ATENÇÃO

- Esta apresentação **não substitui** a leitura da bibliografia complementar;

- **As provas desta disciplina terão como base:**

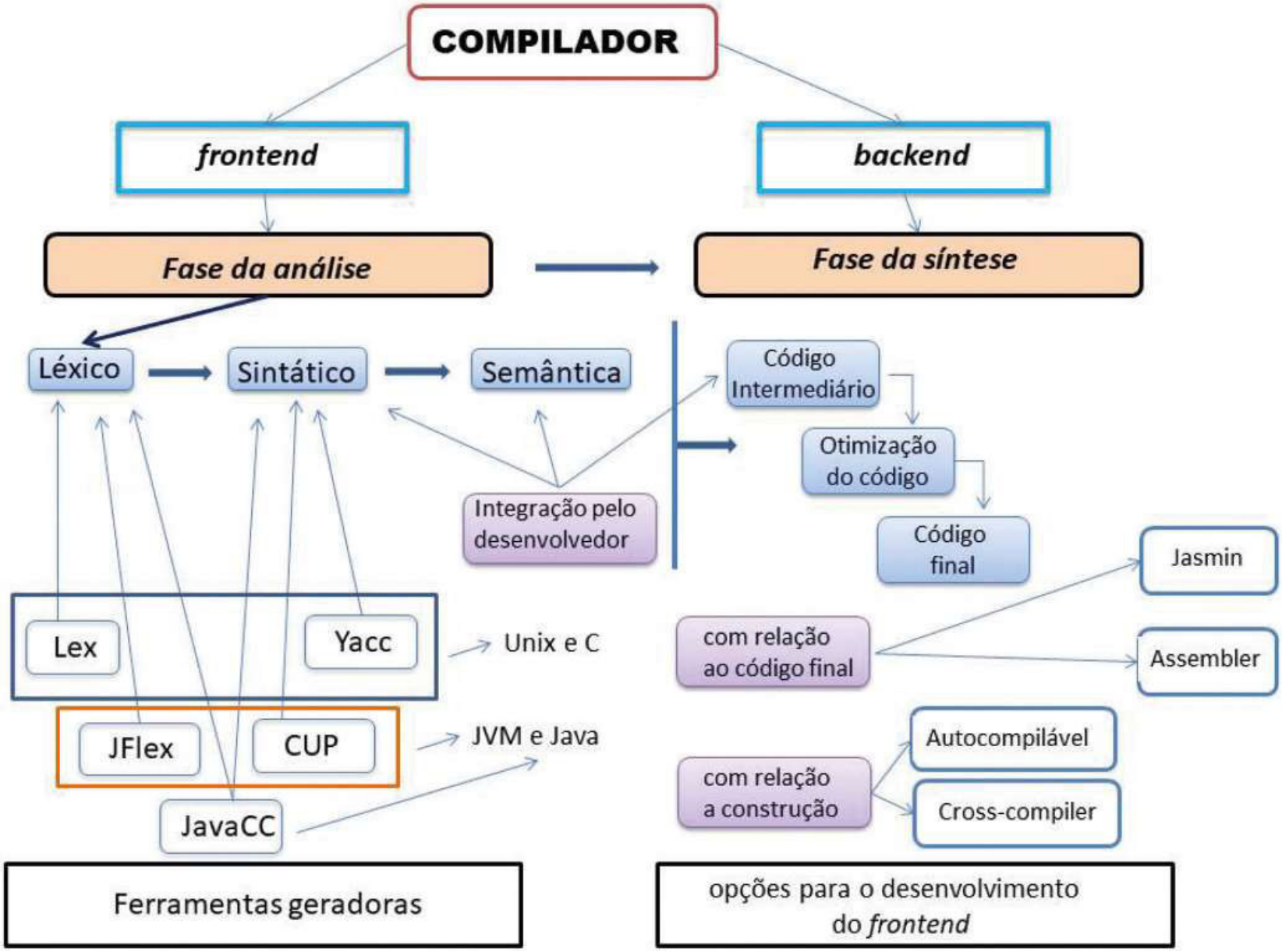
  - **O conteúdo dos livros indicados no PEA;**
  - **As atividades e exemplos dados em sala de aula;**
  - **O conteúdo disponíveis no AVA;**

Estudos

- Estudar as seções 2.1, 2.2 e 2.3 do AVA.

Construção de um compilador

Mapa conceitual do processo de construção do compilador



Para consultar:

- THE CATALOG - Este site apresenta um catálogo com opções de ferramentas para o *frontend* e para o *backend*, e o direciona a os sites oficiais de cada uma.
  - http://catalog.compilertools.net/

Frontend

Fase de Análise

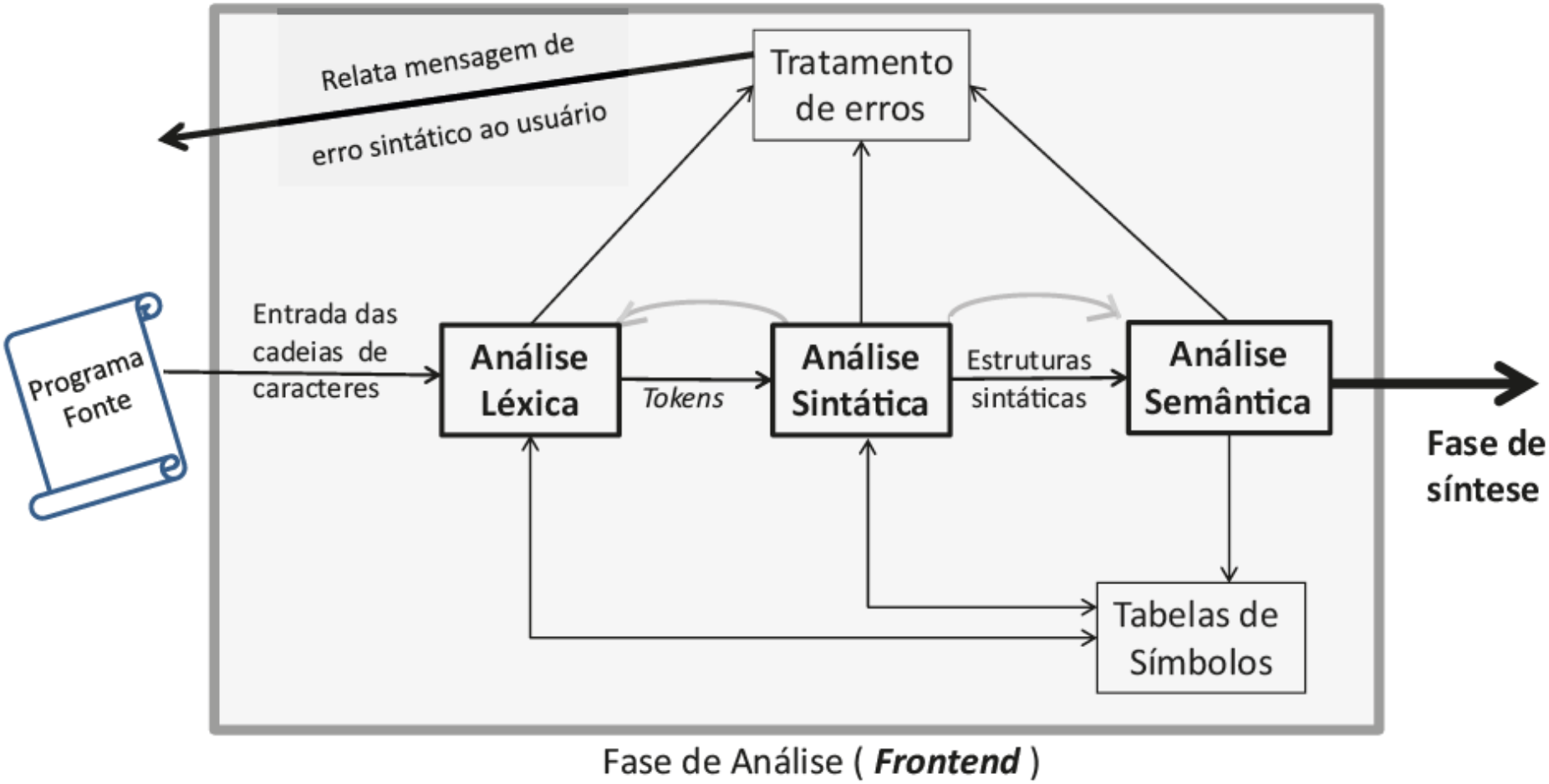
- **Análise léxica** lê os caracteres de entrada, gera um fluxo de dados, os *tokens* e os disponibiliza para o analisador sintático.

- O **analisador sintático** analisa o fluxo de dados gerado de acordo com a gramática livre de contexto (estruturas sintáticas) e, quando necessário, aciona a **análise semântica** para verificar se os elementos presentes na estrutura sintática são compatíveis.

- Se em cada uma dessas etapas não for encontrado erro de escrita no programa fonte, o fluxo de dados será enviado para a fase seguinte, a de síntese, caso contrário, os erros de cada etapa da análise serão reportados ao usuário, o programador, e o processo de compilação será abortado.

Fase de Análise

- A função da análise léxica é ler os caracteres de entrada e produzir uma sequência de *tokens* que serão utilizados pela análise sintática.



Analisador léxico

- **A função do analisador léxico é reconhecer os *tokens* associados às expressões regulares**, ou seja, o subconjunto da linguagem livre de contexto pertencente apenas às linguagens regulares, composto pelos elementos básicos, tais como:

  – identificadores, operadores, constantes, comentários, caracteres especiais e tipos compostos.

Analisador léxico

- **Porque separar o *lexer* (analisador léxico) do *parser* (analisador sintático)?**
  - *Lexer* classifica as palavras; é um processo simples.
  - *Parser* gera derivações gramaticais; é um processo complexo e consequentemente lento.
  - A existência de um *lexer* leva a um *parser* menor e mais rápido, além de agilizar a manutenção.

Dicionário de termos

Dicionário de termos

- ***Token*:** é o nome da produção da gramática.

- ***Lexema*:** é o elemento do token.
  - Se comparássemos a um programa, poderíamos dizer que token é o nome da variável e o lexema o conteúdo da variável.

Dicionário de termos

- ***Scanner*:** é o gerador de analisador léxico, por exemplo: LEX, JFLEX.
  - O scanner lê a especificação em um padrão EBNF e gera um programa que analisa um arquivo fonte (programa) escrito de acordo com a especificação, por exemplo: o Lex gera em C e o JFLEX gera em Java.

- ***Lexer*:** é o analisador léxico. É o programa gerado pelo scanner.

Dicionário de termos

- ***Parser*:** é o gerador de analisador sintático, por exemplo: Yacc, CUP.
  - O parser lê a especificação da GLC (gramática livre de contexto) no padrão EBNF, recebe os tokens analisados pelo lexer e gera um programa que analisa a sintaxe de um arquivo fonte (programa) escrito de acordo com a especificação GLC, por exemplo: o Yacc gera em C e o CUP gera em Java.

- ***Parsing*:** é o analisador sintático. É o programa gerado pelo parser.

Token e Lexemas

Reconhecimento Token e Lexema

- Quando tratamos de análise léxica, devemos encontrar no programa fonte os padrões correspondentes ao par (tipo do *token*, **lexema**).

```
1  int x = 2;
2  x = 10 + 1b * x;
```

Identificamos os seguintes padrões para cada par (tipo do *token*, lexema):

(<tipo de dado>, **int**)
(<variavel>, **x**)
(<atribuicao>, **=**)
(<const_numerica>, **2**)
(<terminador>, **;**)

(<variavel>, **x**)
(<atribuicao>, **=**)
(<const_numerica>, **10**)
(<op_aritmetico>, **+**)
(<*Não_Reconhecido*>, 1b)
(<op_aritmetico>, *)
(<variavel>, **x**)
(<terminador>, **;**)

Reconhecimento Token e Lexema

- Agora vamos entender os passos do analisador léxico no processo de reconhecimento dos tokens:
  - Eliminar espaços em branco;
  - Bufferização;
  - Especificação dos elementos léxicos;

Construção de um analisador léxico (*scanners*)

Análise léxica

- Claramente, os procedimentos até aqui apresentados definem o ferramental necessário para a **construção de um analisador léxico**;

- O analisador léxico reconhece sentenças definidas por uma **expressão regular**;

- Como estudado, as linguagens regulares podem ser especificadas por expressões regulares e há um autômato finito determinístico (AFD) que as representa.

Geradores de analisadores léxicos

LEX (*scanner* que gera *lexers*)

- Lex foi originalmente desenvolvido por Eric Schmidt e Mike Lesk;

- Esse tipo de ferramenta automatiza o processo de **criação do autômato** e o processo de reconhecimento de sentenças regulares a partir da especificação das **expressões regulares**;

- Uma das ferramentas mais tradicionais dessa classe é o programa ***Lex***;

FLEX

Anhanguera

- O Flex é uma evolução da ferramenta Lex sendo mais rápido (*Fast Lex*);
- Flex é uma ferramenta para **geração automática de analisadores léxicos** (*Scanners*), isto é, programas que reconhecem padrões léxicos num texto;
- Ao invés do programador **escrever manualmente** um programa que realize a identificação de padrões numa entrada, o uso do Flex/Lex permite que sejam apenas **especificados os padrões** desejados e as ações necessárias para processá-los;
- Para que Flex/Lex reconheçam padrões no texto, tais padrões devem ser descritos através de **expressões regulares**.

FLEX nas Ling. de Programação

- Você pode encontrar:
  - FLEX para C
  - FLEX para C++
  - JFLEX para Java
  - E outros *Lexers* ...

Como o Flex funciona?

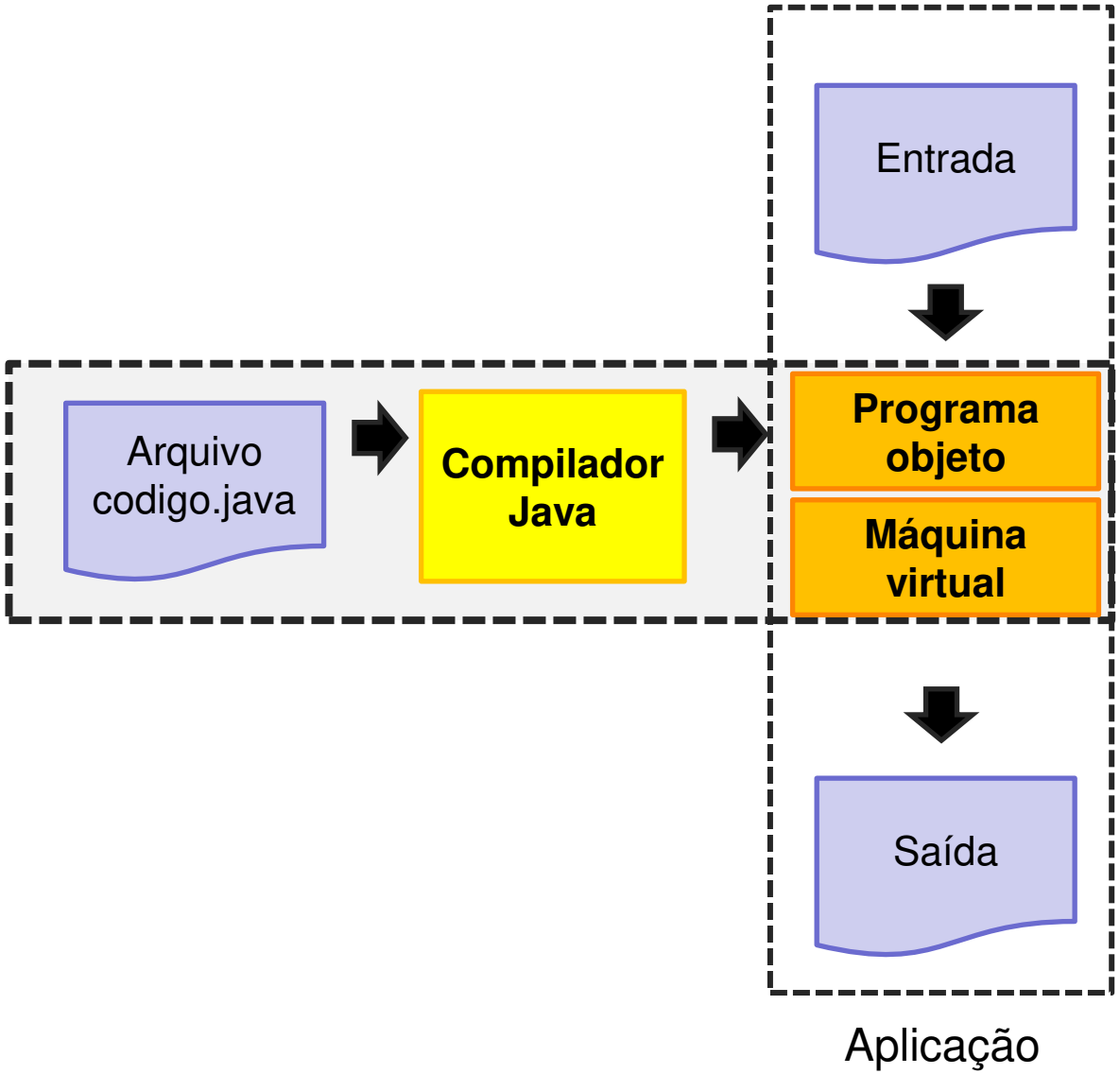
Anhanguera

- Flex lê os **arquivos de entrada** obtendo assim uma descrição do *Lexers* a ser gerado;
  - Este arquivo de entrada é chamado de **arquivo de especificação**;
- A especificação é realizada na forma de pares de **expressões regulares** e **código Java**;
- As regras definem simultaneamente **quais padrões devem ser procurados** e quais as ações devem ser executadas quando é identificado um padrão;
- Para cada padrão desejado pode ser associado um conjunto de ações escritas sob a forma de código Java;

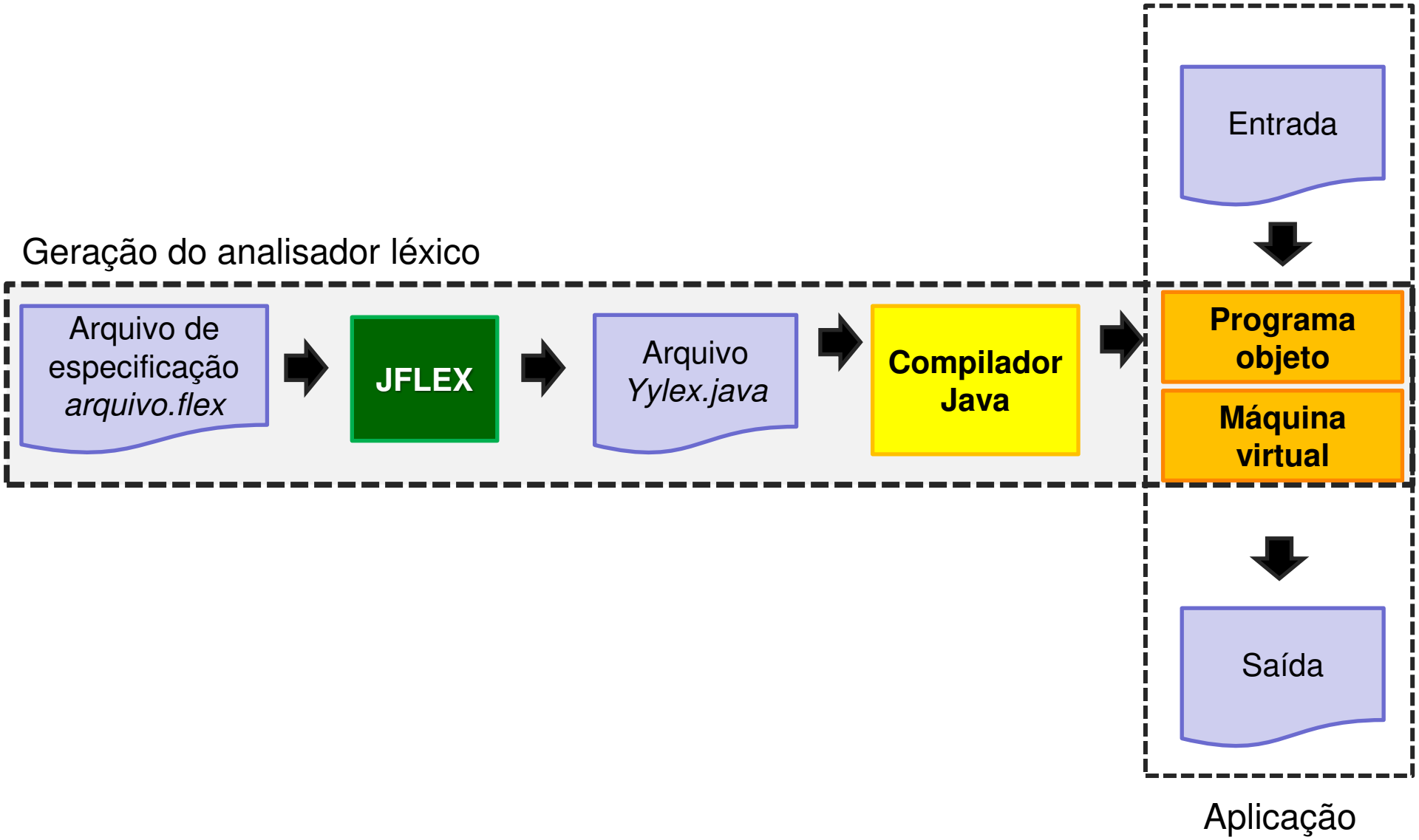
Criar um *Lexer* manualmente em Java

Anhanguera

Geração do analisador léxico



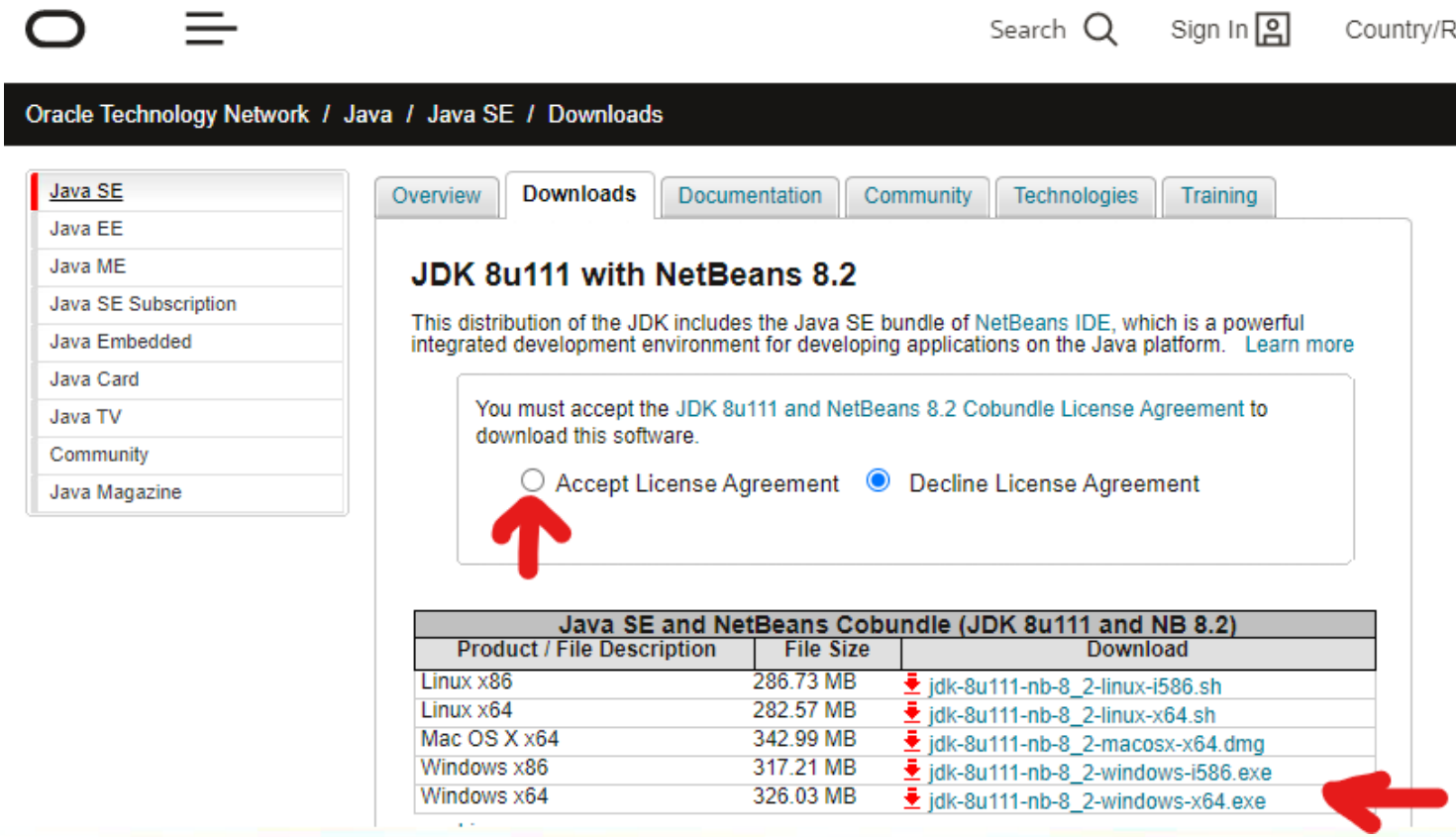
Utilizando *scanner* para gerar um *lexer*



Netbeans e JFLEX

Para desenvolvermos o analisador léxico

- NetBeans 8.2 + JDK
  - https://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-3413139-esa.html
- Faça a instalação do Netbeans em seu computador.



Para desenvolvermos o analisador léxico

- Jflex 1.7
  - https://kroton-my.sharepoint.com/:u:/g/personal/jpsiqueira_anhanguera_com/EVnLb_9YHHlIkb3Hbla_dz0Bzy_YwM2mMjzLi8Y7nUEuMw

- Crie em C:\ uma pasta chamada COMPILADORES

- Dentro de C:\COMPILADORES descompacte o arquivo Jflex 1.7

Prof. Me. João Paulo R. de Siqueira

jpsiqueira@anhanguera.com