Paradigmas de Linguagens de Programação (PLP)

Professor: Carlo Marcelo Revoredo da Silva
Contato: revoredo@gmail.com

UNIVERSIDADE DE PERNAMBUCO

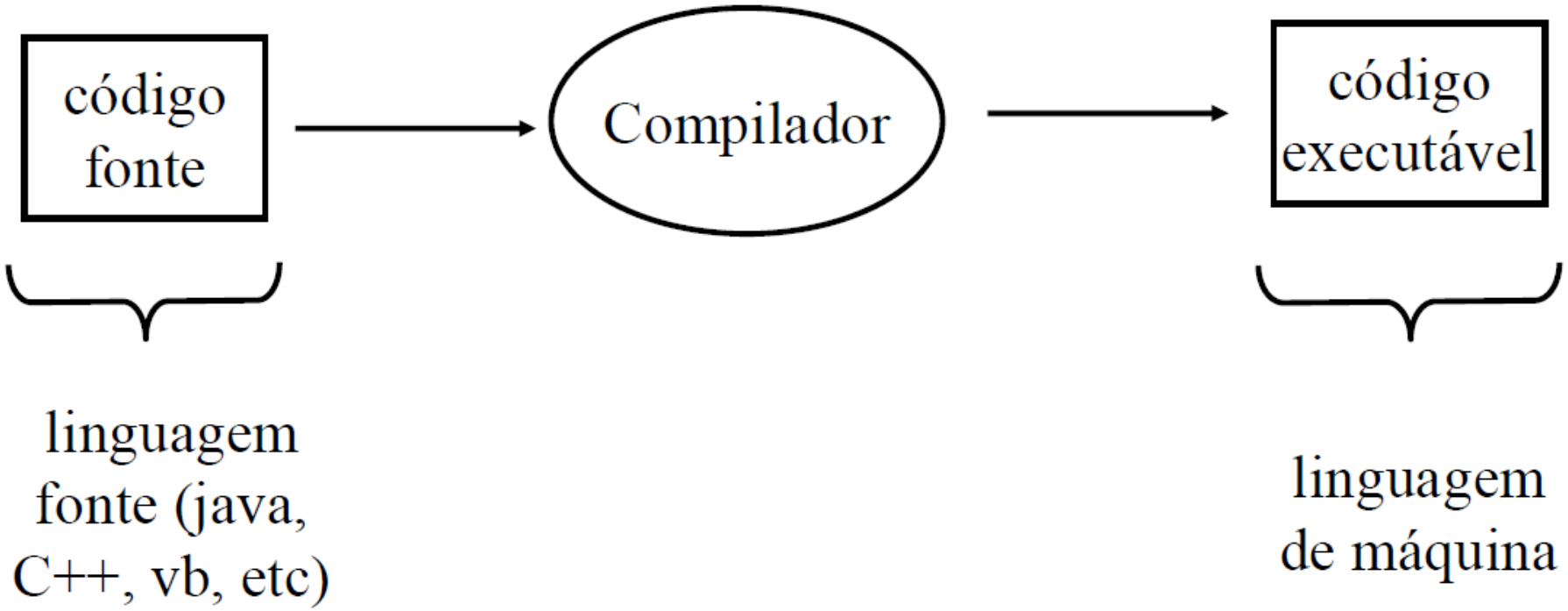
Sumário

- **O processo de compilação: análises léxica, sintática e semântica**
- **Compilação x Interpretação**
- **ANTLR**
- **DSL**

O processo de compilação

Compiladores

O processo de compilação



Compiladores

- A compilação de um programa envolve tipicamente os seguintes eventos:
  - Análise léxica
  - Análise sintática
  - Análise semântica
  - Tradução

Compiladores

- Análise Léxica
  - Ler o texto do código-fonte, caractere a caractere
  - Identificar elementos léxicos da linguagem: identificadores, palavras reservadas, constantes, operadores (TOKENS)
  - Ignorar comentários, brancos, tabs, quebra de linhas
  - Processar importações de bibliotecas

Compiladores

- Análise Sintática
  - Identificar na sequência de elementos léxicos as construções da linguagem

  - if (x > y) {
    System.out.println(“Maior!”);
  - }else{
    System.out.println(“Menor ou igual!”);
  - }

  - Identifica a expressão relacional x > y
  - Identifica uma estrutura de seleção composta
  - Quais operações estão associadas aos desvios da estrutura

Compiladores

- Análise Sintática
  - A parte sintática da linguagem gera muitos nós de forma homogênea, resultando em uma árvore de sintaxe;
  - Essa árvore, genuinamente construída pelo programador, pode trazer uma série de problemas:
    - Muita informação redundante (ambiguidades)
    - Muitos separadores, terminadores, etc de forma desnecessária
    - É preciso um mecanismo que possa "enxugar", "compactar", toda a regra sintática de tal forma que facilite a vida do compilador
    - O mecanismo é a árvore de sintaxe abstrata (AST)
    - O analisador sintático que usa uma AST é conhecido por Parser

Compiladores

- Análise Léxica x Análise Sintática
  - Tem papeis parecidos, já que as duas tratam sequências de símbolos
  - O escopo das duas são separadas por questão de simplicidade e separação de responsabilidades do compilador

Compiladores

- Análise Semântica
  - Verificar se as construções identificadas pela análise sintática estão de acordo com as "regras semânticas" da linguagem.
  - Por exemplo, por ser fortemente tipada, a linguagem java exige que todas as variáveis devem ser declaradas antes de seu uso.
  - É função da análise semântica verificar se as variáveis foram devidamente declaradas.

Compiladores

- Análise Semântica
  - Muitos erros no programa não podem ser detectados sintaticamente, pois precisam de um contexto
    - Exemplo: nomes de variáveis x escopo da variável
  - Diferenciar variáveis x métodos
  - Possibilitar recursos da linguagem (Herança)
    - Atributo herdado e atributo sobrescrito
    - Métodos herdados e métodos sobrescritos
    - Métodos sobrecarregados

Compiladores

- O que é uma gramática?
  - Ferramenta para descrever uma linguagem e seus símbolos, ou seja, a definição dos analisadores léxicos e sintáticos

Compilação x Interpretação

Compiladores x Interpretadores

- O Interpretador traduz um código de alto nível para uma linguagem intermediaria, que não é de alto nível mas também não é de máquina, e aguarda alguma solicitação de chamada das instruções traduzidas.

- A medida que as chamadas ocorrem, o interpretador vai traduzindo a linguagem intermediaria para linguagem de máquina a media que o fragmento é solicitado.

- Já o compilador traduz tudo de uma vez para código de máquina. Por isso que, em geral, o tempo de compilação é mais longo, porém, o tempo de execução de um código compilado é menor do que do código interpretado.

Compiladores x Interpretadores

- Ou seja, na interpretação, por exemplo um arquivo HTML, é possível fazer modificações sem precisar compilar toda a aplicação. Geralmente são múltiplos arquivos fragmentados com responsabilidades distintas.

- Já um código compilado resulta em um único executável, qualquer mudança em uma parte do código precisa compilar todo o código fonte.

- Cada um tem suas vantagens e desvantagens

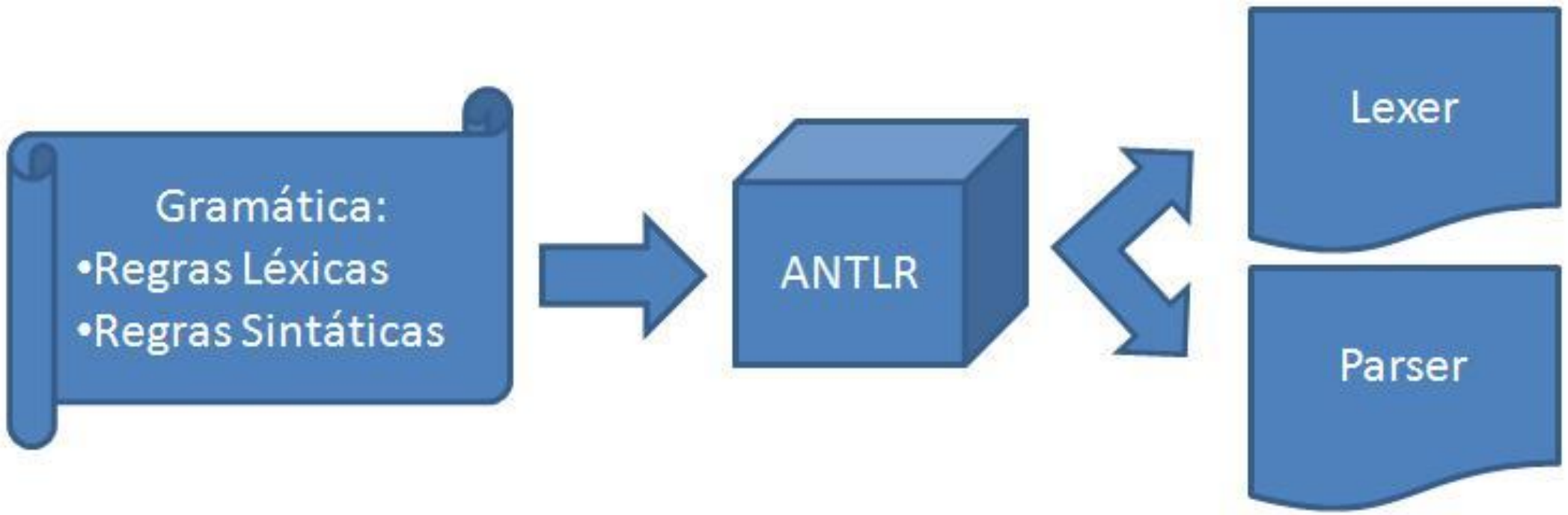
ANTLR

ANTLR

- **AN**other **T**ool for **L**anguage **R**ecognition
- Processo de construção de uma linguagem de programação
- Possibilita o desenvolvimento de compiladores, interpretadores, e afins
- Com ele você constrói sua própria gramática, ou seja, define seu analisador léxico e sintático
- Desenvolvida por Terrence Parr

Funcionamento

- Através da definição de uma gramática, o ANTLR fica responsável por gerar o analisador léxico (Lexer) e o analisador Sintático (Parser)

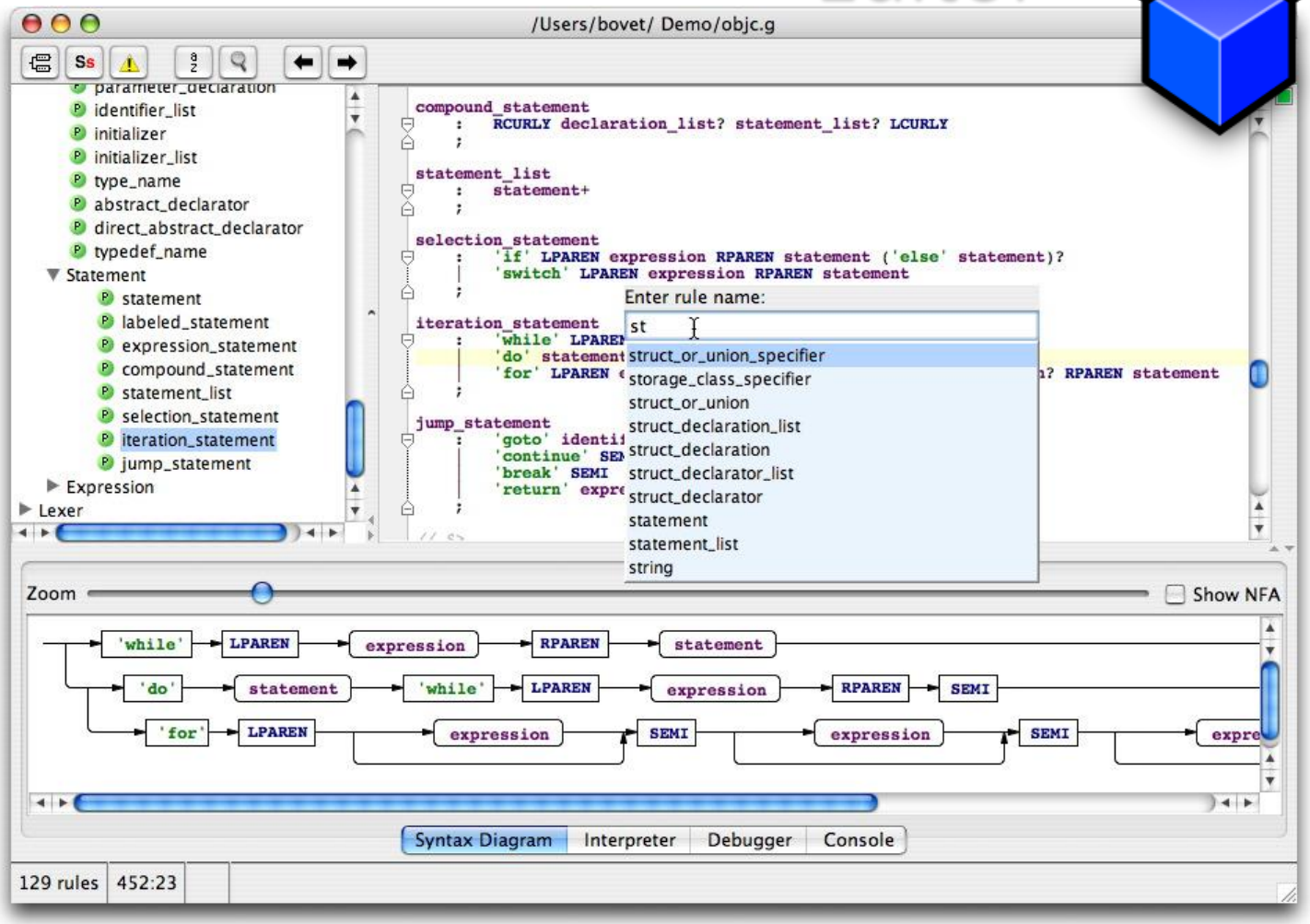


Por que usar o ANTLR?

- Tem uma IDE própria (ANTLRWORKS)
- Plugin no Eclipse
- Ferramenta bem consolidada e leve
- Da suporte a diversas linguagens como C/C++, C#, Java, Phyton, etc;

Por que usar o ANTLR?

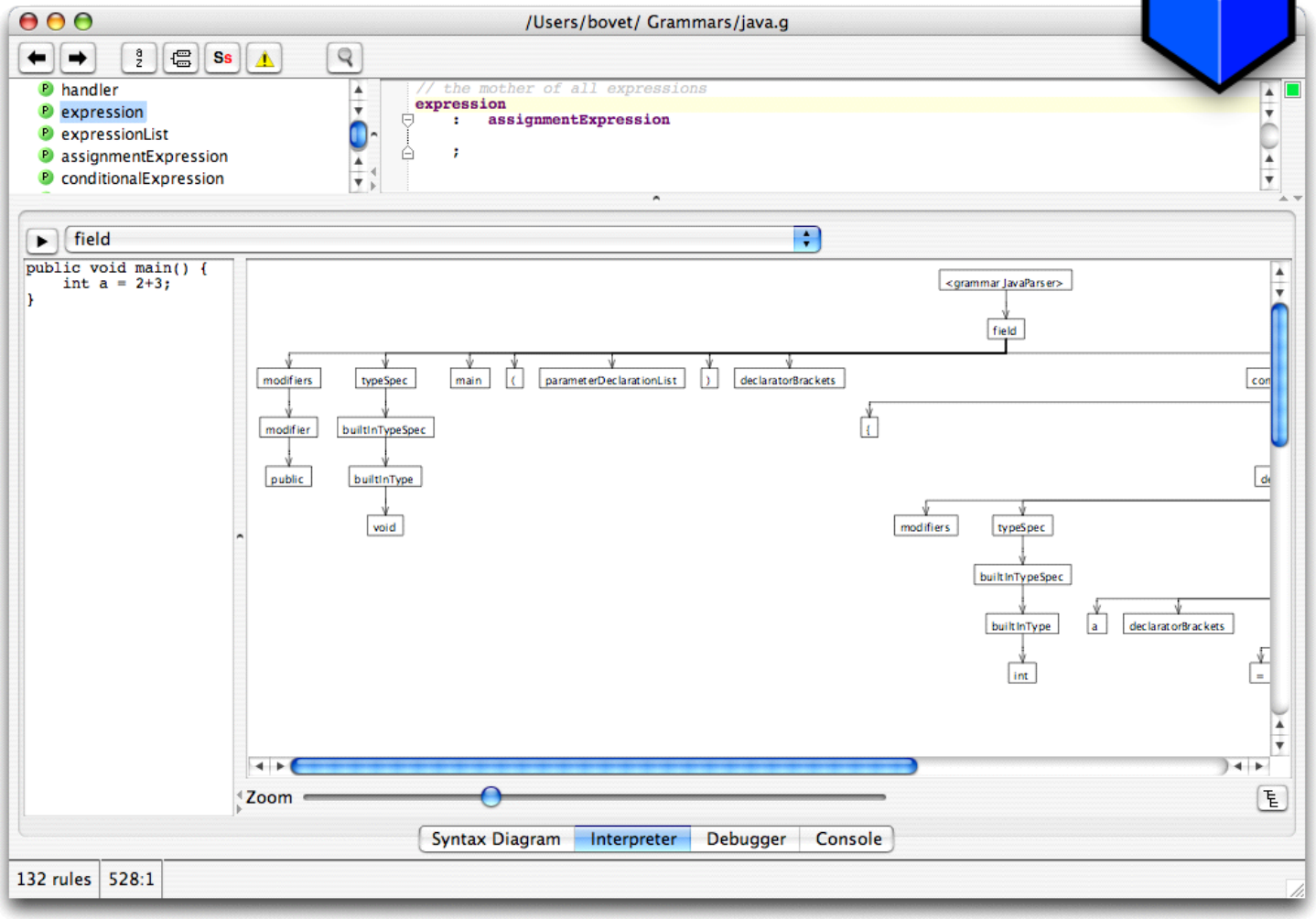
ANTLR Editor



Por que usar o ANTLR?

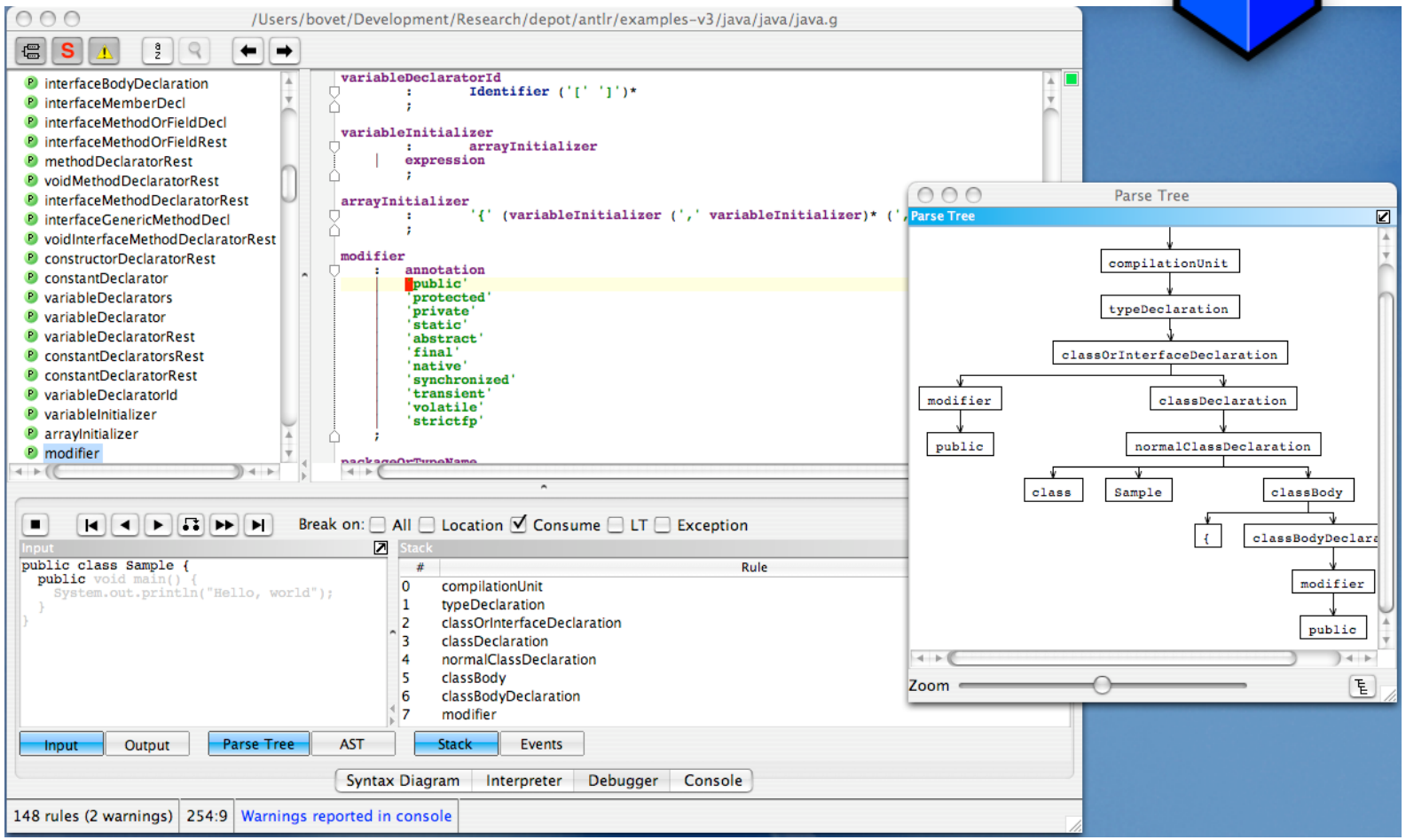
ANTLR

Interpretador



Por que usar o ANTLR?

ANTLR
Debugger



Vamos construir nossa primeira DSL

DSL

- **Uma DSL**
  - **D**omain **S**pecific **L**anguage (Linguagem específica de domínio)
  - Usa a abstração para aumentar o entendimento de domínio
  - Ou seja, estritamente relacionada com um domínio de aplicação
  - Muito poderosa para o domínio em questão, muito provavelmente inútil quando fora do seu respectivo domínio
  - Exemplos: SQL, HTML

- **DSL é diferente de uma GPL**
  - General Propose Languagem (Linguagens para propósitos gerais)
  - Java, C#, etc

Let’s Code!

Grato pela atenção

Disciplina: Paradigmas de Linguagens de Programação (PLP)

Professor: Carlo Marcelo Revoredo da Silva

Contato: revoredo@gmail.com

UNIVERSIDADE DE PERNAMBUCO