Universidade Federal de Pelotas

# Tradução Dirigida Por Sintaxe e Javacc

André Rauber Du Bois
*dubois@inf.ufpel.edu.br*

## TRADUÇÃO DIRIGIDA POR SINTAXE: JAVACC

- Gramática: Não terminais da gramática viram funções
- Tradução Dirigida por Sintaxe: Ações semânticas são adiconadas às regras da gramática
- Podemos executar qualquer código Java dentro dessas funções, basta colocar o código entre chaves
- As funções (regras) podem receber argumentos e retornar valores

# ÁRVORE SINTÁTICA

- Precisamos definir um conjunto de classes que represente um programa na linguagem
- Geralmente, não terminais viram classes
- Se um não terminal possuí possuí várias ALTERNATIVAS gerando de estruturas diferentes, podemos trabalhar com hierarquia de classes
- Podemos usar coleções de objetos, por exemplo, `ArrayList`, para guardar sequências
- A parte inicial do javacc (onde fica o `main`), é um programa Java normal, podemos ali definir todas as classes que precisamos, ex, classes da árvore sintátca

# DEFININDO A ÁRVORE SINTÁTICA

```
SIMPLE -> "main" "{" COMANDOS  "}"
COMANDOS -> COMANDOS  COM ";" | COM ";"
COM -> id ":=" EXP | "print" "(" EXP ")"
EXP -> num | id
```

# EXEMPLO EM SIMPLE

```
main{
   x:=4;
   y:=x;
    print(x);
    print(y);
    print(9);
}
```

# SIMPLE NO JAVACC

```
PARSER_BEGIN(Simple)
import java.io.*;
public class Simple {

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

    Simple parser = new Simple(new FileInputStream(args[0]));
    parser.Simple();
  }

}

PARSER_END(Simple)
```

# SIMPLE NO JAVACC

```
public class Simple {

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

        Simple parser = new Simple(new FileInputStream(args
            [0]));
        SimpleA arvore = parser.Simple();
        System.out.println(gera_java(arvore));

 }
```

# ÁRVORE SINTÁTICA SIMPLE

```
//SIMPLE -> "main" "{" COMANDOS "}"
//COMANDOS -> COM ";" COMANDOS'
//COMANDOS'->   COM ";" COMANDOS' | epsilon

class SimpleA{
    ArrayList<Comando> comandos;

    SimpleA(ArrayList comandos)
    {this.comandos=comandos;}
}
```

# ÁRVORE SINTÁTICA

- Eliminando recursão à esquerda

```
COMANDOS -> COMANDOS  COM ";" | COM ";"
```

- Eliminando:

```
COMANDOS -> COM ";" COMANDOS'
COMANDOS'->   COM ";" COMANDOS' | epsilon
```

- Elimnando 2:

```
COMANDOS -> (COM ";")+
```

# ÁRVORE SINTÁTICA

- Eliminando recursão à esquerda

```
A -> Ab|c
```

- Eliminando:

```
A-> cb*
```

- Recursão:

```
A -> Ab | vazio
```

- Eliminando:

```
A -> b*
```

# ÁRVORE SINTÁTICA SIMPLE

```
//COM -> id ":=" EXP | "print" "(" EXP ")"

class Comando{}

class Atrib extends Comando{

    String id;
    Exp exp;

    Atrib (String id, Exp exp)
    {
     this.id=id;
     this.exp=exp;
    }
}
```

# ÁRVORE SINTÁTICA SIMPLE

```
//  COM -> ... | print ( EXP )
class Print extends Comando{

   Exp exp;

   Print(Exp exp)
   {this.exp = exp;}
}
```

# ÁRVORE SINTÁTICA SIMPLE

```
//   EXP -> num | id

class Exp{}
class Num extends Exp{
   int num;

   Num(int num)
   {this.num = num;}
}
class Var extends Exp{
   String var;

   Var(String var)
   {this.var=var;}
}
```

# GERANDO A ÁRVORE SINTÁTICA

```
// SIMPLE -> "main" "{" COMANDOS "}"

void Simple () :
{}
{

<MAIN> <ACHAVES> Comandos() <FCHAVES>
<EOF>

}
```

# GERANDO A ÁRVORE SINTÁTICA

```
// SIMPLE -> "main" "{" COMANDOS "}"

SimpleA Simple () :
{ArrayList comandos= new ArrayList();}
{

<MAIN> <ACHAVES> Comandos(comandos) <FCHAVES>
<EOF>
 {return new SimpleA(comandos);}
}
```

# SIMPLE NO JAVACC

```
// COMANDOS -> (COM ";")+

void Comandos (ArrayList comandos) :
{Comando c;}
{

  (c=Com() {comandos.add(c);} <PV>)+

}
```

# SIMPLE NO JAVACC

```
// COM -> id ":=" EXP | "print" "(" EXP ")"

void Com () :
{}
{

  (<ID> <ATRIB> Exp())
  | (<PRINT> <APARENTESES> Exp() <FPARENTESES>)

}
```

# SIMPLE NO JAVACC

```
// COM -> id ":=" EXP | "print" "(" EXP ")"

Comando Com () :
{Token id=null; Exp e=null; Comando c=null; Comando result=
    null;}
{

  (

  (id=<ID> <ATRIB> e=Exp() {result = new Atrib(id.image,e);})
  | (<PRINT> <APARENTESES> e=Exp() <FPARENTESES> {result =
      new Print(e);})

  )

  {return result;}
}
```

# SIMPLE NO JAVACC

```
// EXP -> num | id

void Exp () :
{}
{
    <NUM> | <ID>
}
```

# SIMPLE NO JAVACC

```
// EXP -> num | id

Exp Exp () :
{Token t = null; Exp result=null;}
{
    (

     (t=<NUM> {result = new Num(Integer.parseInt(t.image));})
     | (t=<ID>  {result = new Var(t.image);})

    )

     {return result;}

}
```

# SIMPLE NO JAVACC

```
public class Simple {

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

        Simple parser = new Simple(new FileInputStream(args
            [0]));
        SimpleA arvore = parser.Simple();
        System.out.println(gera_java(arvore));

 }
```

## COMENTÁRIOS FINAIS

- Método gera_java: a partir da árvores sintática gera uma String contendo código Java
- toString: descreve como gerar Strings a partir de um objeto
- Definir toString que converte aquela parte do código
- Debug: javacc gera código java
- Se a gramática e tradução possui algum erro, quando for compilado o código java, pode gerar uma mensagem de erro no código gerado
- Voltar na gramática, encontrar o erro, compilar novamente com o javacc