



|       |   |   |   |
|-------|---|---|---|
|       |   |   |   |
| Trab. | - | - | X |
|       |   |   |   |

|                                     |                                 |               |
|-------------------------------------|---------------------------------|---------------|
| Curso:                              | Aspectos Teóricos da Computação | Código/Turma: |
| Professor: Ronaldo Gonçalves Junior |                                 | Data:         |
| Aluno(a):                           |                                 | Matrícula:    |
| Aluno(a):                           |                                 | Matrícula:    |

## INSTRUÇÕES PARA RESOLUÇÃO DO TRABALHO 3

- Preencher o campo Aluno(a) com o nome completo e o campo Matrícula para cada integrante da equipe
- Preencher os passos abaixo com código, fotos ou respostas à mão - apenas soluções legíveis serão corrigidas.
- Enviar o documento em formato PDF pelo AVA na área "Trabalhos"

# Compilador Básico Front-end

Este trabalho possui três etapas, a primeira sendo a construção de um Analisador Léxico usando JFlex. Em seguida, a equipe deve construir um Analisador Sintático usando JCup e, por fim, estes dois componentes devem ser integrados para processar código-fonte como *input*.

## Etapa 1: Analisador Léxico com JFlex ✓

Nesta etapa a equipe deve compreender e implementar um analisador léxico, também conhecido como Scanner. Utilize a estrutura léxica básica encontrada em linguagens de programação convencionais para gerar um analisador léxico com JFlex que gera tokens baseado no código-fonte como entrada. Os entregáveis dessa etapa incluem:

1. **Especificação:** arquivo .flex que define as regras léxicas para os seguintes tipos de token:
  - a. Keywords (if, else, while, return, etc.)
  - b. Identificadores
  - c. Constantes (números inteiros e pontos-flutuantes)
  - d. Operadores (+, -, \*, /, ==, !=, etc.)
  - e. Delimitadores (;, ,, {, }, (, ), etc.)
  - f. Comentários (single-line and multi-line)
2. **Classe Java:** classe gerada automaticamente pelo JFlex
3. **Script de teste:** escreva um script que usa o Scanner criado anteriormente para ler um arquivo de texto de entrada que contém código-fonte e imprime no console todos os tokens e seus respectivos tipos.

## Etapa 2: Analisador Sintático com JCup ✓

Nesta etapa a equipe deve compreender e implementar um analisador sintático, também conhecido Parser. Utilize a estrutura gramatical básica encontrada em linguagens de programação

convencionais para gerar um analisador sintático com JCup que gera uma árvore sintática baseada no código-fonte como entrada. Os entregáveis dessa etapa incluem:

1. **Especificação:** arquivo .cup que define a gramática para uma linguagem de programação simples, incluindo:
  - a. Estrutura do programa
  - b. Declaração de variáveis
  - c. Estruturas de controle (if-else, while loops)
  - d. Definições e chamadas de funções
  - e. Expressões e atribuições
2. **Classes Java:** classes geradas automaticamente pelo JCup
3. **Script de teste:** escreva um script que recebe tokens de entrada e constrói uma árvore sintática utilizando o Parser criado anteriormente e imprime a árvore para o console.

### Etapa 3: Integração ✓

Nesta etapa a equipe deve integrar os componentes produzidos nas etapas anteriores, para que a saída do Scanner, tokens, sejam usados como input para o Parser. O projeto deve combinar o uso das ferramentas JFlex e JCup e entregar os seguintes itens:

1. **Integração:** mostre que o Analisador Léxico opera corretamente em conjunto com o Analisador Sintático
2. **Código-fonte de entrada:**
  - a. Crie um arquivo *input.txt* contendo um programa simples na linguagem definida. Este arquivo deve conter várias construções, como declarações de variáveis, estruturas de controle, definição de funções e assim por diante.
  - b. Integre o JFlex scanner e o JCup parser.
  - c. Demonstre a solução funcionando em um código-fonte de entrada.
3. **Demonstração:** escreva um programa que demonstra a solução integrada para o código-fonte de entrada, mostrando o processo de geração de tokens e o processo de geração da árvore sintática.

### O que deve ser entregue no AVA:

1. Arquivo de especificação JFlex (.flex) e a classe Scanner Java gerada.
2. Arquivo de especificação JCup (.cup) e as classes Parser Java geradas.
3. Scripts de teste e demonstração de integração, imprimindo o processo completo.
4. Código-fonte de entrada (input.txt).
5. Um arquivo README explicando como compilar e executar o scanner, o parser e a solução integrada.

O trabalho deve ser apresentado na semana seguinte à entrega no AVA.