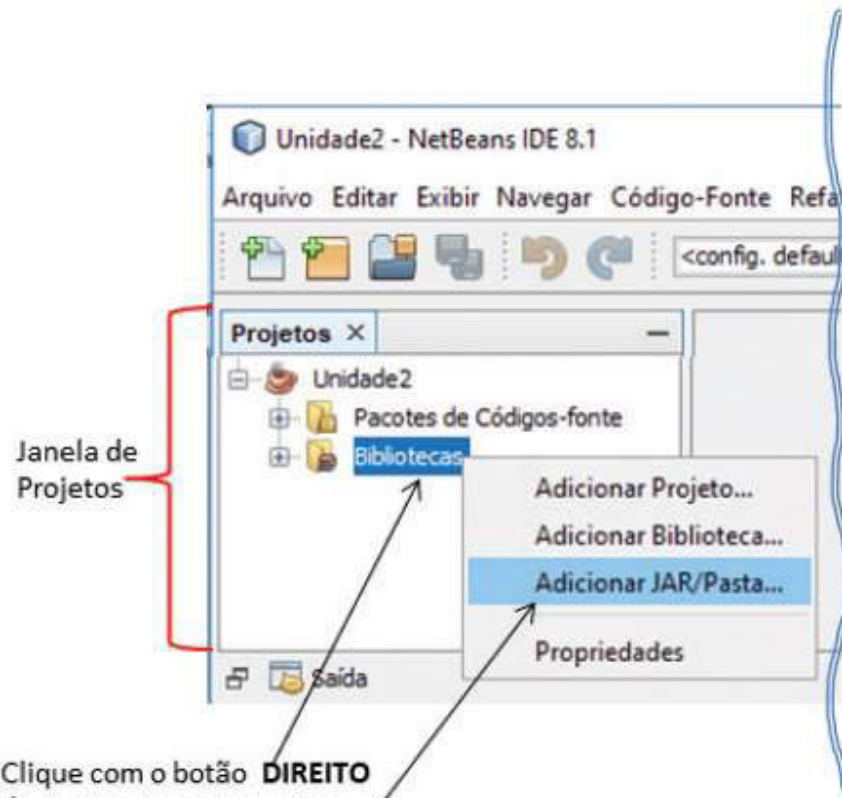
Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).

☐ Criar Classe Principal

< Voltar Próximo > Finalizar Cancelar Ajuda

Adicionar JFlex ao novo projeto

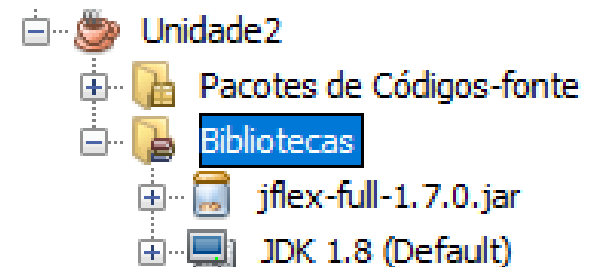
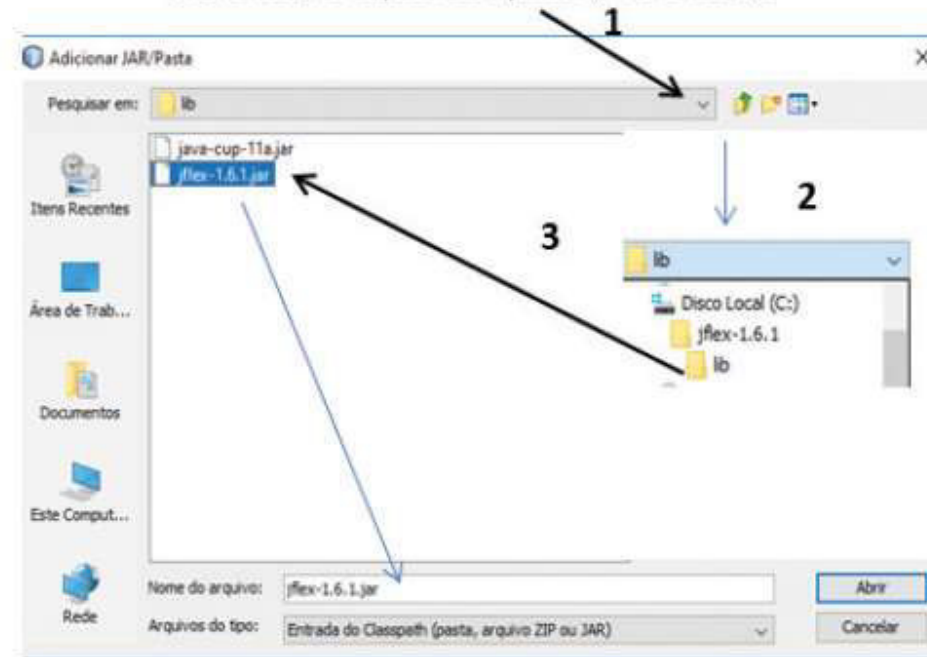
A



Clique com o botão **DIREITO** do mouse para aparecer o menu suspenso
Indicado e escolha a opção **Adicionar JAR/Pasta...**

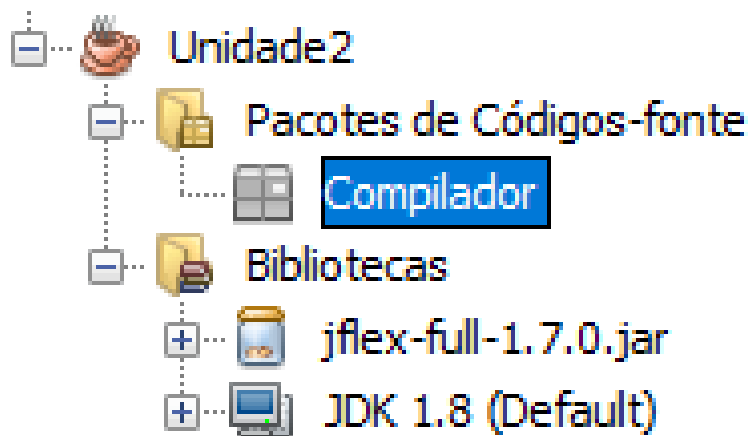
B

Selecione o local onde descompactou a ferramenta Jflex 1.6.1, e depois abra pasta \Jflex 1.6.1\lib



Criar um pacote “Compilador”

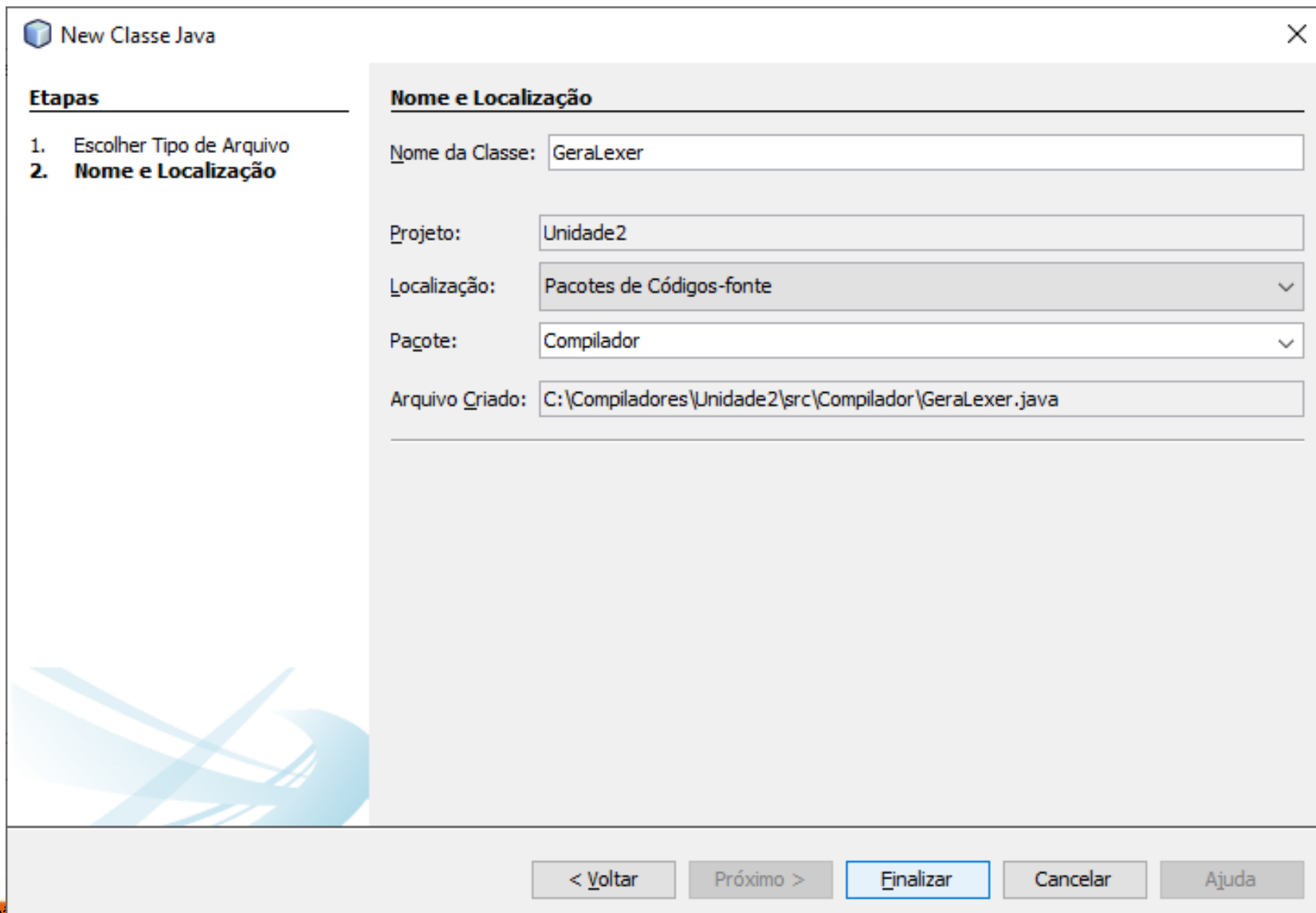
- Botão direito em Unidade2, Novo > Pacote Java



Criar a classe GeraLexer

Criar a Classe GeraLexer

- No pacote Compilador criar Novo > Classe Java...



New Classe Java

Etapas

- Escolher Tipo de Arquivo
- Nome e Localização**

Nome e Localização

Nome da Classe:

Projeto:

Localização:

Pacote:

Arquivo Criado:

< Voltar Próximo > Finalizar Cancelar Ajuda

Criar a classe GeraLexer

```
package Compilador;
```

```
import java.io.*;
```

```
public class GeraLexer {
```

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

```
        String arq = "C:/Compiladores/Unidade2/src/Compilador/especificacao.flex";
```

```
        gerarLexer(arq);
```

```
    }
```

```
        public static void gerarLexer ( String arq) {
```

```
            File file = new File(arq);
```

```
            jflex.Main.generate(file);
```

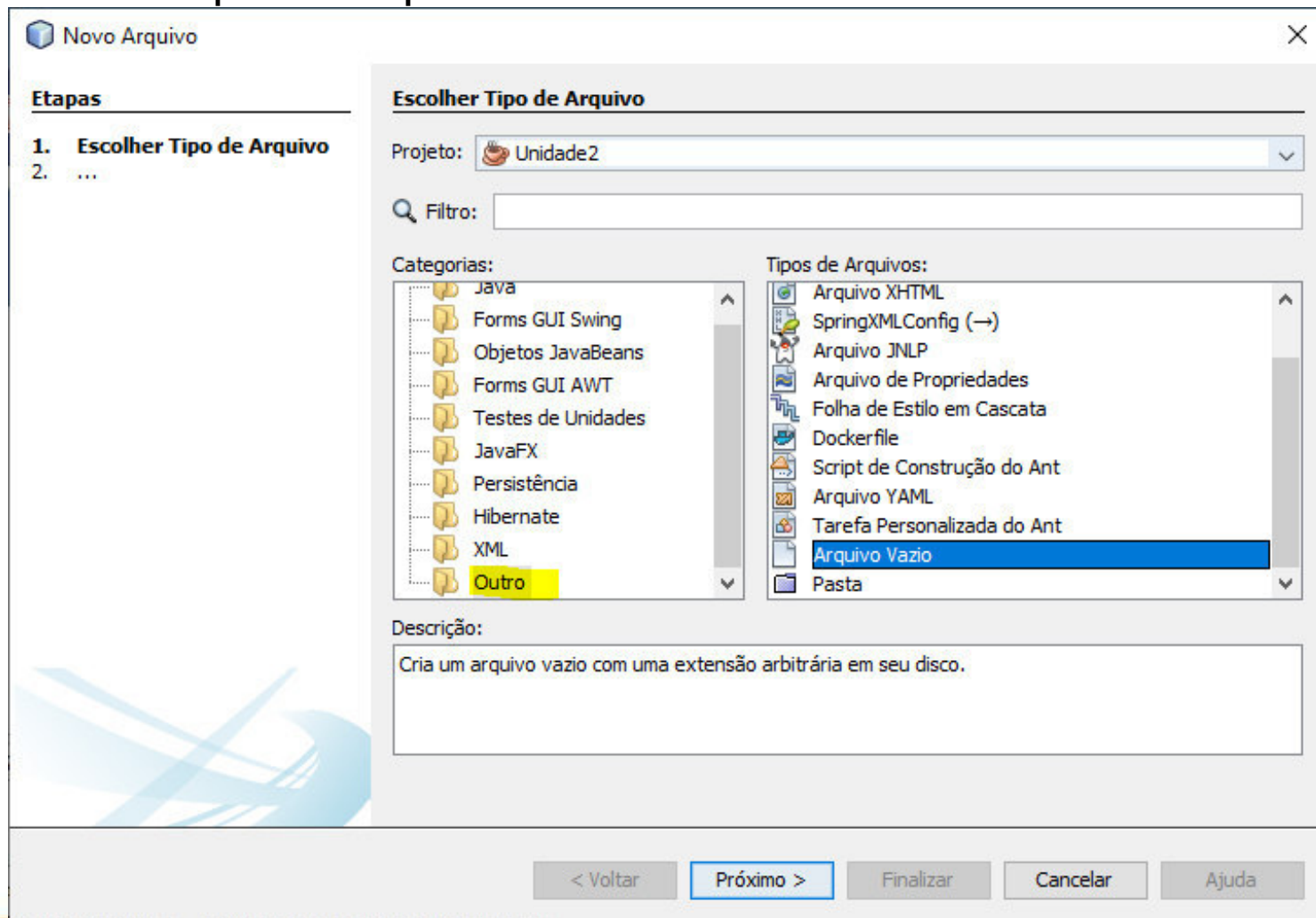
```
        }
```

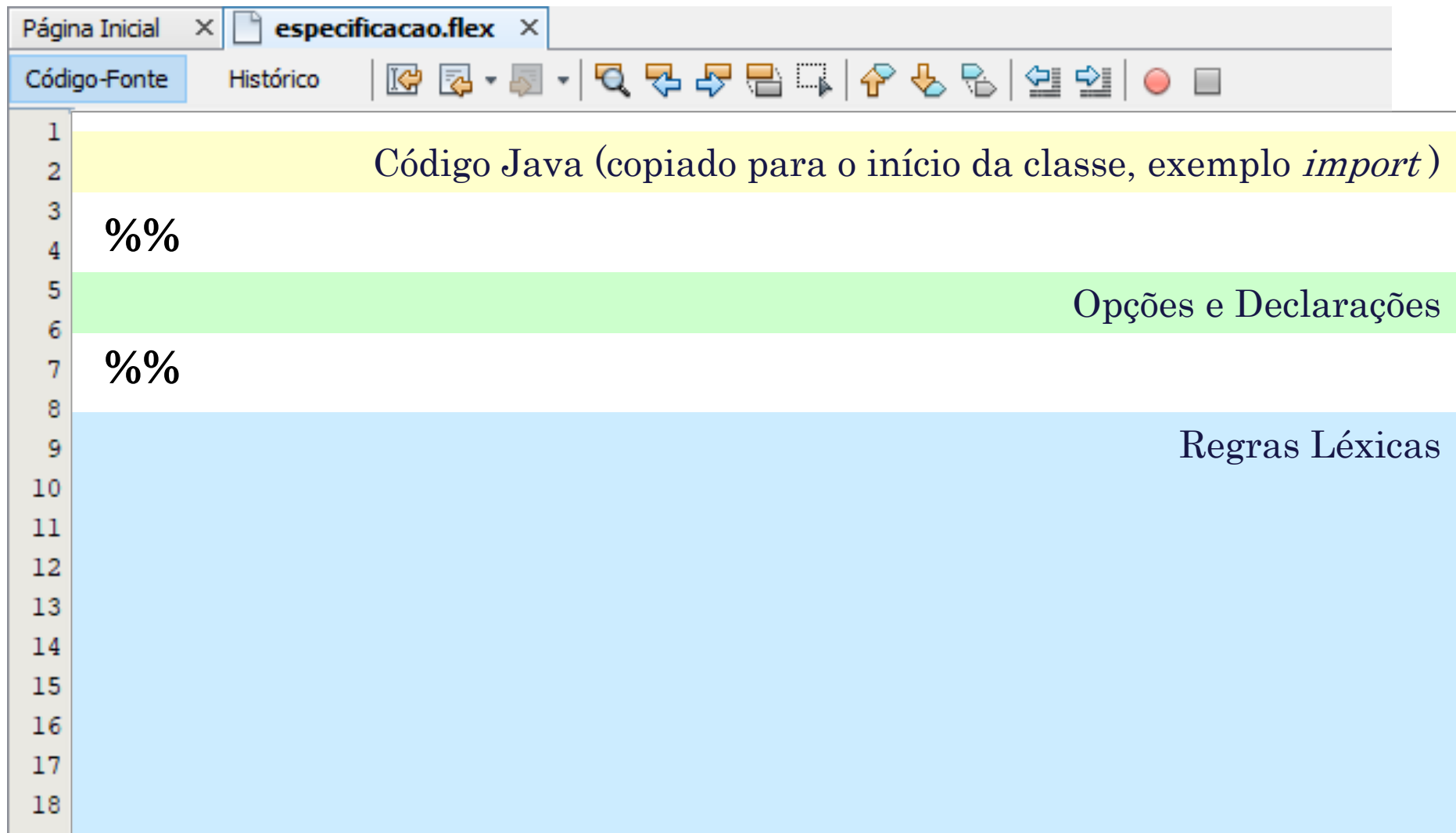
```
    }
```

Criar um arquivo de especificação FLEX

Criar um arquivo de especificação FLEX

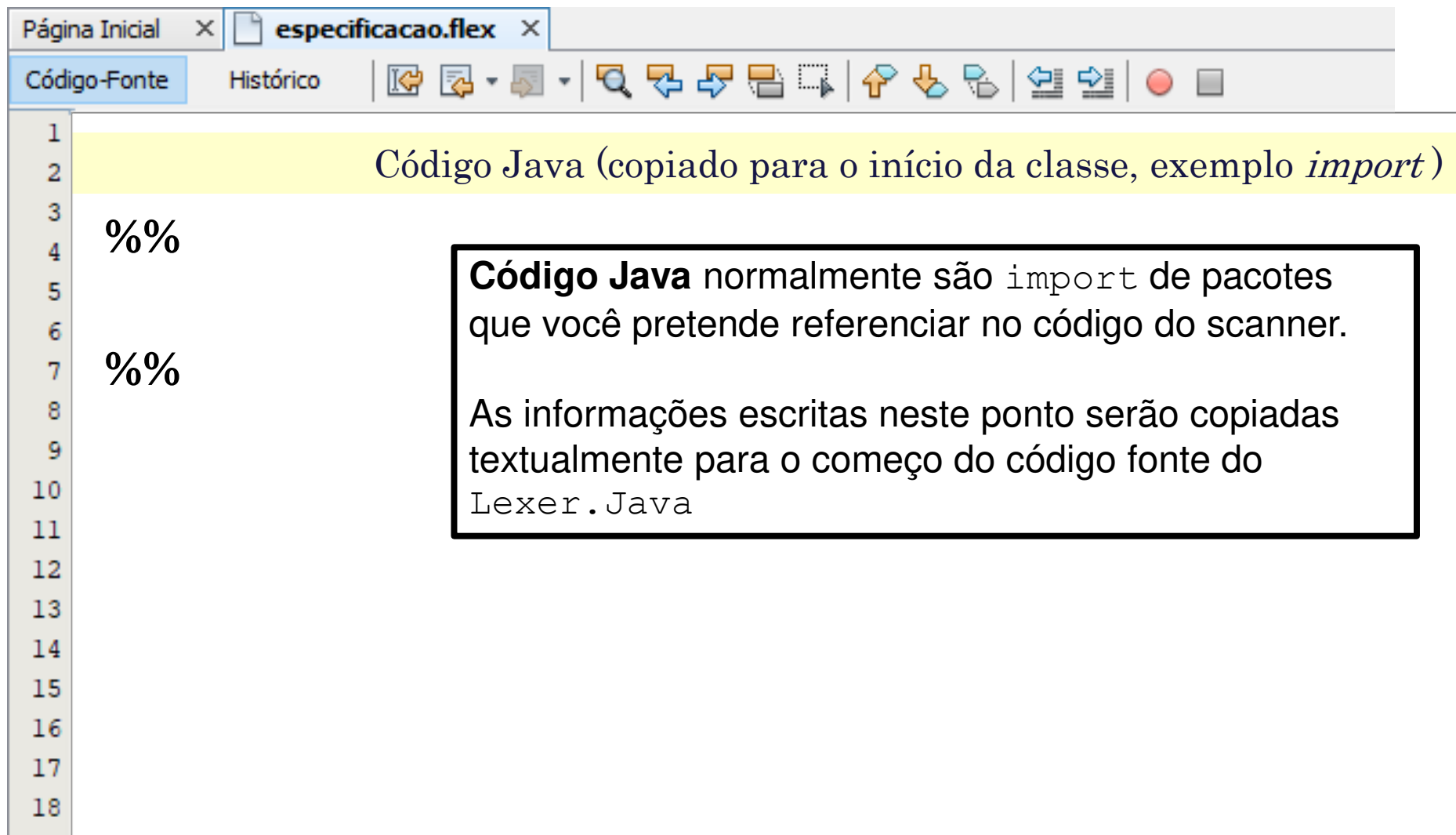
- Botão direito sobre o pacote “Compilador”, Novo > Outros...
- Selecione a categoria “Outro” > Arquivo Vazio
- Nome do arquivo: especificacao.flex





The screenshot shows a Flex IDE window with the following structure:

- Window Title:** `especificacao.flex`
- Menu Bar:** `Código-Fonte` (highlighted), `Histórico`
- Toolbar:** Contains icons for file operations (open, save, print), editing (undo, redo, cut, copy, paste), and navigation (find, go to line, zoom in, zoom out).
- Editor Content:**
 - Line 1:** `Código Java (copiado para o início da classe, exemplo import)` (highlighted in yellow)
 - Line 2:** `%%`
 - Line 5:** `Opções e Declarações` (highlighted in green)
 - Line 7:** `%%`
 - Line 9:** `Regras Léxicas` (highlighted in blue)



Página Inicial x especificacao.flex x

Código-Fonte Histórico

1

2 Código Java (copiado para o início da classe, exemplo *import*)

3

4 %%

5

6

7 %%

8

9

10

11

12

13

14

15

16

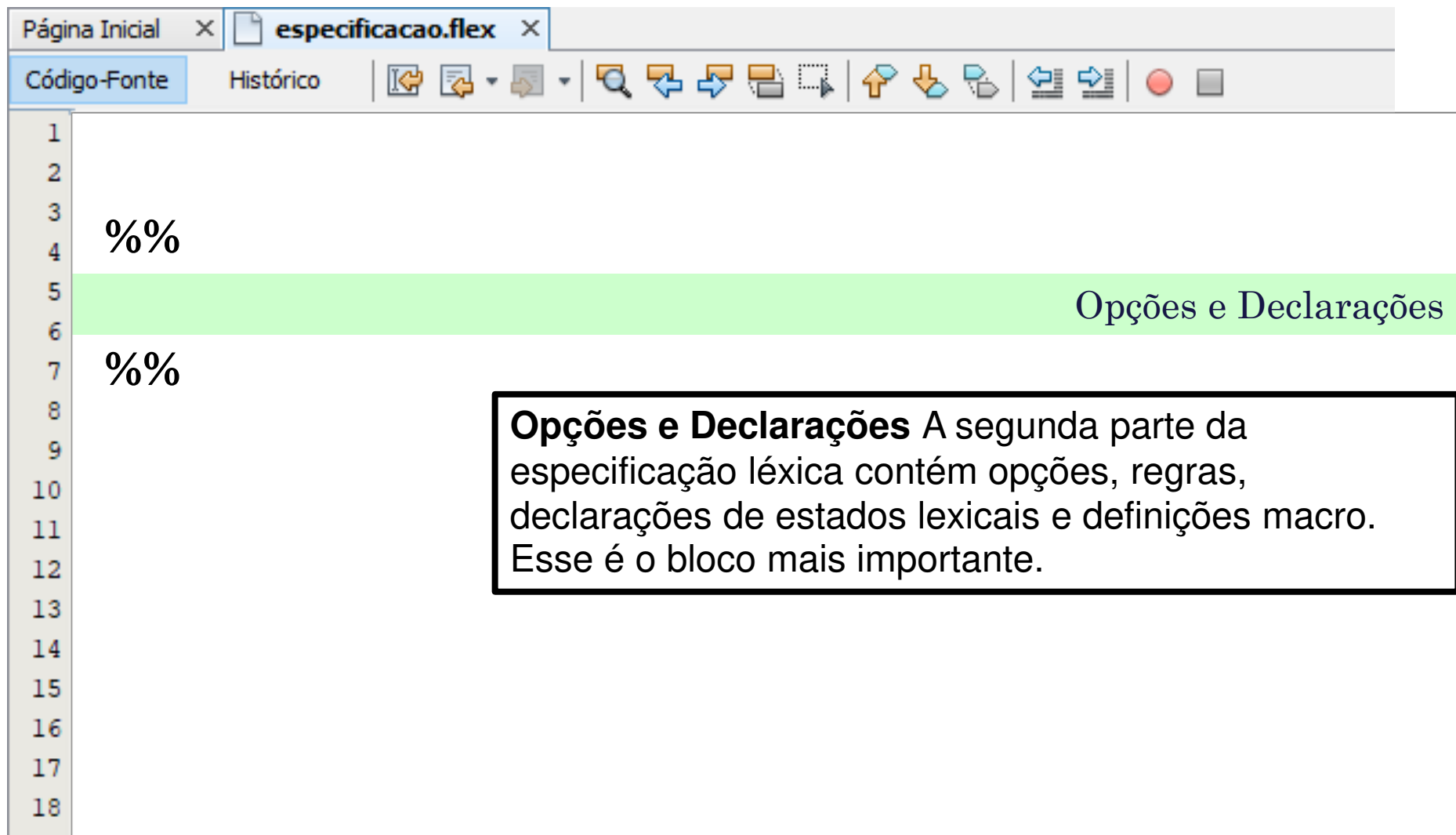
17

18

19

Código Java normalmente são `import` de pacotes que você pretende referenciar no código do scanner.

As informações escritas neste ponto serão copiadas textualmente para o começo do código fonte do `Lexer.java`



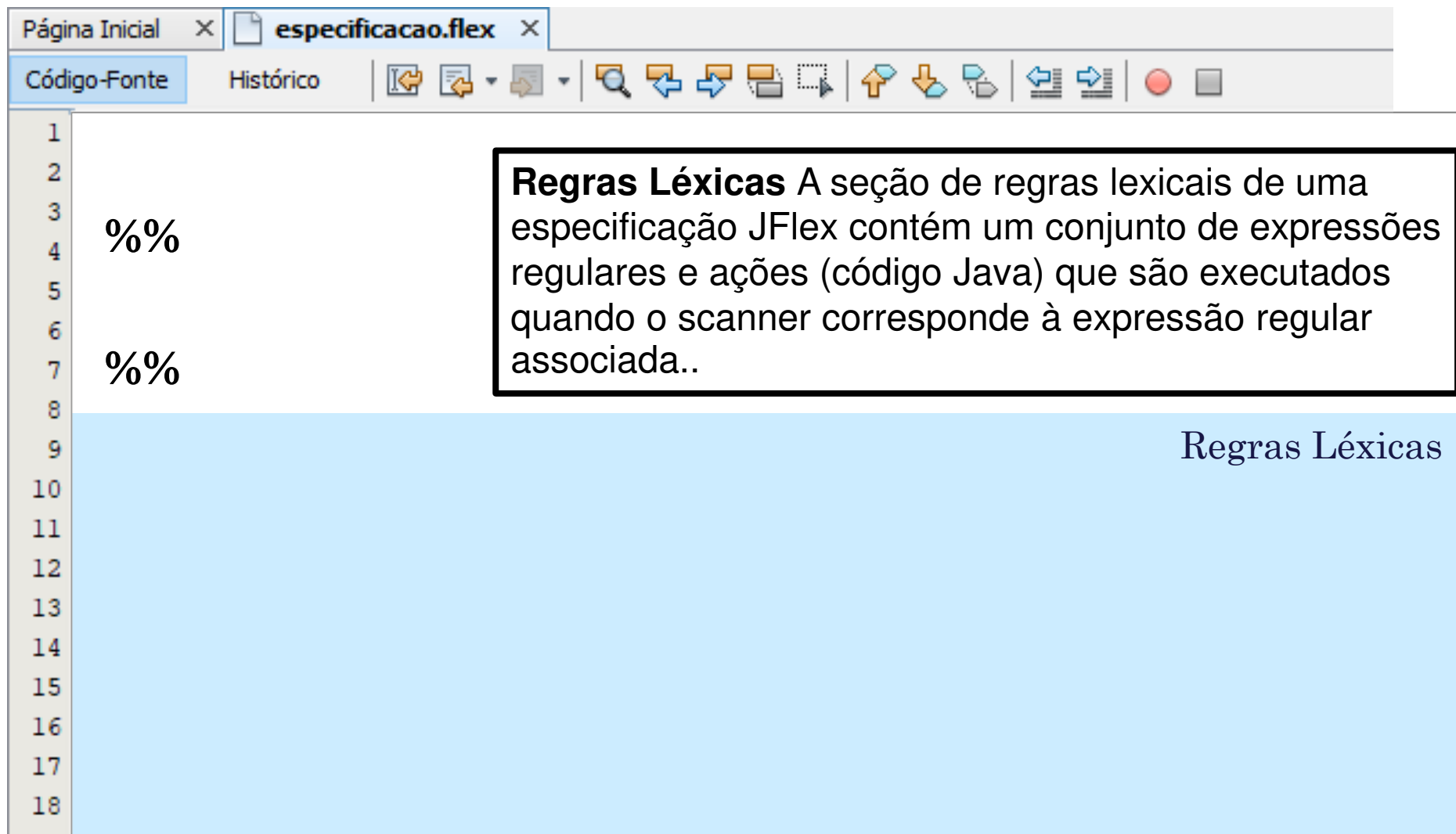
Página Inicial × especificacao.flex ×

Código-Fonte Histórico

1
2
3
4 %%
5 Opções e Declarações
6
7 %%
8
9
10
11
12
13
14
15
16
17
18

Opções e Declarações A segunda parte da especificação léxica contém opções, regras, declarações de estados lexicais e definições macro. Esse é o bloco mais importante.

Especificações Léxicas



Página Inicial × especificacao.flex ×

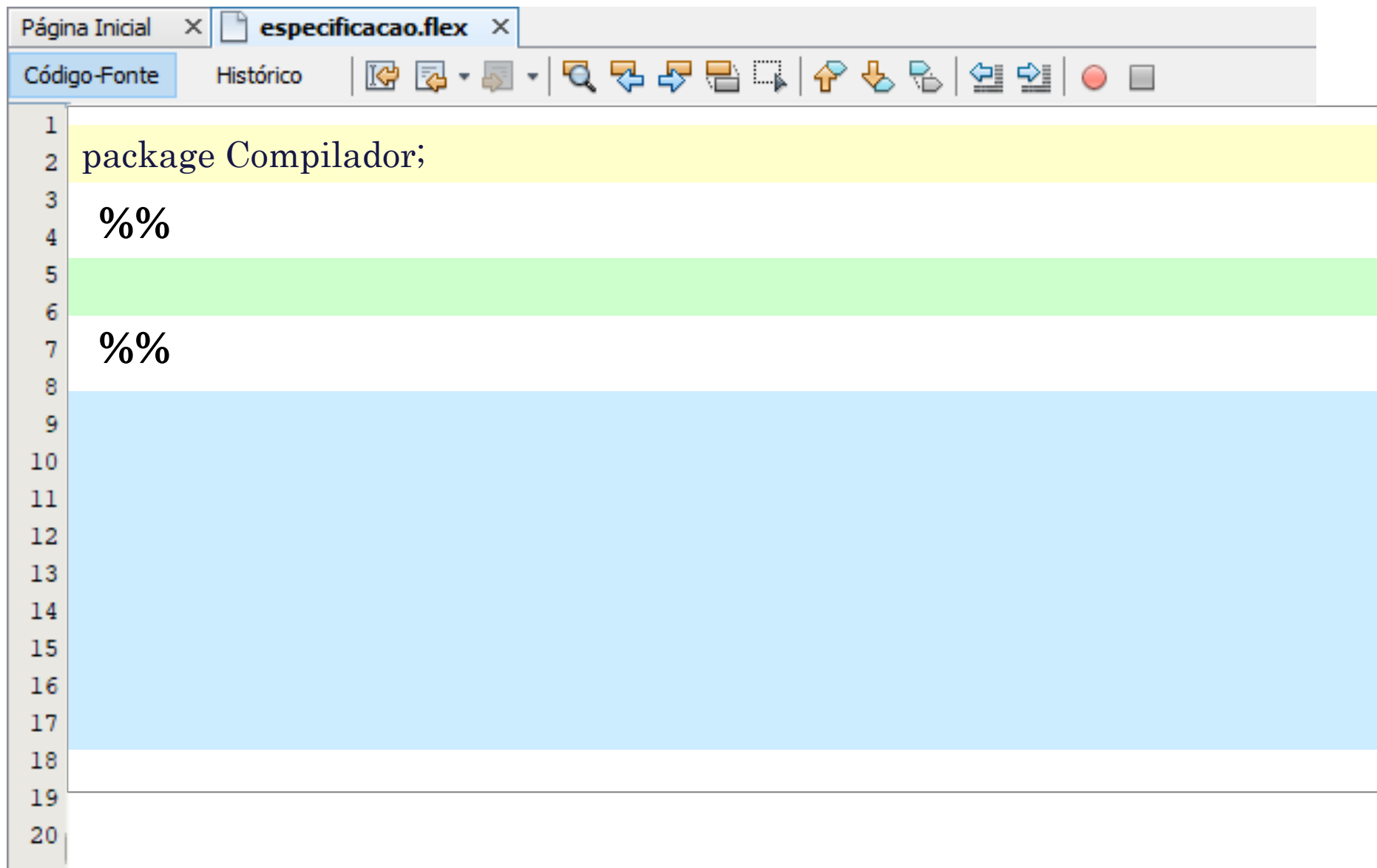
Código-Fonte Histórico

1
2
3 %%
4 %%
5
6
7 %%
8
9
10
11
12
13
14
15
16
17
18

Regras Léxicas A seção de regras lexicais de uma especificação JFlex contém um conjunto de expressões regulares e ações (código Java) que são executados quando o scanner corresponde à expressão regular associada..

Regras Léxicas

Primeira especificação

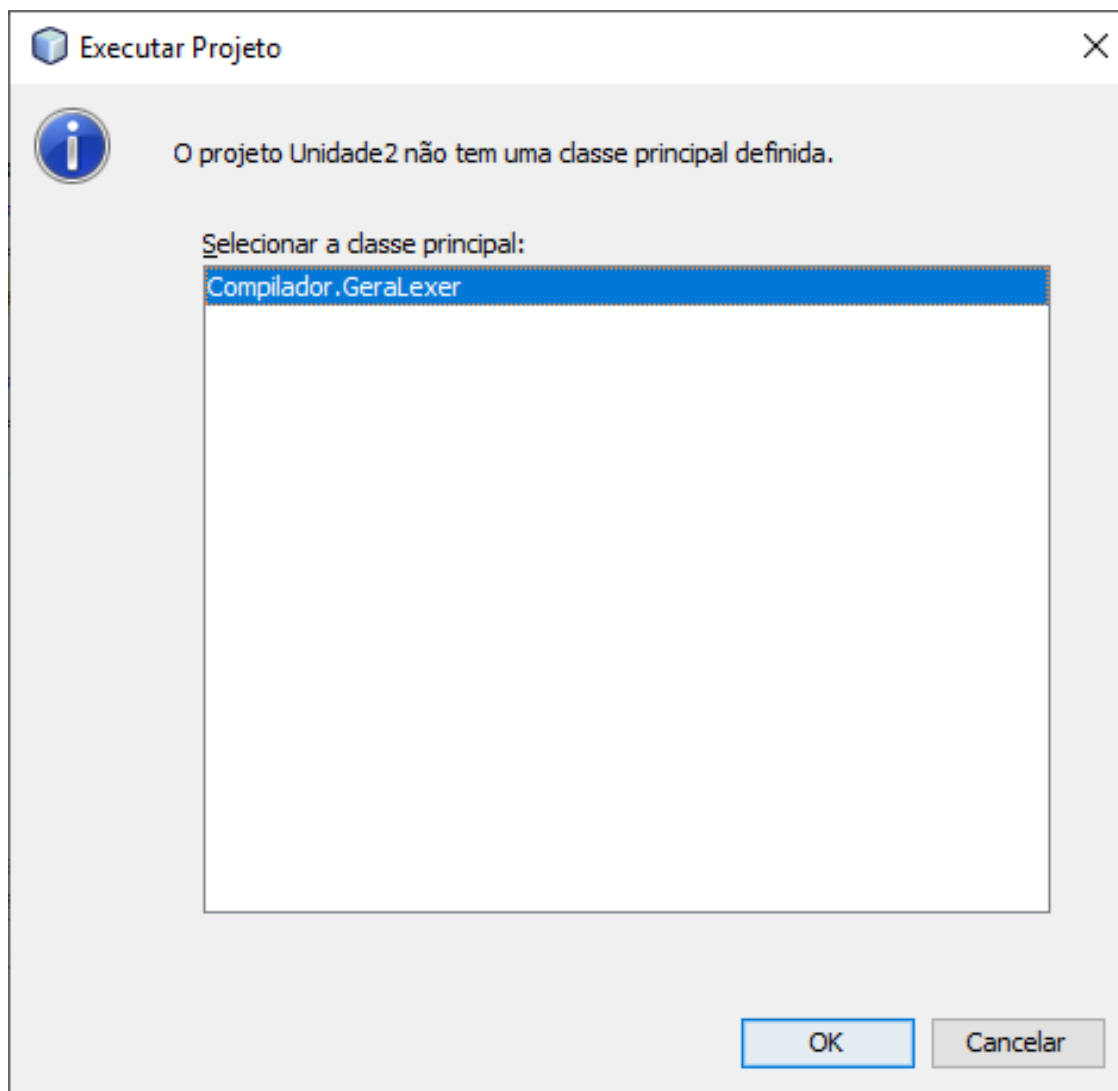


```
1 package Compilador;
2 %%
3
4 %%
5
6
7 %%
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Executar Projeto

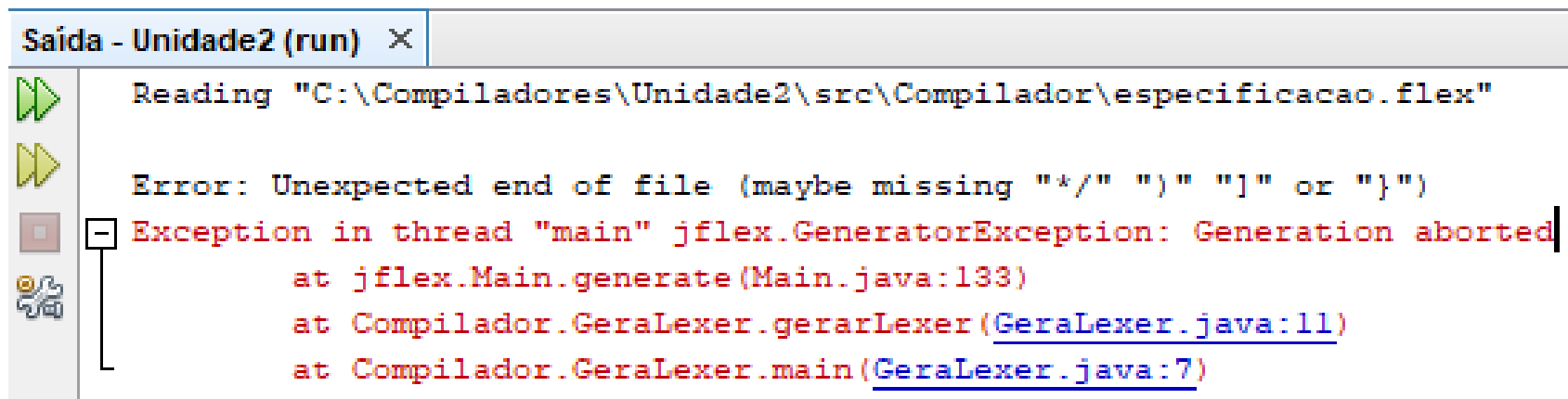
Executar o projeto

- Será questionado qual será a classe principal definida:



Primeiro erro

Error: Unexpected end of file (maybe missing "*" or ")" "]" or "{")



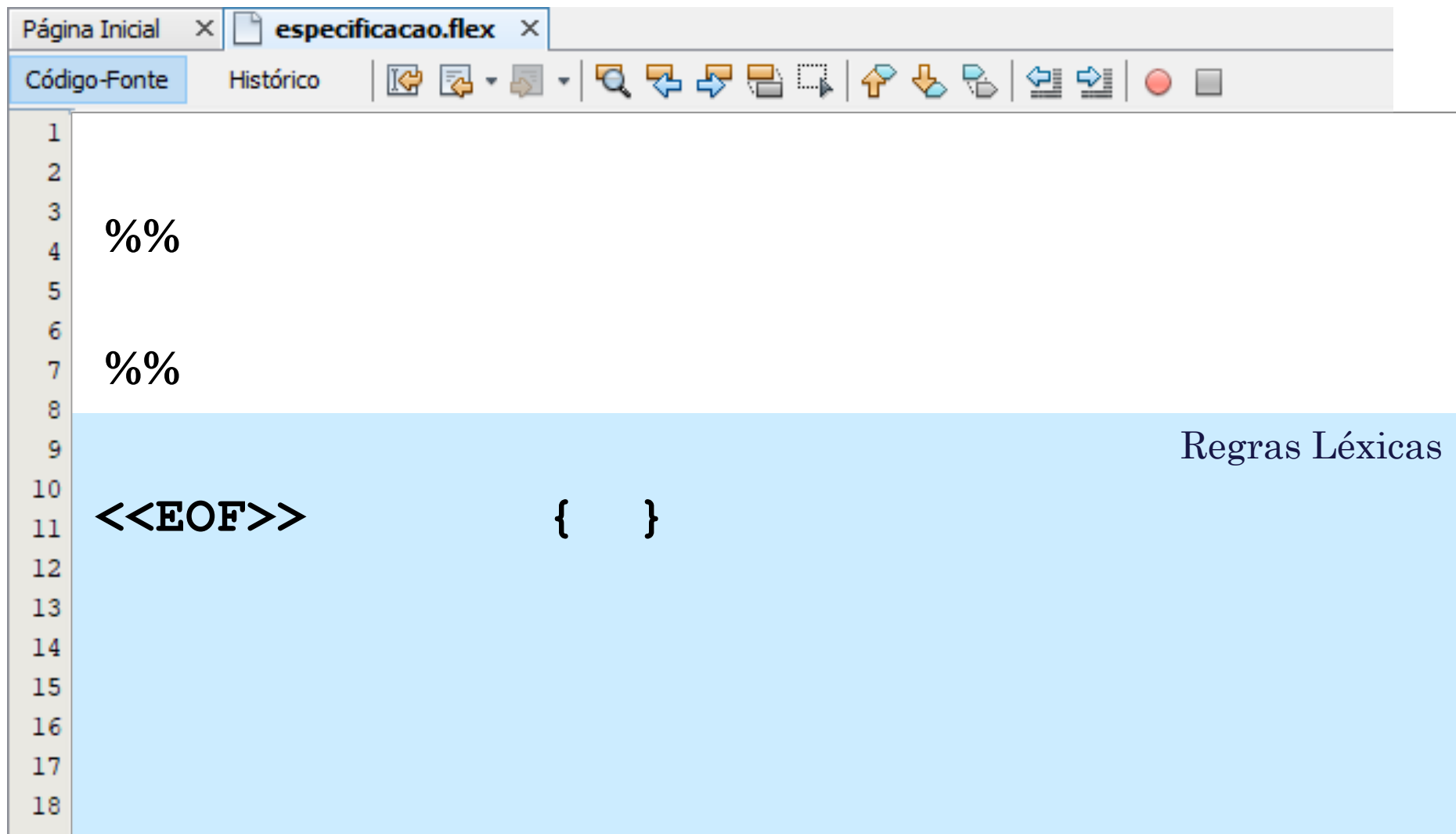
The screenshot shows a console window titled "Saída - Unidade2 (run)". It contains the following text:

```
Reading "C:\Compiladores\Unidade2\src\Compilador\especificacao.flex"

Error: Unexpected end of file (maybe missing "*" or ")" "]" or "{")

Exception in thread "main" jflex.GeneratorException: Generation aborted
    at jflex.Main.generate(Main.java:133)
    at Compilador.GeraLexer.gerarLexer(GeraLexer.java:11)
    at Compilador.GeraLexer.main(GeraLexer.java:7)
```

É obrigatório possuir Regras léxicas



```
1
2
3
4 %%
5
6
7 %%
8
9
10
11 <<EOF>> { }
12
13
14
15
16
17
18
```

Regras Léxicas

Segundo erro

Reading

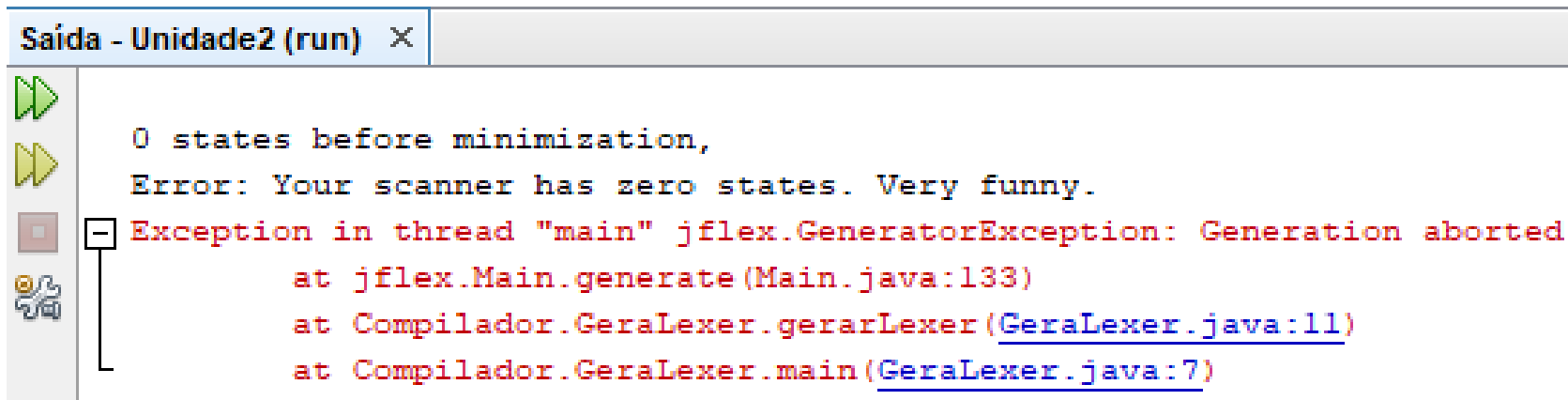
"C:\Compiladores\Unidade2\src\Compilador\especificacao.flex"

Constructing NFA : 2 states in NFA

Converting NFA to DFA :

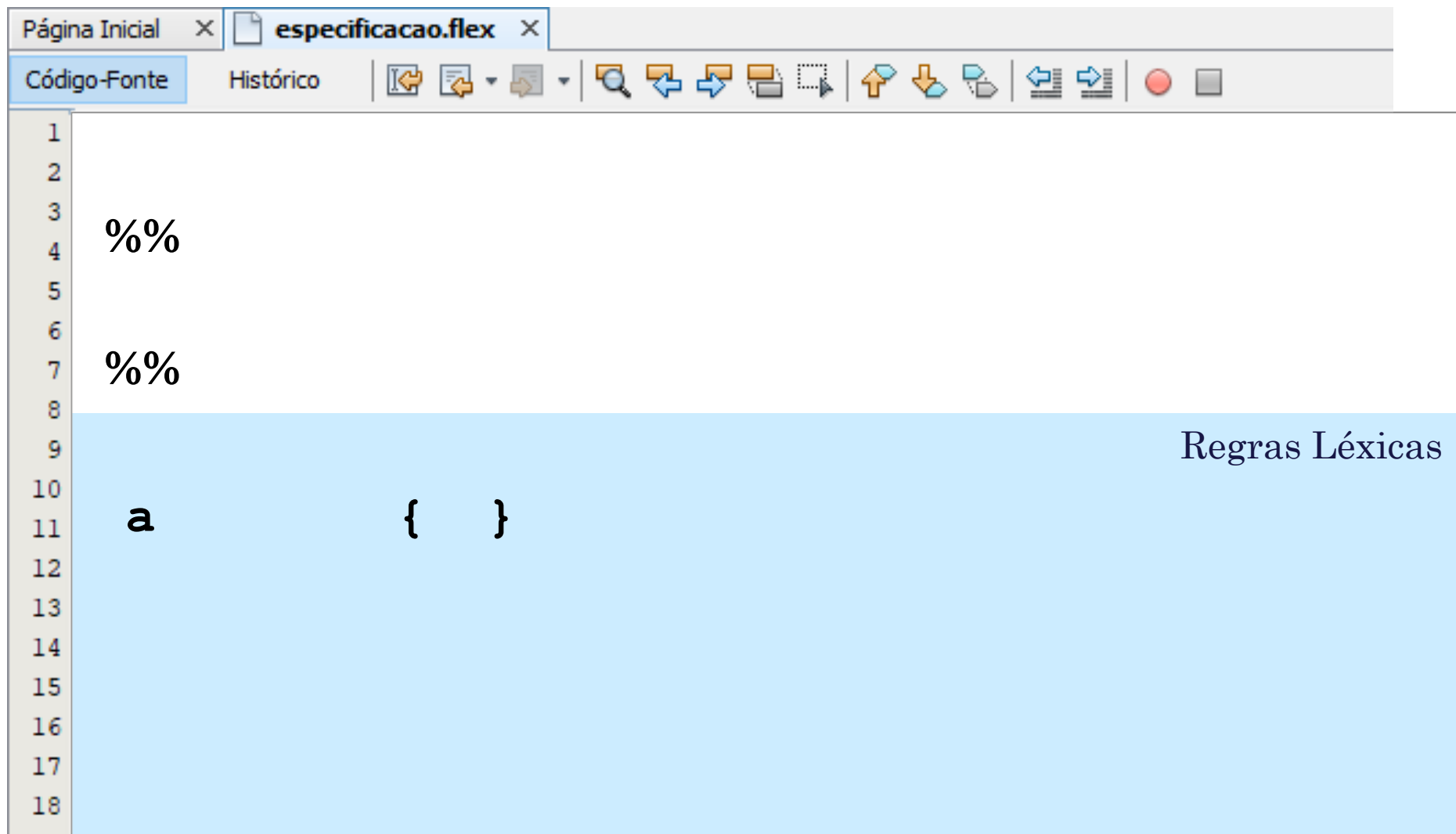
0 states before minimization,

Error: Your scanner has zero states. **Very funny.**



```
Saída - Unidade2 (run) ×  
0 states before minimization,  
Error: Your scanner has zero states. Very funny.  
Exception in thread "main" jflex.GeneratorException: Generation aborted  
    at jflex.Main.generate(Main.java:133)  
    at Compilador.GeraLexer.gerarLexer(GeraLexer.java:11)  
    at Compilador.GeraLexer.main(GeraLexer.java:7)
```

É obrigatório possuir Regras léxicas



The screenshot shows a Flex IDE window with the file 'especificacao.flex' open. The interface includes a toolbar with icons for file operations, editing, and running. The main editor area contains the following code:

```
1  
2  
3  
4 %%  
5  
6  
7 %%  
8  
9  
10  
11 a {  
12  
13  
14  
15  
16  
17  
18
```

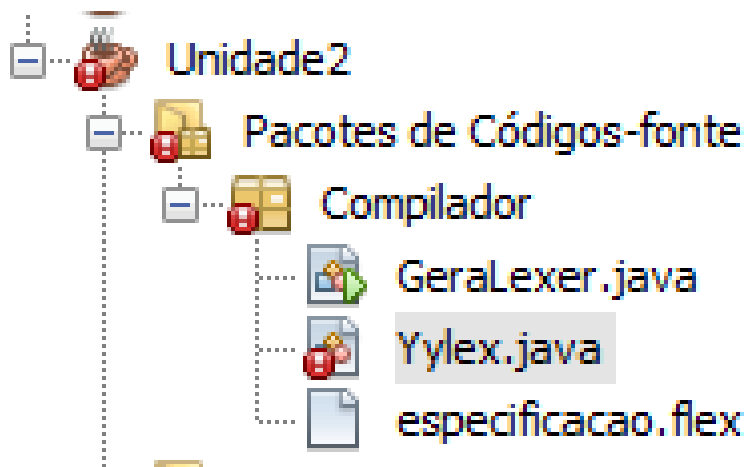
The text 'Regras Léxicas' is displayed in the bottom right corner of the editor area.

Analizador Léxico gerado com sucesso

```
run:
Reading
"C:\Compiladores\Unidade2\src\Compilador\especificacao.flex"
Constructing NFA : 4 states in NFA
Converting NFA to DFA :
.
3 states before minimization, 2 states in minimized DFA
Writing code to
"C:\Compiladores\Unidade2\src\Compilador\Yylex.java"
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```


Temos um novo problema

- Note que há um problema no nosso analisador léxico.



O método YYLEX() necessita retornar um TIPO, por exemplo, INT, FLOAT, ou um NOVO TIPO desenvolvido pelo programador.



```

451  L  */
452  L  -
453  L  -
454  L  -

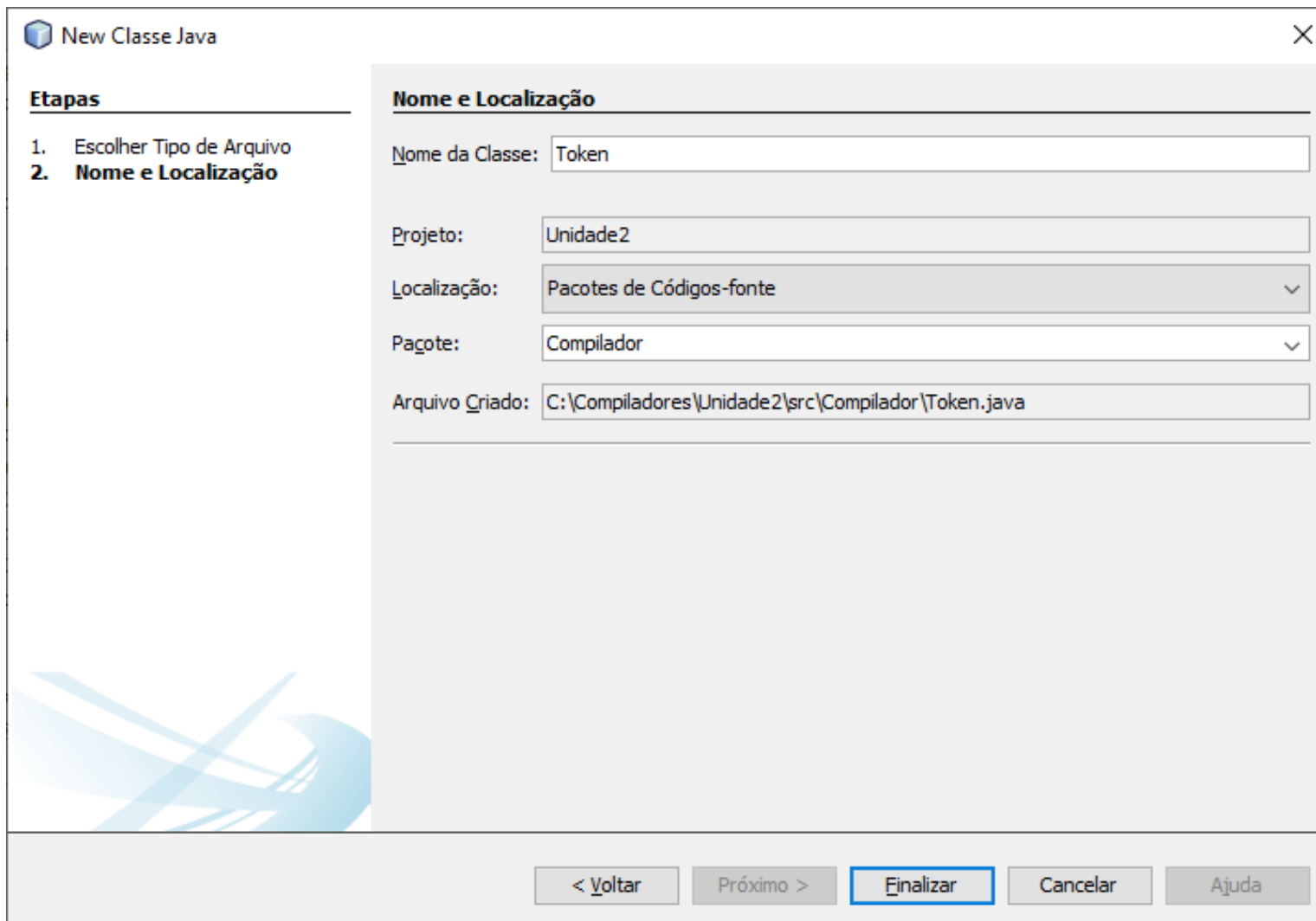
public Yytoken yylex() throws java.io.IOException {
    int zzInput;
    int zzAction;
  
```

(tipo do *token*, **lexema**)

Criando a classe Token

Criando a Classe Token

- No pacote Compilador criar Novo > Classe Java...



New Classe Java

Etapas

- Escolher Tipo de Arquivo
- Nome e Localização**

Nome e Localização

Nome da Classe: Token

Projeto: Unidade2

Localização: Pacotes de Códigos-fonte

Pacote: Compilador

Arquivo Criado: C:\Compiladores\Unidade2\src\Compilador\Token.java

< Voltar Próximo > **Finalizar** Cancelar Ajuda

Criando a Classe Token

- É preciso construir a classe *Token* do tipo eNum, com as constantes declaradas na especificação para cada *token*.
- Inicialmente teremos apenas duas constantes.

```
1 package Compilador;  
2 public enum Token {  
3     // Identifica os tipos de tokens  
4     NOME_VARIAVEL, INT;  
5 }
```