# **Running Javascal**

Before running Javascal ensure to have compiled Javascal:

- following the Getting Started With Javascal Developers Guide or
- downloding the compiled version from javascal sourceforge download page.

1 - Open Command Prompt and go into Javascal\bin folder

2 – Compile a sample pascal application typing: Javascal.bat ..\Examples\Back.pas

C:\WINDOWS\system32\cmd.exe	. 🗆 🗙
Microsoft Windows XP [Versione 5.1.2600] (C) Copyright 1985-2001 Microsoft Corp.	-
C:\Documents and Settings\Marzio>cd workspace	
C:\Documents and Settings\Marzio\workspace>cd Javascal	
C:\Documents and Settings\Marzio\workspace\Javascal>cd bin	
C:\Documents and Settings\Marzio\workspace\Javascal\bin>Javascal.bat\Examp \Back.pas	les
C:\Documents and Settings\Marzio\workspace\Javascal\bin>set CLASSPATH=;.\;\ \Javacup;\lib\jasmin\classes	lib
C:\Documents and Settings\Marzio\workspace\Javascal\bin>java Javascal\Examp s\Back.pas	ple
s vack.pas Generated: C:\Documents and Settings\Marzio\workspace\Javascal\Examples\back\] k.class	bac
source: C:\Documents and Settings\Marzio\workspace\Javascal\bin\BackEnd\Input. ass	.cl
ass dest: C:\Documents and Settings\Marzio\workspace\Javascal\Examples\back\BackEr Input.class	nd∖
source: C:\Documents and Settings\Marzio\workspace\Javascal\bin\BackEnd\Conta: r.class	ine
dest: C:\Documents and Settings\Marzio\workspace\Javascal\Examples\back\BackEr Container.class	nd∖
source: C:\Documents and Settings\Marzio\workspace\Javascal\bin\BackEnd\MathFu t.class	սոշ
dest: C:\Documents and Settings\Marzio\workspace\Javascal\Examples\back\BackEr MathFunct.class	nd∖
Generated: C:\Documents and Settings\Marzio\workspace\Javascal\Examples\back\] cedure\$interface.class	Pro
Generated: C:\Documents and Settings\Marzio\workspace\Javascal\Examples\back\a .class	a\$1
Generated: C:\Documents and Settings\Marzio\workspace\Javascal\Examples\back\J .class	b\$2
Remember: for execute your program go to C:\Documents and Settings\Marzio\workspace\Java al\Examples\back and type: java back In bin/Log you will find log files. Type seeTree.sh for viewing AST Compiled in 1172 msec	asc
C:\Documents and Settings\Marzio\workspace\Javascal\bin>	
or bootmones and occerngs while to workspace to avascar thrin	
	-

3 – Now you can run the compiled ByteCode:

C:\Documents and Settings\Marzio\workspace\Javascal\Examples\back>java back prova 9

C:\Documents and Settings\Marzio\workspace\Javascal\Examples\back>

# See LOG produced by Javascal

Every time you compile a Pascal source file Javascal creates a lot of useful information. In particular you can see:

- The syntax tokens recognized by the scanner
- The semantic phrases recognized by the parser
- The symbol table produced during the parser
- The abstract syntax tree produced after running assembler
- The **assembly** files produces by the assembler.

#### Syntax tokens

#### You can see the syntax tokens opening the file **bin\LOG\pascalToken.log.html**

	np iou vuo cui ioi	in\LOG\pascalToken.log.html - Windows In	ernet Explorer		_ 0
0	👻 🧭 D:\Temp	\Javascal\bin\LOG\pascalToken.log.html		💌 🗲 🗙 🗛 Search	
Ŧ		🔍 Search 🔻 🔣 Imag	is 📑 Weather 🎑 News 🔹 🌽 Highlight 🔑 Resize 🔞 Pop-up Blo	ker	
¢	88 - <b>6</b> D:\T	emp\Javascal\bin\LOG\p	LOG\s 🔏 D:\Temp\Javascal\bin\LO 🗙	🟠 🔹 🔝 🐇 🖶 Pagina 🗣	💮 Strumenti
			Report from Scanner wo	rk's	
NE	LEXER	REGULAR EXPRESSION	TOKEN		
	program	program	PROGRAM		
	Back	{LETTER} ({LETTER}+{DIGIT})			
	;	;	DOTCOMMA and line number		
	var	var	VAR		
-	x	{LETTER} ({LETTER}+{DIGIT})	(A) 2772 ()		
		:	TWODOT and line number		
	integer	<pre>. {LETTER} ({LETTER}+{DIGIT})</pre>			
		·	DOTCOMMA and line number		
	, orogoduro	, procedure	PROCEDURE and line number		
	-				
- 6	d	{LETTER} ({LETTER}+{DIGIT})			
- 9	; 		DOTCOMMA and line number		
	var	var	VAR		
	Y	{LETTER} ({LETTER}+{DIGIT})			
_	-		TWODOT and line number		
-	integer	{LETTER} ({LETTER}+{DIGIT})			
_	;	;	DOTCOMMA and line number		
	procedure	procedure	PROCEDURE and line number		
	0	{LETTER} ({LETTER}+{DIGIT})			
	;	;	DOTCOMMA and line number		
-	begin	begin	BEGIN		
-	x	{LETTER} ({LETTER}+{DIGIT})	* ID and its value		
	:=	:=	ASSIGN and line number		
	x	{LETTER} ({LETTER}+{DIGIT})			
	+	"+"	PLUS and line number		
	7	{DIGIT} <sup>+</sup>	INT and its value		
	;	;	DOTCOMMA and line number		
	end	end	END		
	;	;	DOTCOMMA and line number		
]	begin	begin	BEGIN		
	У	{LETTER} ({LETTER}+{DIGIT})	* ID and its value		
	:=	:=	ASSIGN and line number		
	10	{DIGIT} <sup>+</sup>	INT and its value		
			DOTCOMMA and line number		
-	, b	<pre>/ {LETTER} ({LETTER}+{DIGIT})</pre>			
	(	(DETTERS ((DETTERS ( (DIGIT))	OPEN ROUND and line number		
2			CLOSED BOUND	Screen captu	

## Semantic phrases

#### You can see the syntax recognized by the parser opening the file **bin\LOG\parser.log.html**

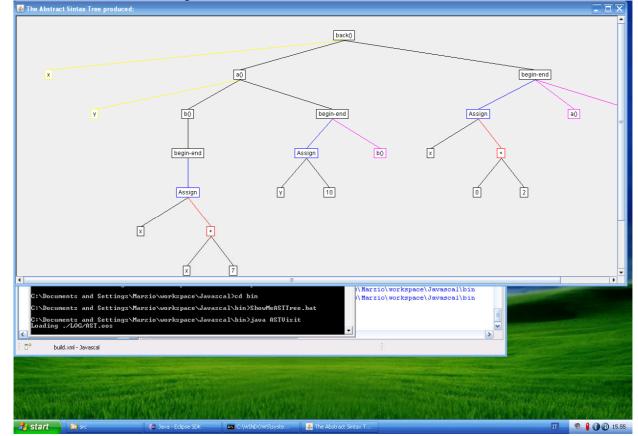
🖉 D: \Tem	np\Javascal\bin\LOG\parser.log.html - Windows Internet Explorer		_ 2 🛛
00	▼ 🌈 D:\Temp\Javascal\bin\LOG\parser.log.html	🖌 🛃 🗙 🗛 Search	P -
∕- [	🔍 Search 🔻 📓 Images 🔤 Weather 📓 News 👻 💋 Highlight 🔑 Resize 🔯 Pop-up Blocker		
🚖 🍄	😢 🔻 🍘 D:\Temp\Javascal\bin\LOG\p 🍘 D:\Temp\Javascal\bin\LOG\s 🍘 D:\Temp\Javascal\bin\LOG\p 🖗 D:\Temp\Javascal\bin\LOG\p	🏠 🔹 🔝 🔹 🍓 🔹 🔂 Pagin	a 🔹 🚫 Strumenti 👻 🎽
	Depart from Depains working		^
	Report from Parsing work's		
S	tart analyzing program back that have not parameters		
	o constant definition found		
S	tartyng type analize		
N	o type definitions found		
т	ype integer found in standardPascalType SymbolTable		
х	, declared as integer		
v	ariables declaration have been analyzed		
	Analyzing a procedure		
	No constant definition found		
	Startyng type analize		
	No type definitions found		
	Type integer found in standardPascalType SymbolTable		
	y, declared as integer		
	Variables declaration have been analyzed		
	Analyzing b procedure		
	No constant definition found		
	Startyng type analize		
	No type definitions found		
	No variable declaration found		
	analyzing assignement at line 8		
	analyzing assignement at line 11		
	Analvzing call to b()	Screen ca	pture 💌
		🕄 Risorse del computer	🔍 100% 🔻

## Symbol table

You can see the Symbol table opening the file bin\LOG\ symbolTable.log.html

	C D:\Temp\Javascal\bin\	LOG\symbolTable.log.html - Windows Int	rnet Explore		
<form><section-header><section-header><form></form></section-header></section-header></form>	🔄 🕞 👻 🌈 D:\Temp\Ja	vascal\bin\LOG\symbolTable.log.html		Ask Search	<b>P</b> -
<section-header><section-header><section-header><section-header><section-header></section-header></section-header></section-header></section-header></section-header>	<b>49</b> -	🔍 Search 🔻 🔀 Image	🔁 Weather	📓 News 🔻 🎉 Highlight 🖉 Resize 🔞 Pop-up Blocker	
Note         By hold table is an Mashtable containing masse of detrictory for more info. Nevry the table table is an Abstract class, you can see bood directory for more info. Nevry the table table is an information or procedure, so when the parse is the set with a large information current function or procedure declaration and proceedures is and when table is a Vector. This symbol table is contains all identifiers description found in current function or procedure, so when the parse is the set with a large information is procedure. The symbol table is a vector is easy undertaind how parsers uses why symbol tables. Never y symbol table is howed here is printed just before delete it.         The is the symbol table of standards types.         Symbol table for b procedure         a name: fext         balance	🔆 🍄 🔠 🕶 🏉 D:\Tem	p\Javascal\bin\LOG\p	0 × 🍘 D:1	(Temp)Javascallbin)LOG(p 🍘 🔹 🔂 🕐 Bgina 🔹 🎯 Strument	d 🔹 🎇
The Sphol table is a mathatable containing masses of identifiers, found in corre program, as String and theirs Descriptor is, is such table contains all identifiers description found in current function or procedure, so when the parses tars analyzed is function or procedure (all identifiers are correctly decilared in current symbol table or in procedure, so when the parses tars analyzed is function or procedure (all identifiers are correctly decilared in current symbol table or procedure (all identifiers are correctly be procedure).         Will be the in the Weeter. Men the parser have analyzed is function or procedure (all identifiers are correctly symbol table or in procedure (all identifiers are correctly point table table or in procedure (all identifiers are correctly point in the last or in procedure (all identifiers are correctly point in the last or in procedure (all identifiers are correctly point in the last or in procedure (all identifiers are correctly point in the last or in procedure (all identifiers are correctly point in the last or in procedure (all identifiers are correctly point).         The second point of procedure is table and parser parse have analyzed is function or procedure (all identifiers are correctly point).       Integration (all identifiers are correctly point).         Symbol table for bable for bable cort bable cort procedure (all identifiers) are point point (all identifiers).       Integration (all identifiers) are point (all identifiers).         Symbol table for b procedure       Integration (all identifiers) are point).       Symbol table for a procedure.         Matrifier name possiptor Type value.       Integration (all identifiers).       Symbol table for a more in possiptor).         Symol table for a procedure       Integ		How Symbol	able	was developed during compilyng?	~
Identifier name Descriptor Type Value integer       Rename: Int         integer       Rename: Int         char       Rename: Text         real       Rename: Real         bolean       Procedure         a()       Proceedure         y       Var declared as Rename: Int         Symboltable forhackprogram:       Identifier name         da()       Procedure         x       Var declared as Rename: Int	The Symbol table i Descriptor is an a adds a new symbol starts analyzing th for them in the V analyzed. This resc and program source	bstract class, you can see Doc table to a Vector. This symbol e function or procedure it che fector. When the parser have a blve nested function or procedu is easy understand how parser u	directory table con ks if all alyzed a e and shad es symbol	for more info. Every time that the parser finds a new function or procedure declaration, is tains all identifiers description found in current function or procedure, so when the parse identifiers are correctly declared in current symbol table or in previous symbol table lookin function or procedure it deletes symble table of function or procedure that has just been lowed variables. Here is showed every symbol table that the parser have build. Witch this lo tables. Every symbol table showed here is printed just before delete it.	it er ng en
intager       Rename: Int         char       Rename: Text         real       Rename: Text         real       Rename: Real         boolean       Rename: PBoolean         Symbol table for b procedure         Identifier name       Descriptor Type Value         Symbol table for a procedure         Identifier name       Descriptor Type Value         0.0       Procedure         a(1)       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name       Descriptor Type Value         a(1)       Procedure         x       Var declared as Rename: Int	This is the sy	mbol table of standard:	types:		
char       Rename: Char         text       Rename: Text         teal       Rename: Real         boolean       Rename: Real         boolean       Rename: PBoolean         Symbol table for b procedure         Identifier name/Descriptor Type Value         0()       Procedure         a()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name       Descriptor Type         a()       Procedure         x       Var declared as Rename: Int	Identifier name	Descriptor Type Value			
text       Rename: Text         real       Rename: Real         boolean       Rename: PBoolean         Symbol table for b procedure         Identifier name       Descriptor Type Value         b()       Procedure         Symbol table for a procedure         Identifier name       Descriptor Type Value         b()       Procedure         a(1)       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:       Identifier name         Identifier name       Descriptor Type         v       Var declared as Rename: Int         symboltable forbackprogram:       Identifier name         a()       Procedure         x       Var declared as Rename: Int	integer	Rename: Int			
real       Rename: Real         boolean       Rename: PBoolean         Symbol table for b procedure         Identifier name Descriptor Type Value         b()       Procedure         identifier name Descriptor Type Value         b()       Procedure         identifier name Descriptor Type Value         b()       Procedure         identifier name Descriptor Type Value         a()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name Descriptor Type Value         a()       Procedure         x       Var declared as Rename: Int					
boolean       Rename: PBoolean         Symbol table for b procedure         identifier name bescriptor Type Value         b()       Procedure         identifier name       Descriptor Type Value         b()       Procedure         id()       Procedure         id()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name       Descriptor Type         id()       Procedure         a()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:       Identifier name         id()       Procedure         x       Var declared as Rename: Int	and an extension of the second s				
Symbol table for b procedure         Identifier name/Descriptor Type/Value         Symbol table for a procedure         Identifier name       Descriptor Type         Value         0()       Procedure         a()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name       Descriptor Type         Value         a()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name       Descriptor Type         a()       Procedure         x       Var declared as Rename: Int	1000000000				
Identifier name Descriptor Type Value         b()       Procedure         Identifier name       Descriptor Type         a()       Procedure         a()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:       Identifier name         Identifier name       Descriptor Type         Var declared as Rename: Int       Var declared as Rename: Int					
b()       Procedure         Symbol table for a procedure         identifier name       Descriptor Type         b()       Procedure         a()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name       Descriptor Type         a()       Procedure         x       Var declared as Rename: Int					
Symbol table for a procedure         Identifier name       Descriptor Type         b()       Procedure         a()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name       Descriptor Type         a()       Procedure         a()       Procedure         a()       Procedure         x       Var declared as Rename: Int					
Identifier name       Descriptor Type       Value         b()       Procedure					
b ()       Procedure         a ()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name       Descriptor Type         a ()       Procedure         a ()       Procedure         x       Var declared as Rename: Int					
a ()       Procedure         y       Var declared as Rename: Int         Symboltable forbackprogram:       Identifier name         Identifier name       Descriptor Type       Value         a ()       Procedure       int         x       Var declared as Rename: Int       int	Identifier name	Descriptor Type	Value		
y       Var declared as Rename: Int         Symboltable forbackprogram:         Identifier name       Descriptor Type         a()       Procedure         a()       Var declared as Rename: Int         x       Var declared as Rename: Int	b()	Procedure			
Symboltable forbackprogram: Identifier name Descriptor Type Value a() Procedure x Var declared as Rename: Int X Streen cepture	a()	Procedure			
Identifier name       Descriptor Type       Value         a()       Procedure	У	Var declared as Rename: 3	nt		
a ()     Procedure       x     Var declared as Rename: Int   Streen cepture	Symboltable fo	rbackprogram:			
x Var declared as Rename: Int	Identifier name	Descriptor Type	Value		
	a()	Procedure			
	x	Var declared as Rename: 3	nt		
	-				
				Screen capture	~
					-

#### Abstract syntax tree



You can see the AST running the ShowMeASTTree.bat located into Javascal\bin folder

### Assembly

You can see the assembly files taking a look into Examples folder (.asm files)

🔄 Examples				
File Modifica Visualizza Preferiti Strumenti ?				
🔇 Indietro 🔹 🕥 - 🏂 🔎 Cerca 陵 (	Cartelle 🛄 👻			
Indirizzo 🛅 C:\Documents and Settings\Marzio\workspace	e\Javascal\Examples			🔽 🄁 Vai
Cartelle ×	Nome 🔺	Dimensione	Tipo	Data ultima modif
🗉 🦳 All Users 🗛	🗋 🛅 back		Cartella di file	05/04/2009 15.3
🗆 🧰 Marzio	🖥 🖬 Back.pas	1 KB	File PAS	05/04/2009 14.5
387A0090-5AB6-4838-B146-96	🚵 backa\$1	1 KB	Assembler Source	05/04/2009 15.3
C Desktop	🚾 backa\$1.class	0 KB	File CLASS	05/04/2009 15.3
🗉 🧰 Documenti	🚈 backb\$2	1 KB	Assembler Source	05/04/2009 15.3
E C Menu Avvio	🖻 backb\$2.class	0 KB	File CLASS	05/04/2009 15.3
🗄 🧙 Preferiti	🔤 backback	2 KB	Assembler Source	05/04/2009 15.3
workspace	🖬 backback.class	0 KB	File CLASS	05/04/2009 15.3
E C .metadata	📔 🔤 backProcedure\$interface	1 KB	Assembler Source	05/04/2009 15.3
Avascal	backProcedure\$interface.class	0 KB	File CLASS	05/04/2009 15.3
isettings	🖬 Funct.pas	1 KB	File PAS	05/04/2009 14.5
🛅 bin	🖬 nested.pas	1 KB	File PAS	05/04/2009 14.5
Cini doc	Proc.pas	1 KB	File PAS	05/04/2009 14.5
	🖬 readwrite.pas	1 KB	File PAS	05/04/2009 14.5
Examples Examples back	🖬 recordArray.pas	1 KB	File PAS	05/04/2009 14.5
E Chib	🖬 simple.pas	1 KB	File PAS	05/04/2009 14.5
⊞ 🛅 src	🖬 Struct.pas	1 KB	File PAS	05/04/2009 14.5
🗉 🧰 Programmi	Test.pas	1 KB	File PAS	05/04/2009 14.5
E 🔔 Unità DVD (D:)				
<ul> <li>Grand Dyb (D.)</li> <li>Pannello di controllo</li> </ul>				
Control of the second of				
Cocumenti - Marzio				
🗊 🌠 Diamaa di waba				
	<			
Oggetti: 18 (nascosti 1) (Spazio disponibile: 53,7 GB)		6,39 k	3B 🛛 🚽 Risorse	del computer