
# USER'S MANUAL

VERSION 0.11

# INDEX OF CONTENTS

# INDEX OF FIGURES

# 1. SYSTEM REQUISITES

## 1.1. USER

*ACIDE - a Configurable IDE* does require neither special system configurations nor special system requirements because the executable file is attached in its distribution, making the execution of ACIDE - A Configurable IDE easy and comfortable to the users.

*ACIDE – A Configurable IDE* is **cross-platform** and has been tested on ***MS Windows XP/Vista/7***, ***Ubuntu Linux 10.04.1***, ***Ubuntu Linux 12.04***, and ***MacOSX Snow Leopard***. Executables for all of these operating systems are provided. The only mandatory requirement is the previous installation of the ***Java Virtual Machine (JVM)***. The user will have to get the *JRE installation file* with **1.6 and later versions**, which is available in the following link:

*http://www.java.com/es/download/manual.jsp.*

Only with this easy and fast step the user will be able to run *ACIDE - A Configurable IDE* on his computer without problems. However, in order to fully enjoying all the features of the application such as *ACIDE - A Configurable IDE grammar configurations*, two extra tools will have to be also installed: ***javac.exe*** and ***jar.exe***.

Those tools are available in the ***Java Development Kit (JDK)*** installation file, which is available in the following link:

*http://www.oracle.com/technetwork/java/javase/downloads/index.html*

At last, in order to visualize the present document, it is mandatory for the users to have previously installed any software for **PDF files visualization**.

## 1.2. DEVELOPER

For developers, it is mandatory to have previously installed the ***Java Development Kit (JDK)*** with 1.6 and later versions and any software for the edition of the source code.

The source code has been fully edited with the **Eclipse IDE** tool which is available in:

*http://www.eclipse.org/*

Furthermore, with the *ACIDE - A Configurable IDE* source code distribution, the Eclipse *project file* is available. The developer has to import the project file into Eclipse and start the edition, fast and simple.

## 1.3. EXECUTING ACIDE

User has to unpack the distribution archive file into the directory he wants to instal *ACIDE – A Configurable IDE*, which will be referred to as the distribution directory from now. Since it is a portable application, it needs to be started from its distribution directory, which means that the start-up directory of the shortcut must be the distribution directory.

To execute *ACIDE – A Configurable IDE* on the different *SOs,* user only has to run the **des_acide.jar** file to open an instance of the application preconfigured to work with *DES*. At Windows, the user only have to do double click in the file. He also can create a script or an alias for executing the file at the distribution root, typing:

```
java –jar des_acide.jar
```

or, to avoid that shell depends on executable:

```
javaw –jar des_acide.jar
```

*Linux* and *Linux* the user can create a script or an alias for executing the file **des_acide.jar** at the distribution root, typing:

```
java –jar des_acide.jar
```

# 2. INTRODUCING ACIDE

*ACIDE – A Configurable IDE* is a cross-platform, open-source Integrated Development Environment (IDE). It has been developed by different teams of students coursing *Computing Systems* and directed by Fernando Sáenz Pérez. Next, *ACIDE – A Configurable IDE* features will be further explained:

## 2.1. TECHNOLOGY

The implementation of the application has been completely done using *Java* under *Eclipse.* Version control was kindly provided by *Tortoise SVN.*

## 2.2. THE MAIN GUI

*Figure 1* shows the main GUI of ACIDE. It consits of four main panels. The left panel shows the organistation of the current project, the MDI windows in the right are the opened files which may belong to the project (files may be opened without assigning them to the project). Below, the left panel shows the databases system connected with ACIDE, which allows user interaction. Beside, the shell panel is shown. The case shown is the DES console. The databases, shell and project panels can be hidden. Moreover, there is no need to work with projects if this flexibility is not needed;a regular user may use the system as is. The status of the GUI is remembered for the next time the tool is executed. If the tool opens a project, its status when it was last saved is restored.

The menu bar includes some common entries:

- **File**: For file related operations.
- **Edit**: For clipboard related operations, *Search*, *Replace*, *Undo, Redo, Select All* and *Go To Line*.
- **Project**: For project related operations.
- **View**: For showing/hiding project, shell and databases panels, and displaying the log. Window arrangements are not possible up to now, but

usual features are cascading and tiling windows both vertically and horizontally.

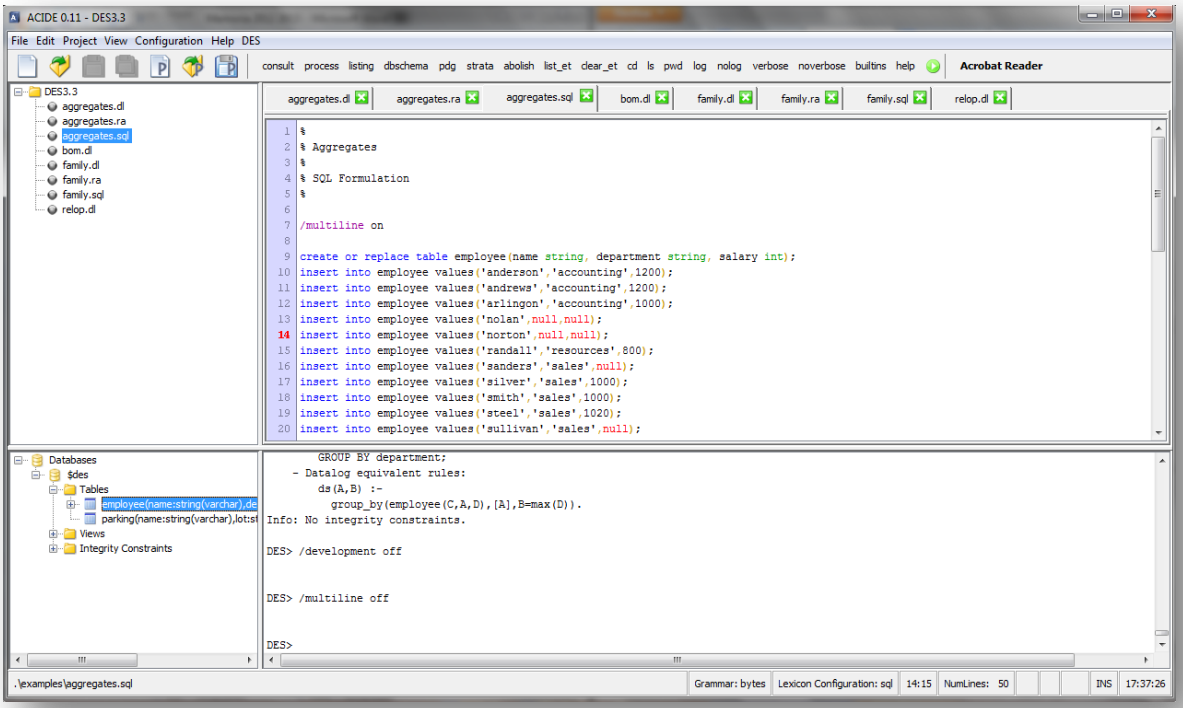



**Figure 1: ACIDE Main GUI**

- **Configuration:** This entry allows to configure *Lexicon (*for syntax highlight ing.), *Grammar* (for parsing on the fly), *Compiler* (for compiling the project), *File Editor (*for changing the display and behavior of the editors), *Shell* (the shell in the right bottom panel), *Database Panel*( the database panel in the left bottom panel), *Language* of the GUI*, Menu* (for the configuration of the menu bar) and *Toolbar* for the commands, which can be displayed either as icons or textual descriptions. Tooltips for toolbar commands can be configured.
- **Help:** This entry contains *Show Help* and *About ACIDE*.

In addition, there is a fixed toolbar, which includes common buttons for the file and project related basic operations: *New, Open, Save* and *Save All* (this last one only for files). Next to the fixed toolbar, there is the configurable toolbar.

Finally, the status bar gives information about some items: The complete path of the selected file the selected grammar and lexicon, the line and column numbers, Caps Lock, Scroll Lock, Num Lock and current time.

All these components will be further explained throughout this document.

## 2.3. PROJECTS

A project contains the whole status of a session, which is defined by all the possible configurations as well as the current display status. It consists of files arranged in folders (with any tree depth), all the configurations for the session (lexicon, grammar, compiler, shell, language, file editor, menu and toolbar), main GUI arragement (panel sizes, and opened files in the project), and file attributes. File attributes identify a file in a project as compilable and/or main. If a file is compilable, then the compiler configuration can be set to compile each of these files. If a file is a main file (there is only one in the project), then it can be used in the compiler configuration or in the toolbar commands.

The project structure shown in folders is a logical view which may coincide with the physical structure of the *OS* folders, but this is not needed. User can include in a given project a file belonging to another tree structure, therefore allowing to share files for different projects.

## 2.4. CONFIGURATION

The main objetive of this system is to be as highly configurable as possible, keeping the configurations easy and portable by means of text files. The user can configure the *Lexicon, Grammar, Compiler, File Editor, Shell, Database Panel, Language, Menu* and *Toolbar.* These configurations will be further explained throughout this document.

# 3. MENU BAR

Next we further detail each one of the submenus that *ACIDE – A Configurable IDE* as default. As is explained in *Chapter 3.5.8* the user can insert new menu submenus and items. In *Chapter 3.7* and *Chapter 0* we explain how to use these objects. We also explain how to configure this menu externally with *XML* files in *Chapter 13.3.1.*

## 3.1. FILE MENU

It contains the following menu items for the files management:

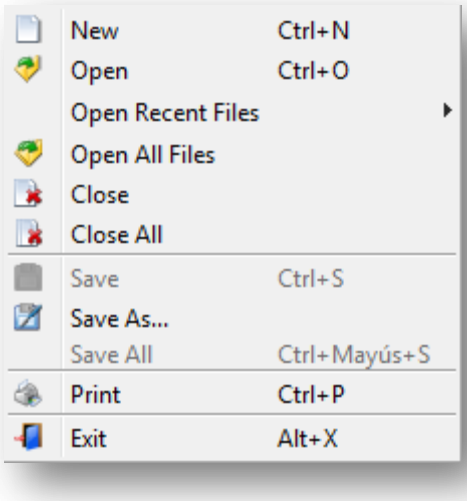



**Figure 2: File Menu**

Next, all the previous menu items will be further explained:

### 3.1.1. NEW

Creates a new empty file in the file editor.

### 3.1.2. OPEN

Open a previously saved file into the file editor.

### 3.1.3. OPEN RECENT FILES

Displays a list which contains all the files that have been opened previously in the file editor and the option to set the list to empty.

### 3.1.4. Open all files

Open all the files associated to the current project in the file editor.

### 3.1.5. Close file

Close the active file in the file editor, asking to the user if he wants to save it if the file was previously modified.

### 3.1.6. Close all files

Close all the opened files in the file editor, asking to the user if he wants to save them if the files were previously modified.

### 3.1.7. Save file

Save the active file in the file editor at the same path that it was previously saved.

### 3.1.8. Save file as

Save the active file in the file editor into a different path than it was saved before.

### 3.1.9. Save all files

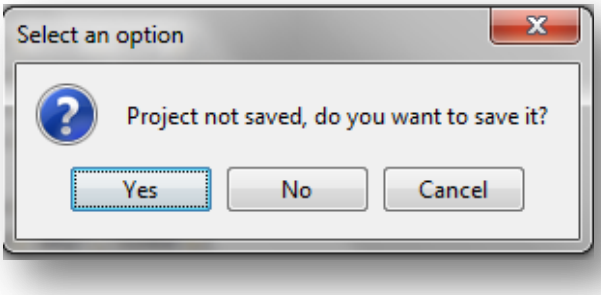
Save all files opened in the file editor.

### 3.1.10. Print file

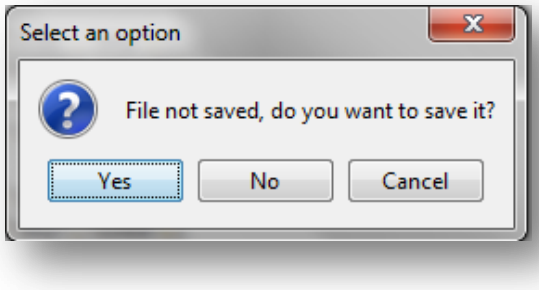
Prints the active file in the file editor.

### 3.1.11. Exit

Closes the application and if any changes have been encountered in the current project configuration, displays the following dialog to the user:



Adittionally, if any of the opened files in the file editor has been modified, it will asked to the user for saving them with the following dialog:




The user can abort the exit process in any time by cancelling any of the previous dialogs.

## 3.2. EDIT MENU

It contains the following menu items for the common file editor management:

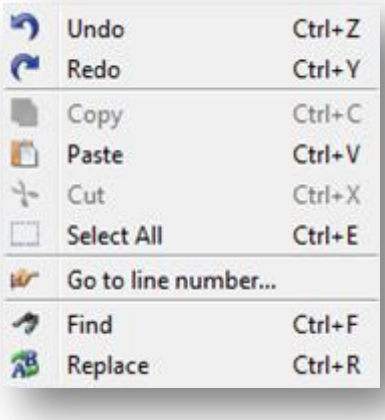



**Figure 3: Edit Menu**

Next, all the previous menu items will be further explained:

### 3.2.1. UNDO

Undo the changes in the file editor setting the focus on the file which is the owner of the change.

### 3.2.2. REDO

Redo the changes in the file editor setting the focus on the file that is the owner of the change.

### 3.2.3. COPY

Copy the selected text in the active file in the file editor or in the console and put it into the system clipboard.

### 3.2.4. PASTE

Paste the text stored in the system clipboard in the current position of the active file in the file editor or in the console.

### 3.2.5. CUT

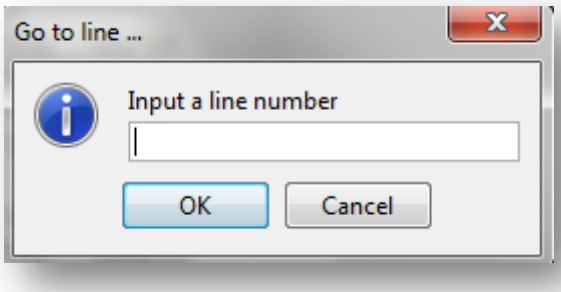
Cut the selected text in the active file in the file editor or in the console and put it into the system clipboard.

### 3.2.6. SELECT ALL

Selects all the content of the active file in the file editor.

### 3.2.7. GO TO LINE

It displays a dialog in which the user will type down the number of the line where he wants to place the caret cursor in the active file in the file editor:




### 3.2.8. SEARCH

Shows the search text window of the file editor:

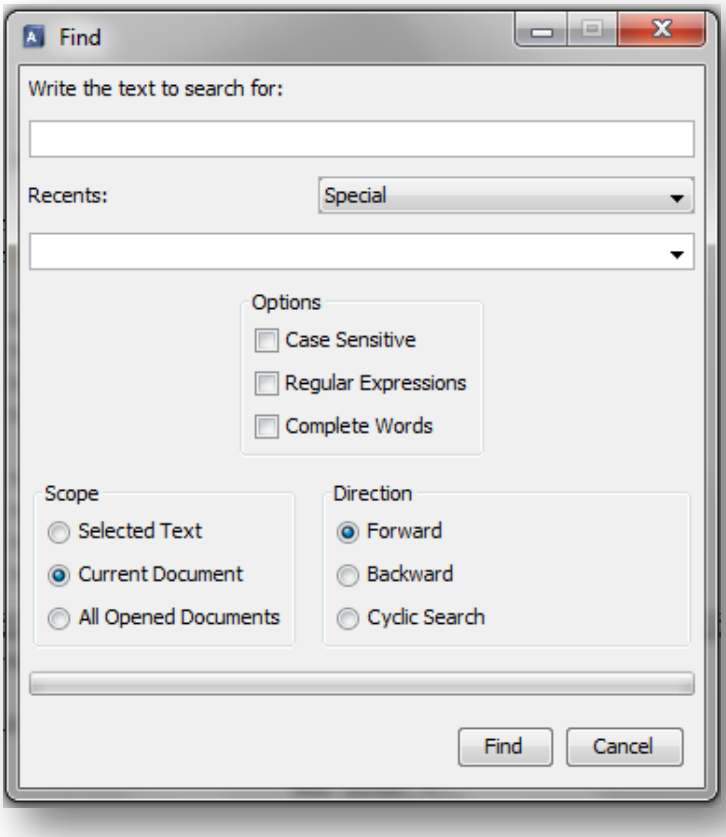



**Figure 4: Search window**

Then we proceed to describe each component of the window:

- **Text box:** Here is where user enters the search text.
- **Special:** You can search for paragraph breaks and tabs by the special marks ^p and ^t. These special marks can be written in the text box or selected in the Special combo menu.
- **Recents:** This combo menu displays a list which contains all the recent searches that have been executed before. When user selects one, this appears in the Text box.
- **Options:**
    - **Case sensitive:** this option is used to search for strings without having or taking into account the Upper / Lowercase.
    - **Regular expressions:** regular expressions search associated with a search pattern. More information about Regular Expressions in *Chapter 14*.
    - **Whole words:** find whole words only.
- **Scope:**
    - **Selected text:** search within a selected text.
    - **Current document:** document-search starting in a certain position of the active file of File Editor.
    - **All opened documents:** searches in all opened files on the file editor.
- **Direction:**
    - **Forward:** searches from the current caret position to the end of the file in the source file editor.
    - **Backward:** searches from the current caret position to the beginning of the file in the source file editor.
    - **Cyclic:** searches from the current caret position to the end of the file in the source file editor, and star from the beginning until the starting position.
- **Progress bar:** shows the progress of the active search.

### 3.2.9. REPLACE

Displays the replace text window on the file editor:

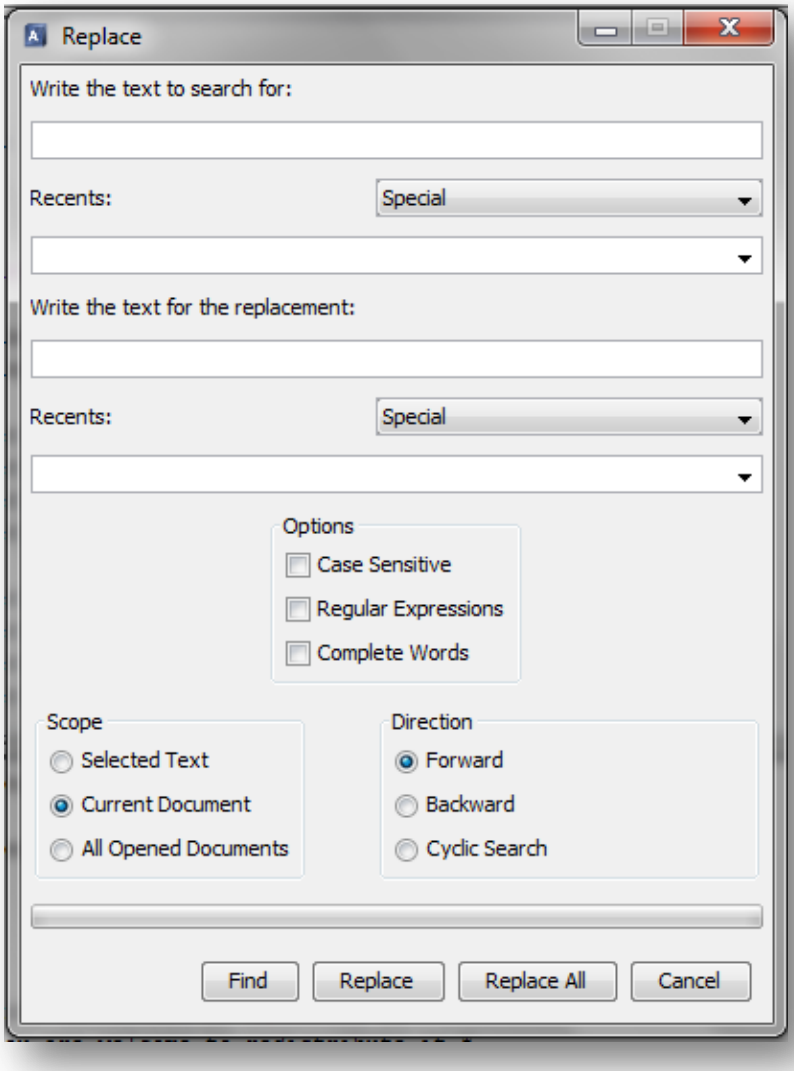



**Figure 5: Replace window**

It offers the same options than the search window and also the **replace buttons** and the **replace text field** to select the text to use for the replacements:

- **Search text box:** Here is where user enters the search text.
- **Special:** You can search for paragraph breaks and tabs by the special marks ^p and ^t. These special marks can be written in the text box or selected in the Special combo menu.
- **Recents searches :** This combo menu displays a list which contains all the recent searches that have been executed before. When user selects one, this appears in the Text box.
- **Replace text box:** Here is where user enters the replace text.

- **Special:** You can replace with paragraph breaks and tabs by the special marks ^p and ^t. These special marks can be written in the text box or selected in the Special combo menu.
- **Recents replaces :** This combo menu displays a list which contains all the recent replacements that have been executed before. When user selects one, this appears in the replaces Text box.
- **Options:**
    - **Case sensitive:** this option is used to search for and replace strings without having or taking into account the Upper / Lowercase.
    - **Regular expressions:** regular expressions search associated with a search pattern. More information about Regular Expressions in *Chapter 14*.
    - **Whole words:** find whole words only.
- **Scope:**
    - **Selected text:** search within a selected text.
    - **Current document:** document-search starting in a certain position of the active file of File Editor.
    - **All opened documents:** searches in all opened files on the file editor.
- **Direction:**
    - **Forward:** searches from the current caret position to the end of the file in the source file editor.
    - **Backward:** searches from the current caret position to the beginning of the file in the source file editor.
    - **Cyclic:** searches from the current caret position to the end of the file in the source file editor, and star from the beginning until the starting position.
- **Progress bar:** shows the progress of the active search or replacement.

When a general replacement is performed, it displays the following dialog to the user informing of the *number of replacements:*

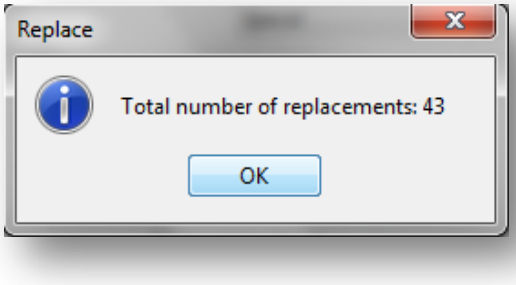



**Figure 6: Number of replacements**

## 3.3. PROJECT MENU

It contains the menu items required for the project configurations management:

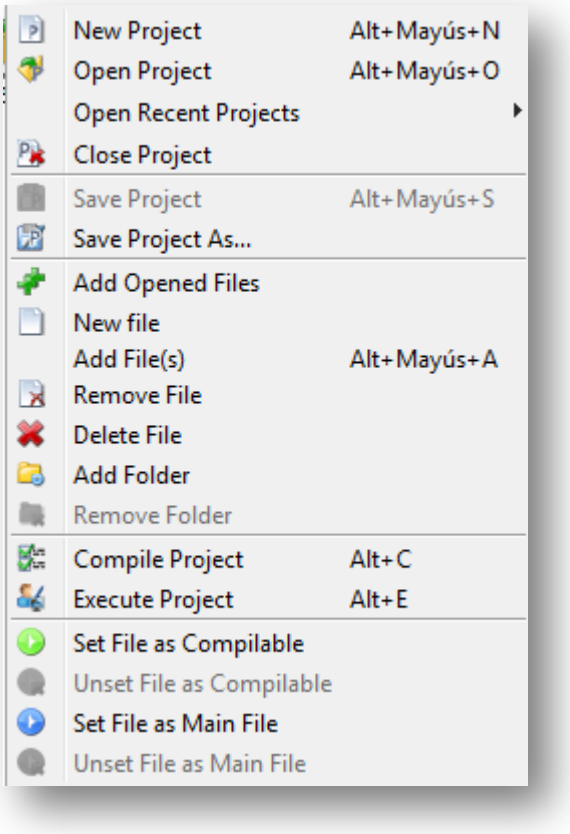



**Figure 7: Project menu**

Next, all the previous menu items will be further explained:

### 3.3.1. NEW PROJECT

Configures a new project displaying the following configuration window:

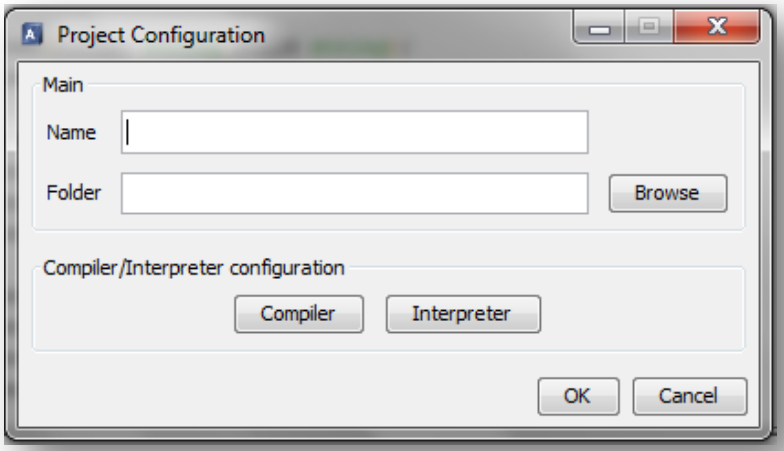
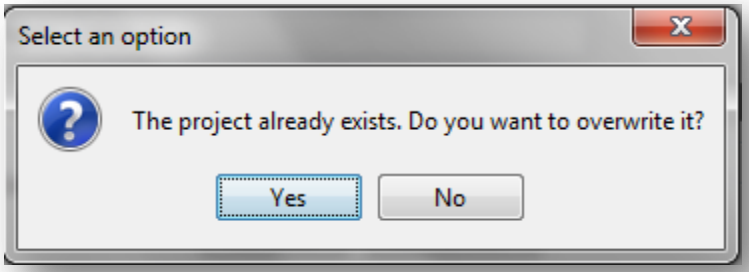



**Figure 8: Project configuration**

Next, the window options are further described:

- **Name:** indicates the project name.
- **Folder:** indicates the folder where the project file will be placed. If the project file already exists in the folder the application will give the chance to the user to overwrite it or not:




- **Compiler/Interpreter options**
  - **Compiler:** selects the compiler configuration for the new project.
  - **Interpreter:** selects the console panel configuratoin for the new project.

### 3.3.2. OPEN PROJECT

Open an existing project.

### 3.3.3. Open recent projects

Displays a list with the projects that have been already opened in the application and the option to set the list to empty.

### 3.3.4. Close project

Closes the current project and sets the default configuration.

### 3.3.5. Save project

Saves the current project configuration into its configuration file.

### 3.3.6. Save project as

Saves the current project configuration into a different configuration file.

### 3.3.7. New project file

Creates a new empty file in the file editor and adds it to the current project configuration after asking to the user for its final destination.

### 3.3.8. Add all opened files

Adds all the opened files in the file editor to the current project configuration. Files that already belong to the project will not be included again.

### 3.3.9. Add file

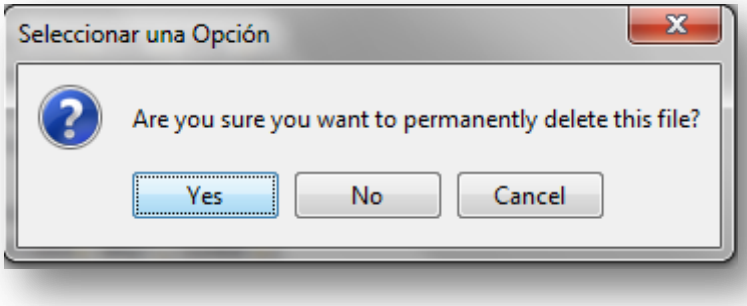
Adds the active file in the file editor to the project configuration.

### 3.3.10. Remove file

Removes the file from the project configuration but does not deletes it from disk.

### 3.3.11. Delete file

Removes the file from both project configuration and disk previous user confirmation:




### 3.3.12. ADD FOLDER

Adds a new folder to the project in the selected level at the explorer tree, and checks that it does not exist another folder with the same name before adding it:

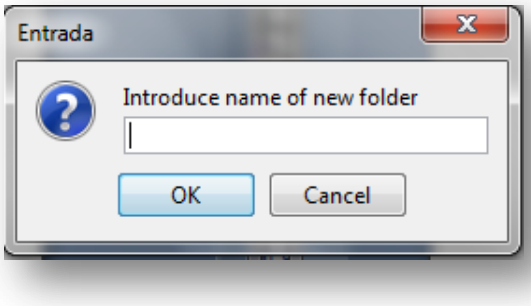
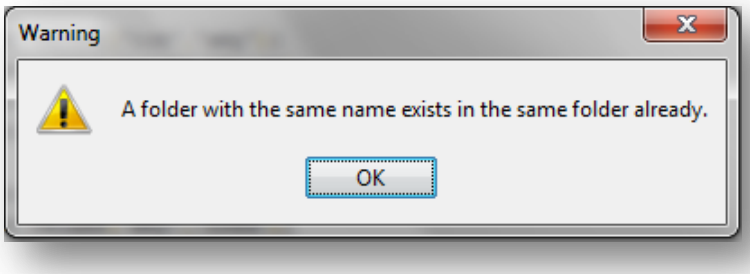


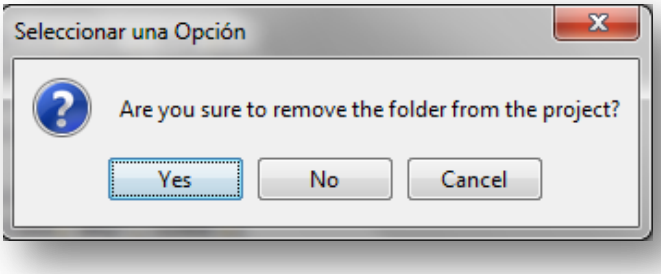

**Figure 9: Add folder**

If already exists another folder with the same name at the same level at the explorer tree, displays the following message and does not add it:




### 3.3.13. REMOVE FOLDER

Removes the folder from the project configuration previous user's confirmation:




### 3.3.14. COMPILE PROJECT

The project is compiled with the selected parameters in the compiler configuration window that will be further detailed in the following chapters of the present document. Next, we ilustrate its usage with two examples.

#### 3.3.14.1. COMPILATION BASED ON "EXTENSION"

The process has the following steps:

- First, in the compiler configuration window the user selects the extension of the files that he wants to compile:

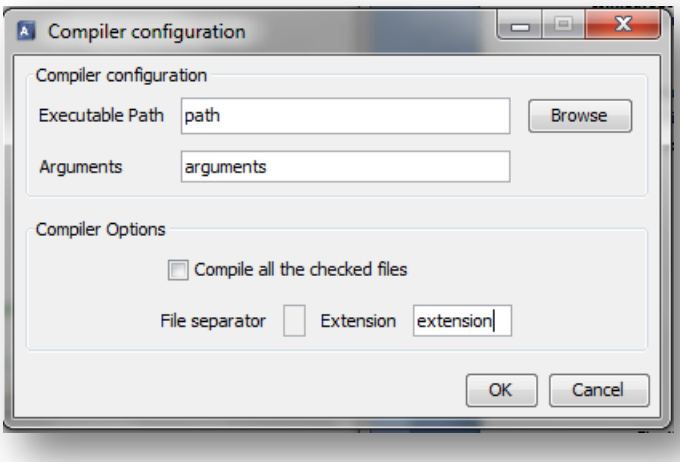



**Figure 10: Compilation by extension**

Finally, the project is compiled using the *Menu /Project/Compile* menu item option.

### 3.3.14.2. COMPILATION BASED ON "MARKED FILES FOR COMPILATION"

The process has the following steps:

- First, the user marks all the files that he wants to compile in the file editor or in the explorer tree using the option for this purpose:

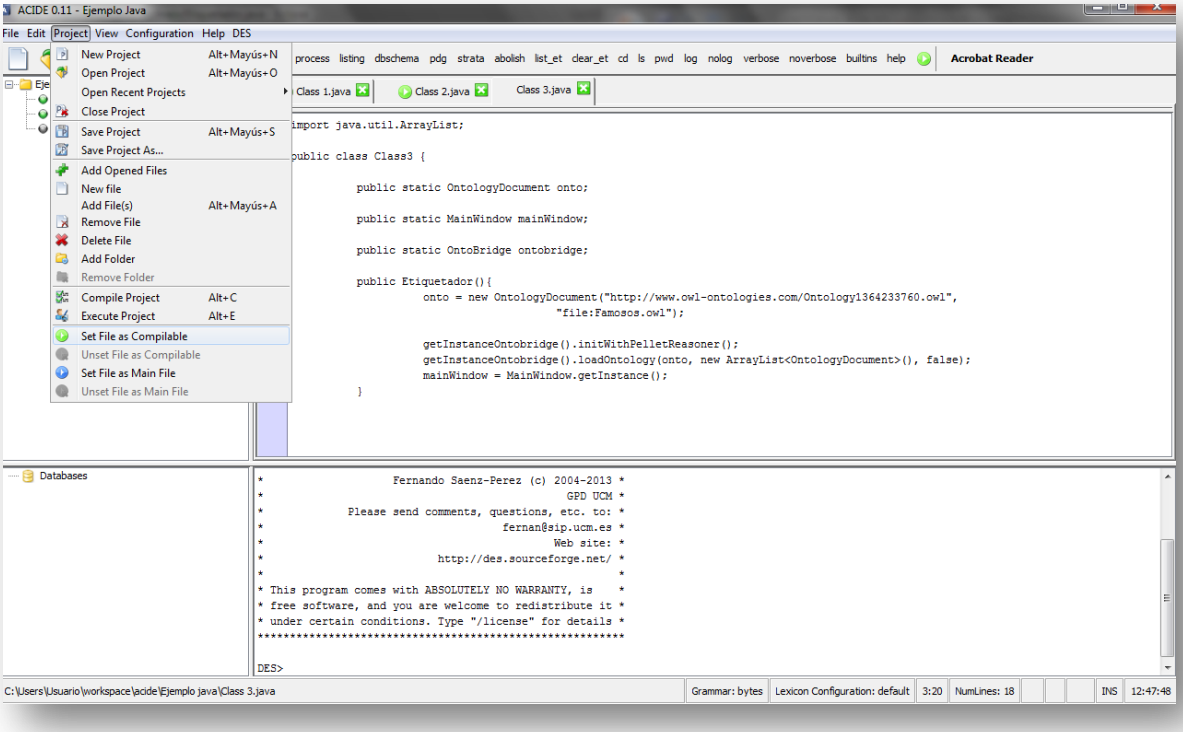



**Figure 11: Marking files**

- Next, the user configures the compiler options in the compiler configuration as follows:

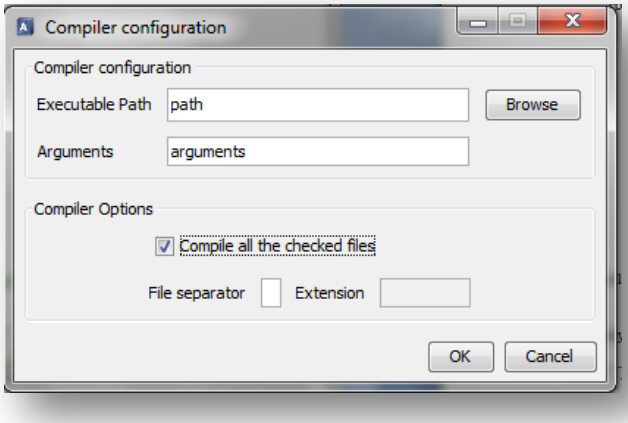



**Figure 12: Compilation by marked files**

Finally, the user selects the *Menu/Project/Compile* menu item option.

### 3.3.15. EXECUTE PROJECT

It displays the following configuration window:

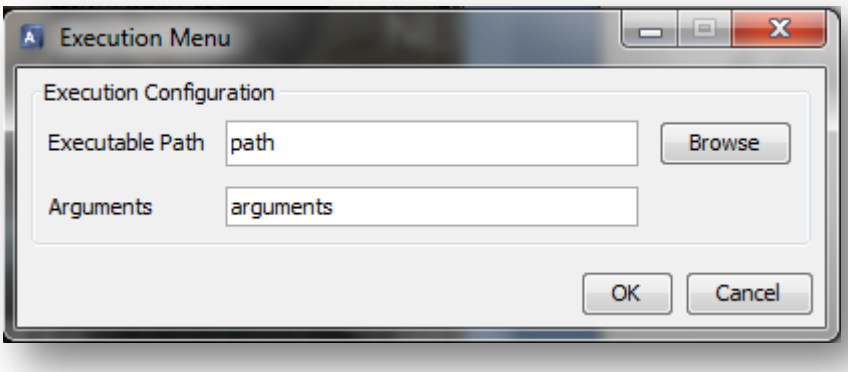



Figure 13: Execution menu

Next, we further detail all the window components:

- **Executable path:** path of the selected executable.
- **Executable arguments:** arguments for the selected executable.

The result of the execution is displayed in the following progress window:

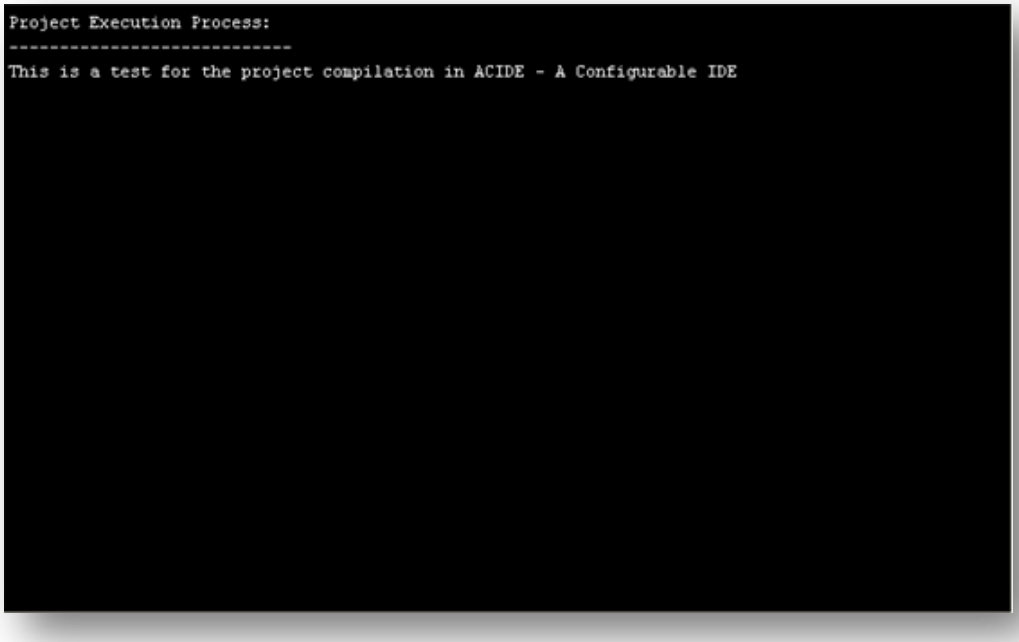



Figure 14: Execution process

### 3.3.16. SET COMPILABLE FILE

Set the active file in the file editor as compilable.

### 3.3.17. UNSET COMPILABLE FILE

Unset the active file in the file editor as compilable.

### 3.3.18. SET MAIN FILE

Set the active file in the file editor as main.

### 3.3.19. UNSET MAIN FILE

Unset the active file in the file editor as main.

## 3.4. VIEW MENU

It contains the menu items for the displaying management of the visible parts of the application and the log visualization:

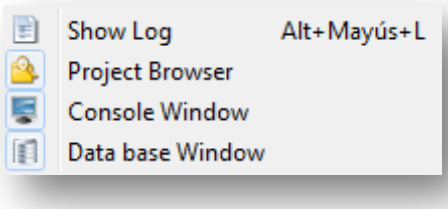



**Figure 15: View menu**

Next, all the previous menu items will be further explained:

### 3.4.1. SHOW LOG

Shows the application log in the file editor.

### 3.4.2. PROJECT BROWSER

Hides or shows the explorer panel at the left side of the main window of the application.

### 3.4.3. CONSOLE WINDOW

Hides or shows the console panel at the bottom side of the main window of the application.

### 3.4.4. DATA BASE WINDOW

Hides or shows the data base panel at the lower left corner of the application.

## 3.5. CONFIGURATION MENU

It contains all the menu item options for the configuration management of all the modules of the application:

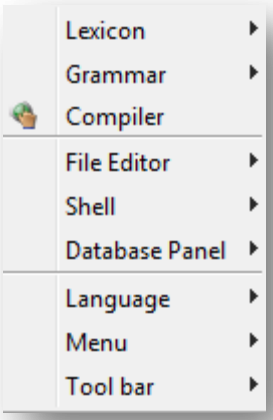



**Figure 16: Configuration menu**

### 3.5.1. LEXICON CONFIGURATION

It contains all the menu item options for the lexicon configuration management of the application:

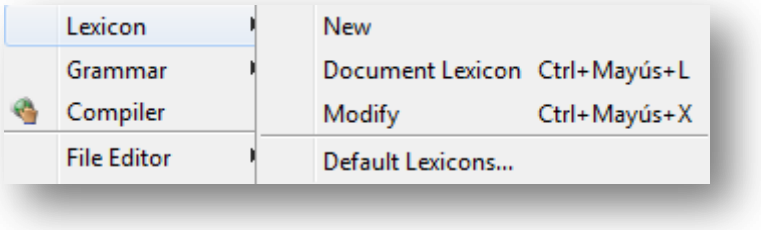



**Figure 17: Lexicon menu**

We also explain how to configure lexicons externally with *XML* files in *Chapter 13.5.* Next, all the previously mentioned options are further explained:

#### 3.5.1.1. NEW LEXICON

Creates a new lexicon configuration with the name that the user types down in the following window applying it to the active file in the file editor:

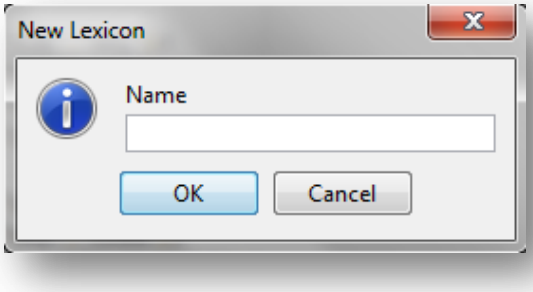



**Figure 18: New lexicon**

#### 3.5.1.2. DOCUMENT LEXICON

Loads the lexicon configuration file with **XML** extension in the active file in the file editor.

#### 3.5.1.3. MODIFY LEXICON

Open the lexicon configuration window that contains the following tabs:

#### 3.5.1.3.1. RESERVED WORDS CONFIGURATION

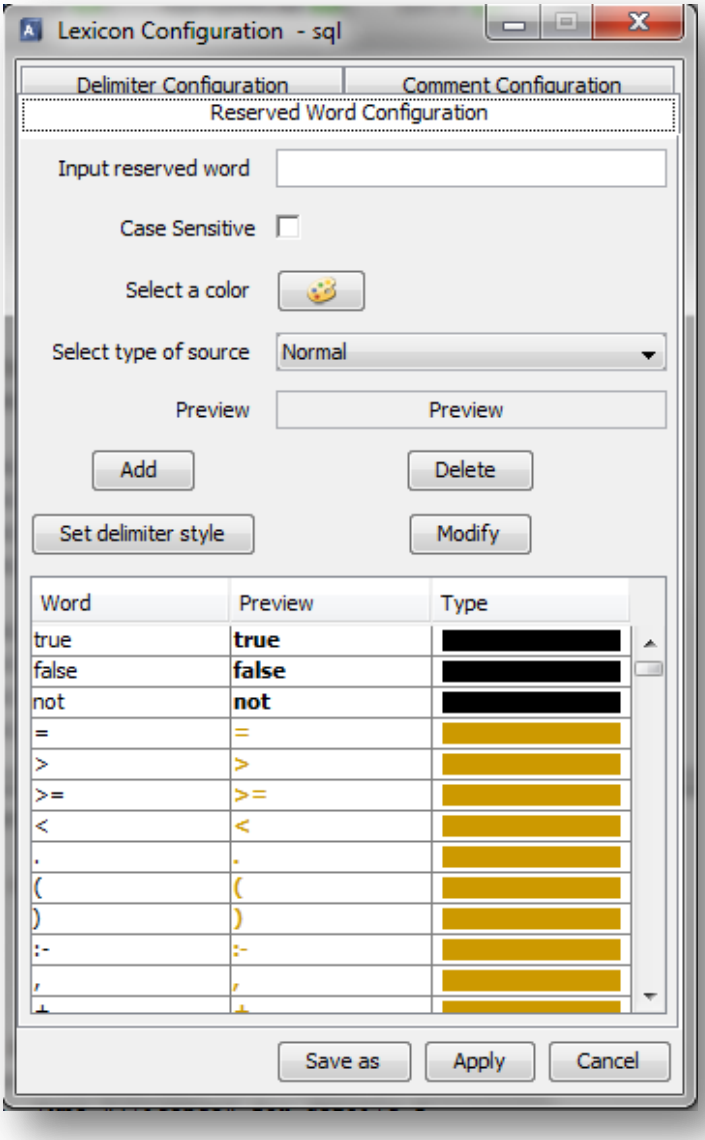



**Figure 19: Reserved words**

Next, we further describe each one of its components as follows:

- **Add:** adds a new table reserved word entry.
- **Delete:** removes a table reserved word entry.
- **Modify:** modifies a table reserved word entry.
- **Set delimiter style:** the delimiter list now is also taken as reserved words.
- **Table:** contains the list with the reserved words groups by types and colors.

  *Note:* it is not allowed to modify the table entries directly on the table itself

and the changes will not be applied until the **modify button** is pressed down.

#### 3.5.1.3.2. DELIMITERS CONFIGURATION

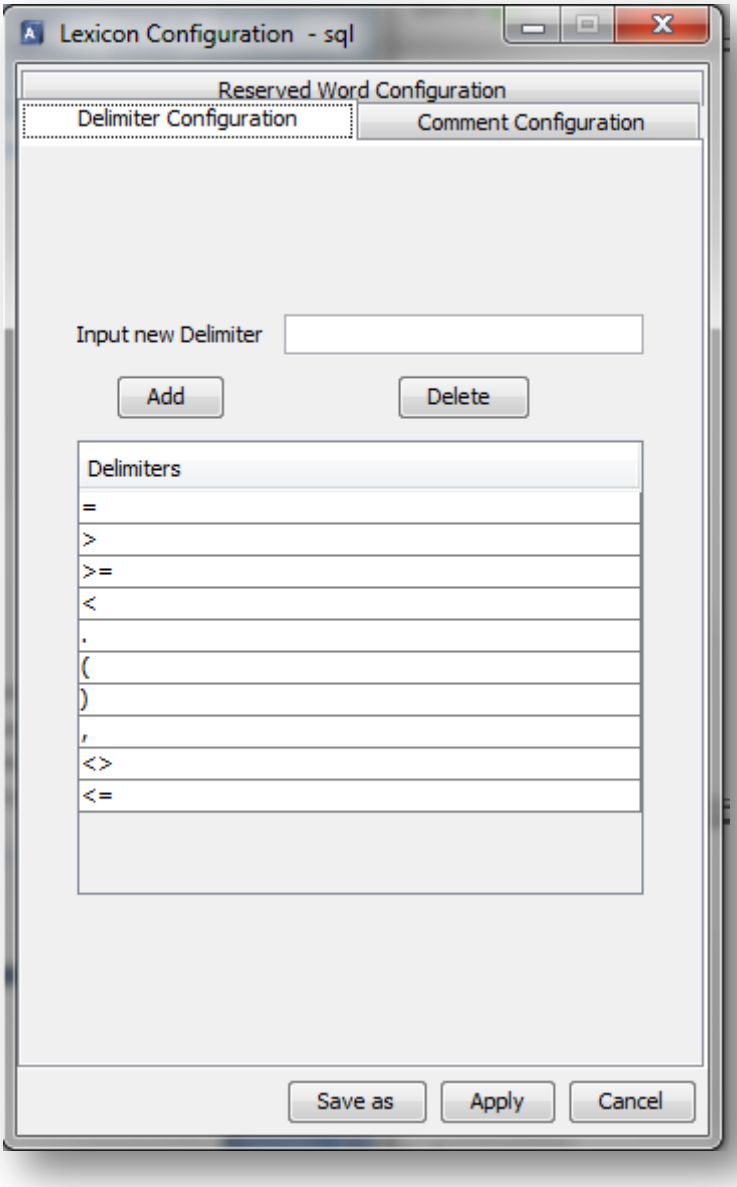



**Figure 20: Delimiters configuration**

Next, all its components are further detailed:

- **Input new delimiter text field:** the user inputs the name of the new delimiter.
- **Add button:** adds the input delimiter in the text field to the table.
- **Delete button:** removes selected delimiter from the table.

- **Table:** contains the delimiter list, and it is possible to modify it directly on it.

#### 3.5.1.3.3. REMARKS CONFIGURATION

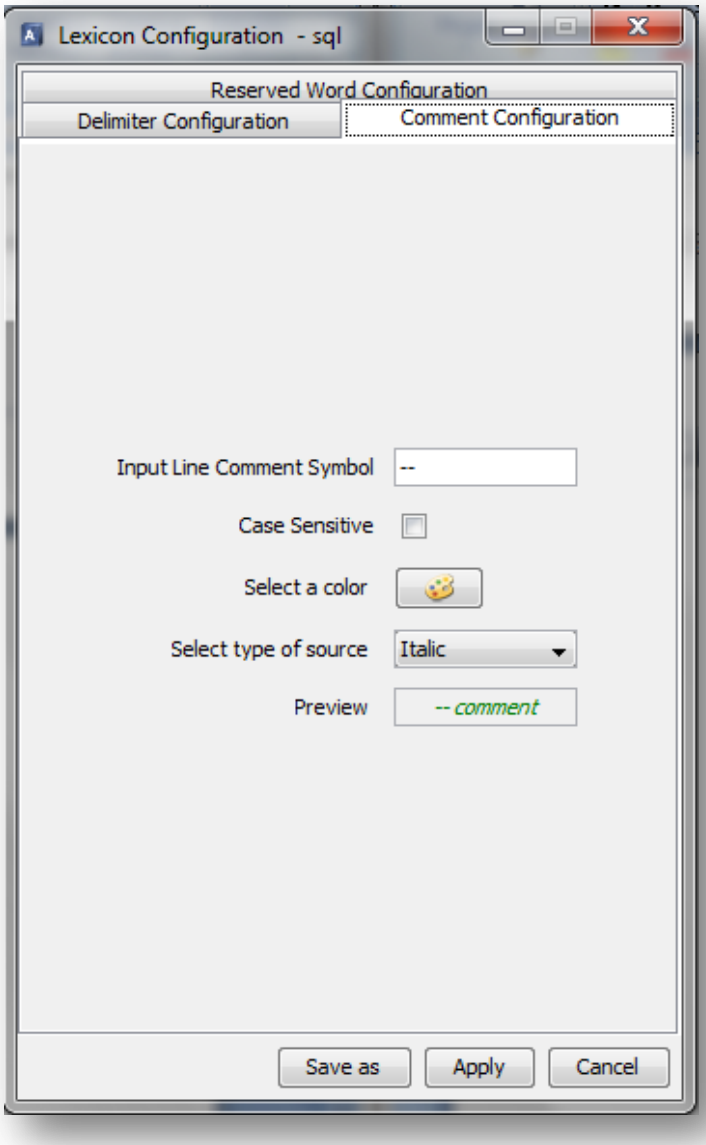



Figure 21: Remarks configuration

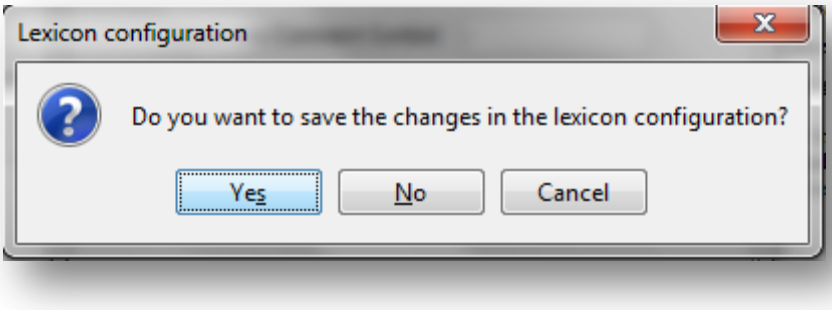
Next, we further detail all its components:

- **Comment symbol text field:** for input the remark symbol.
- **Case sensitive check box:** for specify if the remark is case sensitive or not.
- **Color selection button:** for the color selection of the remarks.
- **Font style combo box:** for the font style selection.
- **Preview text field:** shows a preview of the remarks.

The lexicon configuration window has in the bottom side the following buttons:

- **Save as:** saves the current lexicon configuration in other path with **XML** extension.
- **Apply:** applies the changes to all the opened files with the current lexicon configuration in the file editor and saves the changes in the configuration file with **XML** extension.
- **Cancel:** closes the lexicon configuration window without applying the changes.

Finally, if there are any changes in the current configuration in the previously described panels and the user closes the window with the close button or the *ESC* key, the following dialog will be displayed:




#### 3.5.1.4. DEFAULT LEXICONS

Shows the default lexicons configuration window:

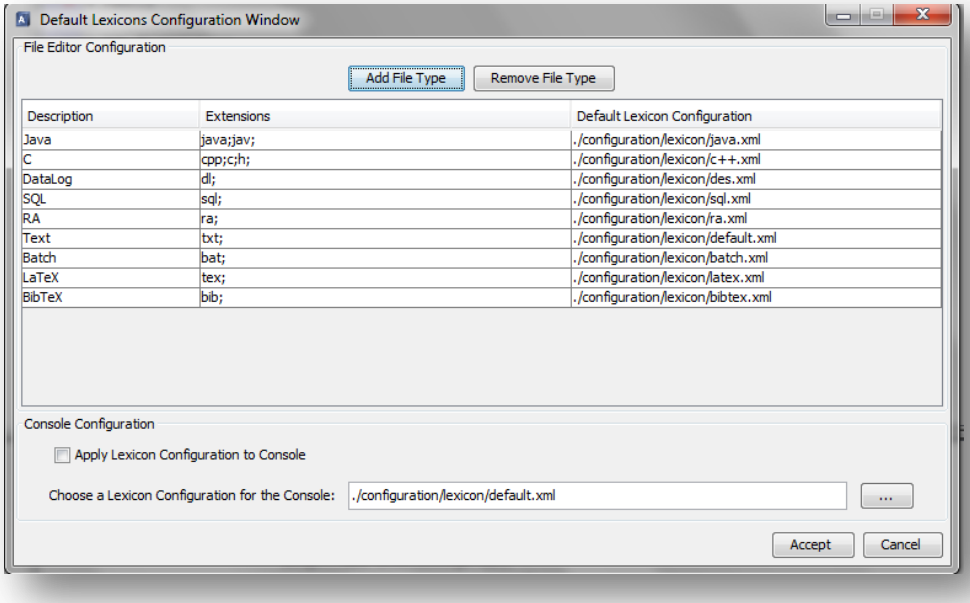



**Figure 22: Default lexicons**

Next, we explain each one of its components:

- **File editor configuration:** contains the elements for the default lexicon configurations management in the file editor:
  - **Add file type:** adds a new default lexicon configuration to the table.
  - **Remove file type:** removes a default lexicon configuration from the table.
  - **Table:** contains the following columns:
    - **Description.**
    - **Extensions:** extensions list separed by ";". *Note*: the format ".txt" is not a valid extension.
    - **Default lexicon configuration.**
- **Console configuration:** contains the elements for the default lexicon configurations management in the console panel:
  - **Apply lexicon configuration to the console:** indicates if the default lexicon configuration has to be applied or not to the console panel.
  - **Console lexicon configuration.**

### 3.5.2. GRAMMAR CONFIGURATION

It contains the menu item options for the grammar configuration management:

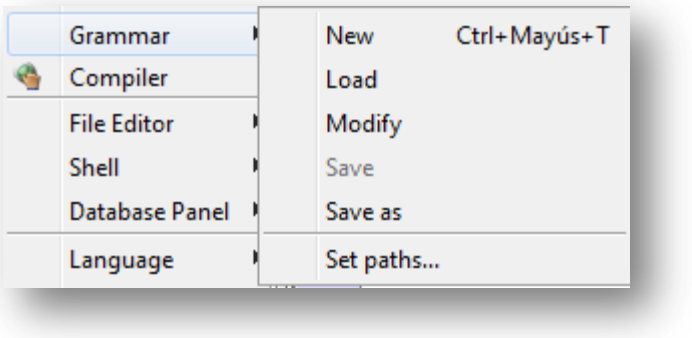



**Figure 23: Grammar menu**

Next, we further explain each one of the previous menu item options:

#### 3.5.2.1. NEW GRAMMAR

Creates the new grammar configurations from lexicon categories and grammar rules with *EBNF* format in the following configuration window:

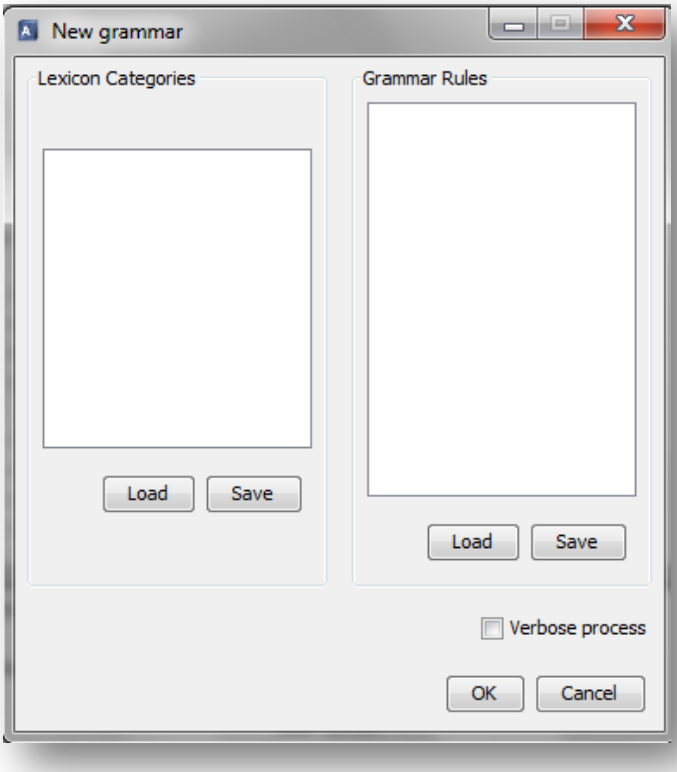



**Figure 24: New grammar**

The window has the following components:

- **Lexicon categories panel:**
    - **Lexicon categories text area:** shows the content of the lexicon categories plain text file with **TXT** extension.
    - **Load button:** loads the content of the lexicon categories plain text file with **TXT** extension into the lexicon categories text area.
    - **Save button:** saves the content of the lexicon categories text area into a plain text file with **TXT** extension.
- **Grammar rules panel:**
    - **Text box of grammar rules:** shows the content of the grammar rules plain text file with **TXT** extension.
    - **Load button:** loads the content of the grammar rules plain text file with **TXT** extension into the grammar rules text area.
    - **Save button:** saves the content of the grammar rules text area into a plain text file with **TXT** extension.
- **Accept button:** initializes the grammar creation process.
- **Cancel button:** closes the window without applying the changes.

In the moment that the new grammar is created, it is not saved until the user selects the save menu option. In the case that the user closes the application without saving it, the last grammar configuration will be loaded.

If the user selects to verbose the grammar creation process, the following window will be displayed:

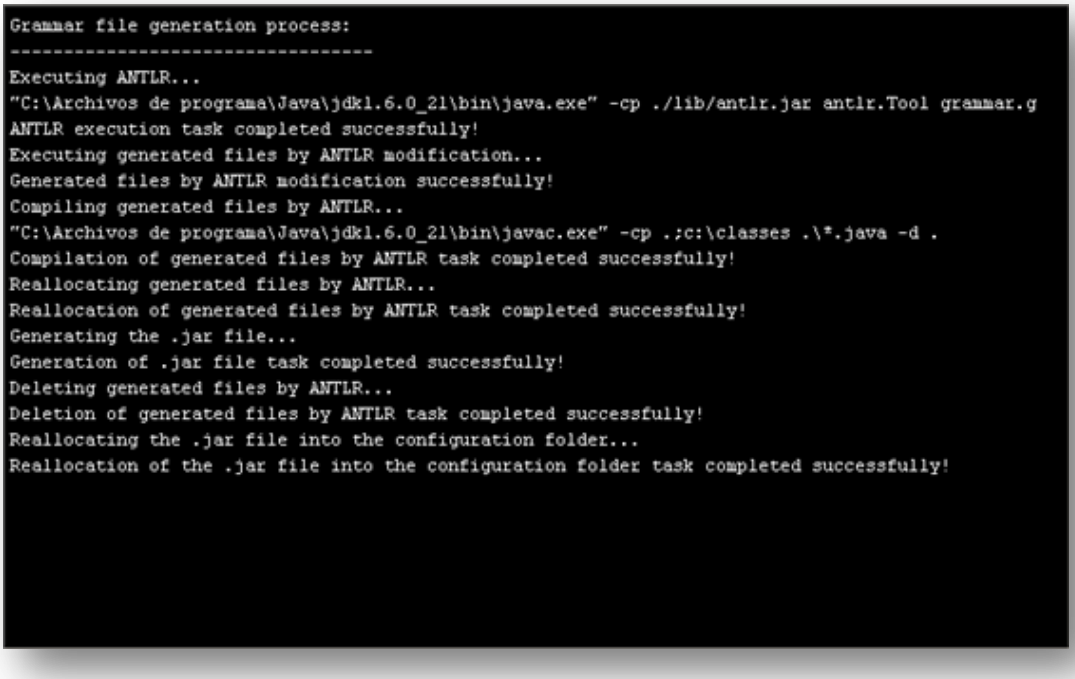



**Figure 25: Grammar generation process**

#### 3.5.2.2. LOAD GRAMMAR

Loads a grammar configuration with **JAR** extension.

#### 3.5.2.3. MODIFY GRAMMAR

Displays the same grammar configuration window than the **New Grammar** menu item option but it contains the lexicon categories and grammar rules text areas filled with their file contents:

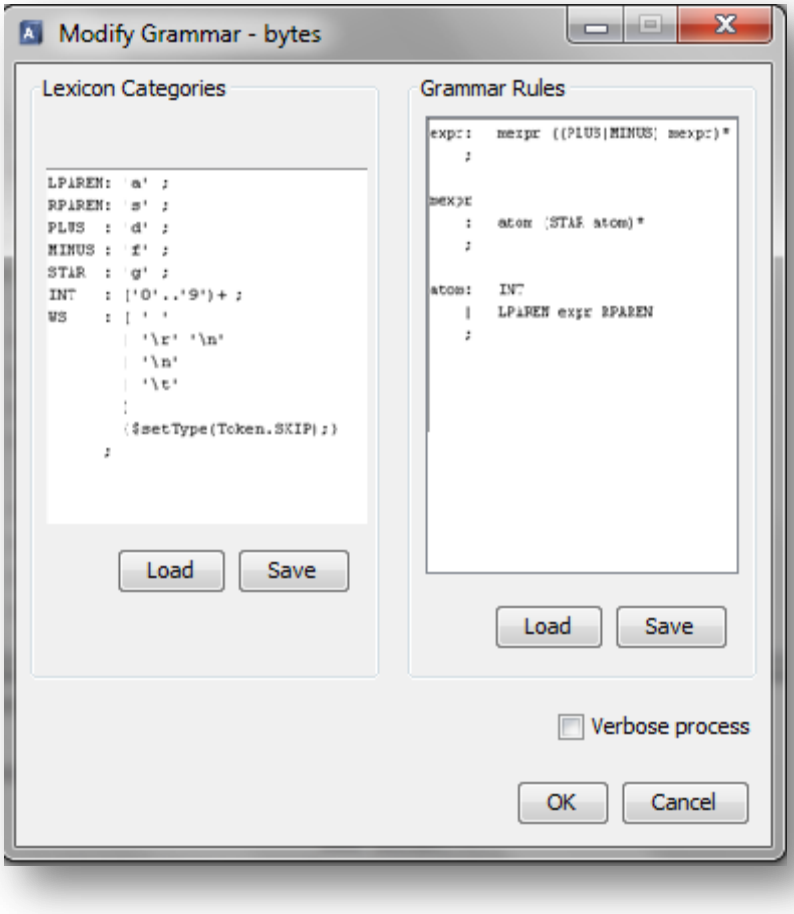



**Figure 26: Modify grammar**

#### 3.5.2.4. SAVE GRAMMAR

Saves the current grammar configuration into a file with **JAR** extension.

#### 3.5.2.5. SAVE GRAMMAR AS

Saves the current grammar configuration into a file with **JAR** extension in a different path.

#### 3.5.2.6. CONFIGURE PATHS

For the creation, modification and grammar configurations to hand it is mandatory to define the required tools paths as it was mentioned in the first chapter of the present document.

It displays the following window:

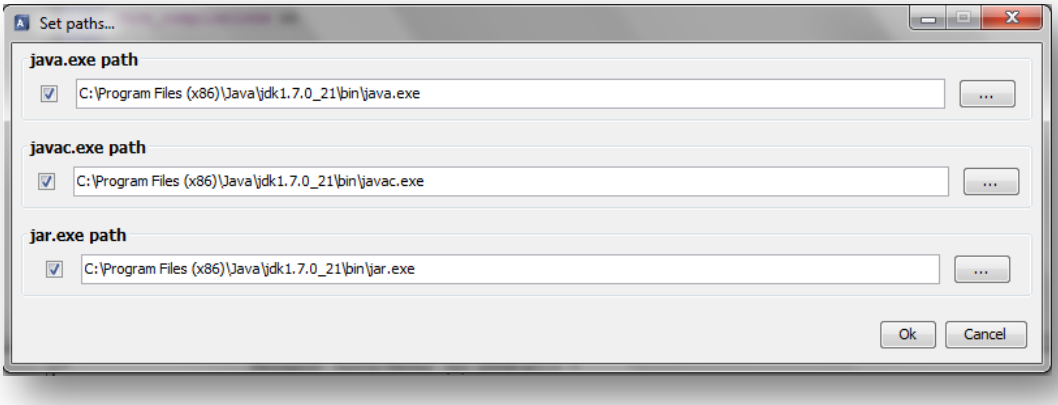



**Figure 27: Set paths**

In each one of the text fields the user will select the path to each one of the required tools. The window also contains the following components:

- **Check box:** if it is selected the application will use the path selected in the text field that corresponds; if it is disabled the application will use its Operative System CLASSPATH.
- **Explorer buttons:** open a dialog window for the files selection.

### 3.5.3. COMPILER

The following window will be displayed:

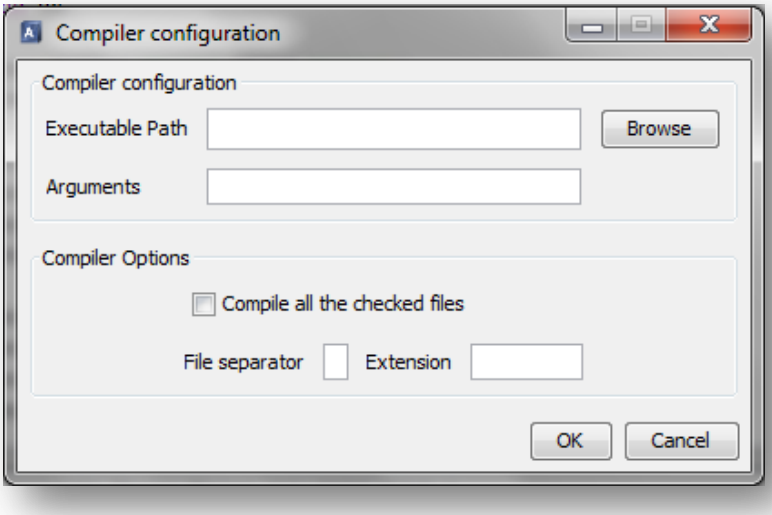



**Figure 28: Compiler configuration**

The window has the following components:

- **Compiler configuration panel:**
  - **Executable path:** path that contains the compiler executable file.
  - **Compiler arguments:** arguments for the compiler.
- **Compiler options panel:**
  - **Compile all the checked files:** indicates if all the compilable files have to be compiled or not.
  - **File separator:** file separator to separate each one of the files to compile.
  - **Extension:** file extension of the files to compile.
- **Accept button:** apply the changes.
- **Cancel button:** close the window and do not apply the changes.

### 3.5.4. FILE EDITOR CONFIGURATION

It contains the menu item options for the file editor configuration management:

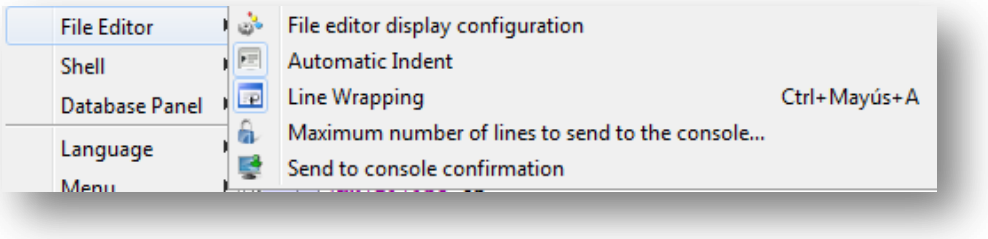



**Figure 29: File editor configuration**

We also explain how to configure the file editor externally with XML files in *Chapter 13.3.3.* Next, we further detail each one of the previous menu item options:

#### 3.5.4.1. FILE EDITOR DISPLAY OPTIONS CONFIGURATION

Displays the following configuration window:

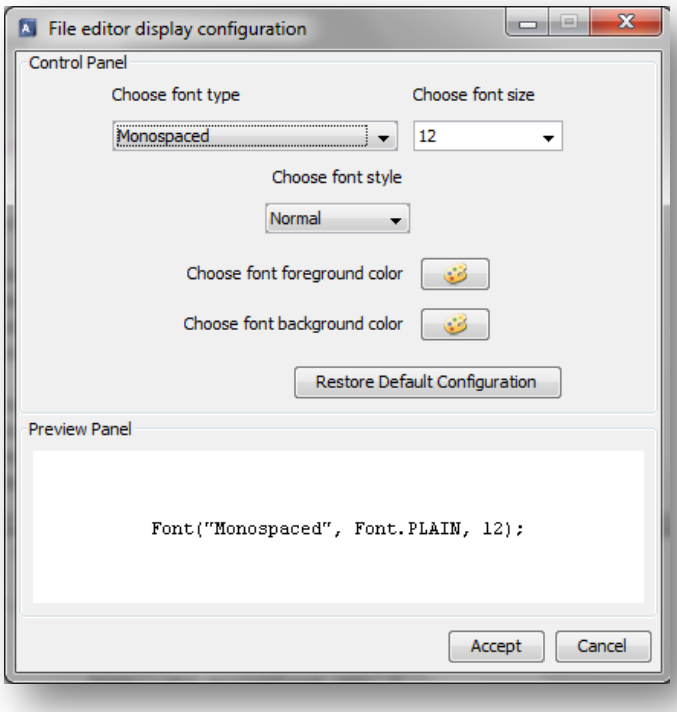



**Figure 30: File editor display options**

In the configuration window, the user can configure the following parameters:

- **Font type.**
- **Font size.**
- **Font style.**
- **Foreground color.**
- **Background color.**
- **Restore default values:** applies the default configuration:

*"Monospaced" font, plain, size of 12, black with white blackground.*

#### 3.5.4.2. AUTOMATIC INDENT

Enables or disables the automatic indent in the file editor.

#### 3.5.4.3. LINE WRAPPING

Enables or disables the line wrapping in the file editor.

#### 3.5.4.4. MAXIMUM LINE NUMBER TO SEND TO CONSOLE

Asks to the user for the maximum number of lines to send to the console panel from the active file in the file editor:

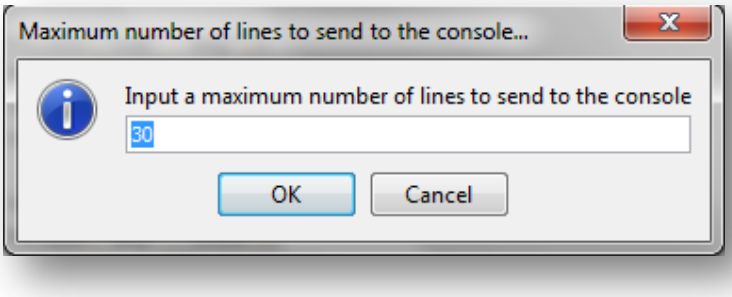
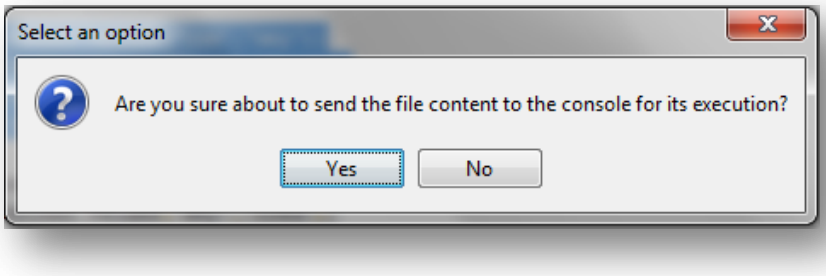



**Figure 31: Maximum line number**

#### 3.5.4.5. SEND TO CONSOLE CONFIRMATION

If this option is selected, when the user sends contents of the active file in the file editor the application will display the following confirmation message:




If this option is not selected, when the user sends contents of the active file in the file editor the application simply sends the contents to the console panel adding each sent line as a separate command in the console panel command record.

### 3.5.5. CONSOLE CONFIGURATION

It contains the menu item options for the console panel configuration management:

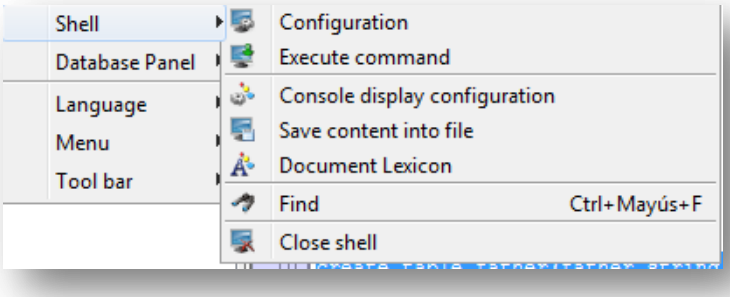



**Figure 32: Console menu**

We also explain how to configure console panel externally with *XML* files in *Chapter 13.3.4.*

#### 3.5.5.1. CONFIGURE

Configures the shell configurations that are loaded in the console panel:

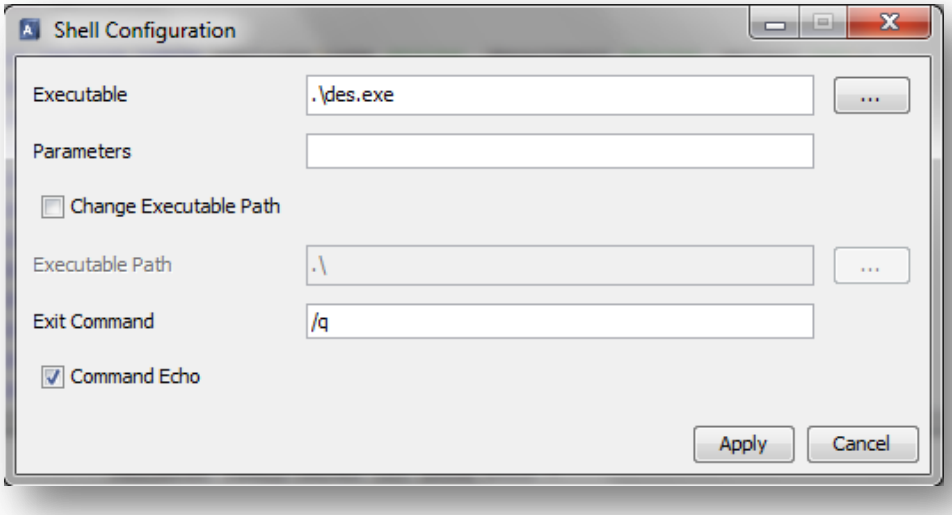



**Figure 33: Shell configuration**

It contains the following components:

- **Executable:** executable file path.
- **Parameters:** shell is configured with these parameters.

- **Change executable path:** it is used for especifying a different folder where the executable file is placed.
- **Executable path:** executable file folder.
- **Exit command:** exit command for closing the data stream.
- **Command echo:** indicates if the commands typed in the console panel have to be displayed or not.

#### 3.5.5.2. EXECUTE EXTERNAL COMMAND

Executes a command into a shell and displays the result in a separate window that looks like:

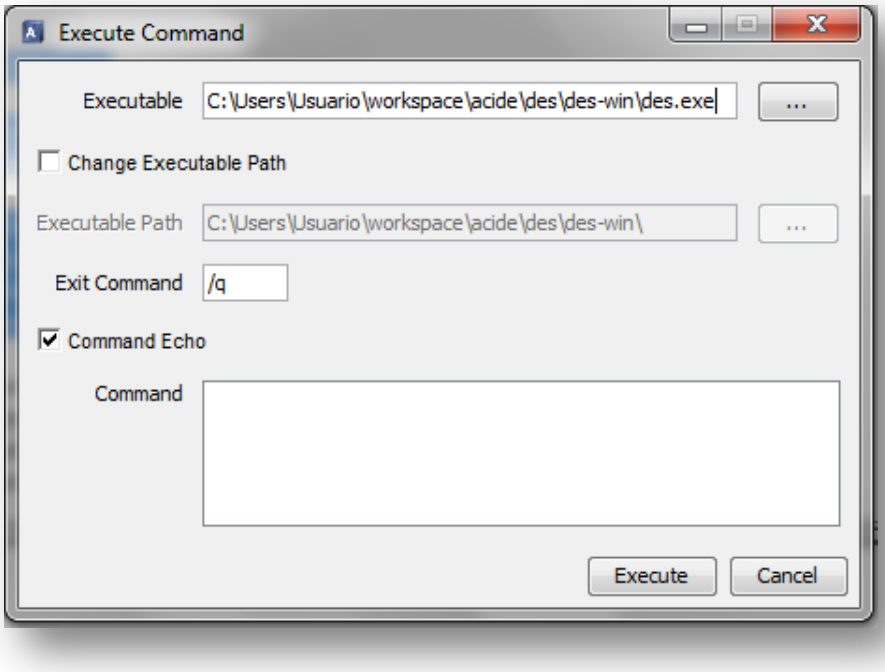



**Figure 34: Execute external command**

#### 3.5.5.3. CONSOLE DISPLAY CONFIGURATION

Displays the following configuration window:

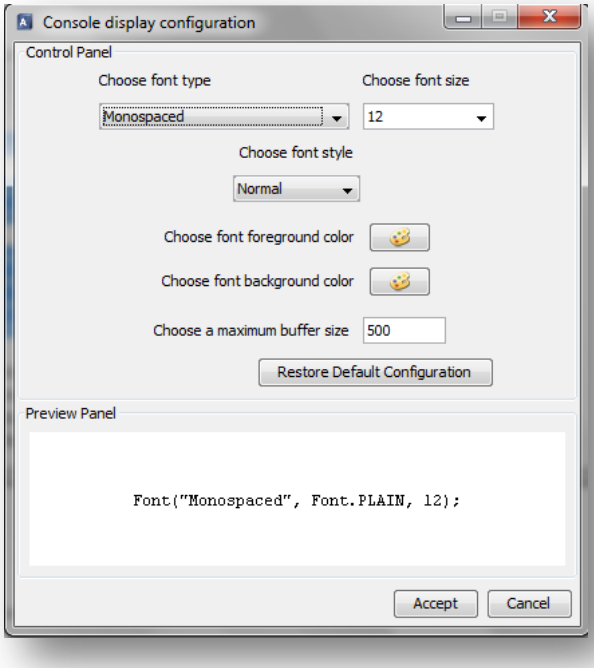



**Figure 35: Console display configuration**

The user can selects:

- **Font type.**
- **Font size.**
- **Font color.**
- **Background color.**
- **Maximum buffer size:** specifies the maximum number of lines that are displayed in the console panel.
- **Restore default configuration:** applies the default configuration for the console panel:

*"Monospaced" font, plain, size of 12, black with white background*

#### 3.5.5.4. SAVE CONTENT INTO FILE

Saves the console content into a file.

#### 3.5.5.5. DOCUMENT LEXICON

Loads a lexicon configuration with **XML** extension into the console panel.

### 3.5.5.6. FIND

Displays the search text window for the console panel:

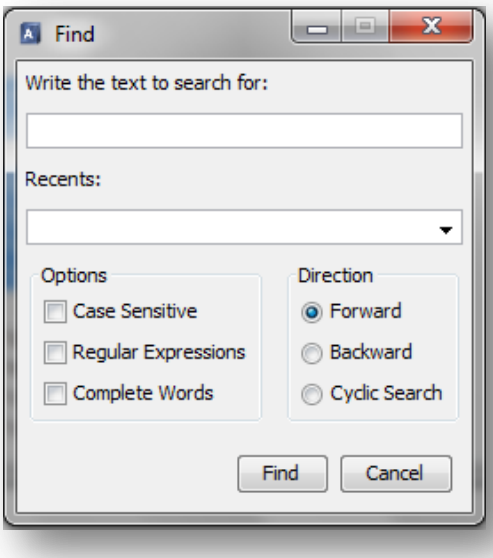



**Figure 36: Console search window**

Then we proceed to describe each component of the window:

- **Text box:** Here is where user enters the search text.
- **Recents:** This combo menu displays a list which contains all the recent searches that have been executed before. When user selects one, this appears in the Text box.
- **Options:**
  - **Case sensitive:** this option is used to search for strings without having or taking into account the Upper / Lowercase.
  - **Regular expressions:** regular expressions search associated with a search pattern. More information about Regular Expressions on *Chapter 14*.
  - **Whole words:** find whole words only.
- **Direction:**
  - **Forward:** searches from the current caret position to the end of the file in the source file editor.
  - **Backward:** searches from the current caret position to the beginning of the file in the source file editor.

- **Cyclic:** searches from the current caret position to the end of the file in the source file editor, and star from the beginning until the starting position.

#### 3.5.5.7. CLOSE CONSOLE

Closes the active shell in the console panel.

#### 3.5.5.8. RESET CONSOLE

Only available in the *popup menu* of the console panel. Resets the active shell in the console panel.

#### 3.5.5.9. CLEAR CONSOLE BUFFER

Only available in the *popup menu* of the console panel. Clears the console panel content and leaves only the last line of the previous buffer content.

### 3.5.6. DATABASE PANEL CONFIGURATION

It contains the menu item options for the database panel configuration management:


**Figure 37: Database panel menu**

Then we proceed to describe each component of the menu:

#### 3.5.6.1. DES PANEL

When this item is selected, the database panel in the left lower corner is connected with *DES*.

#### 3.5.6.2. ODBC PANEL

When this item is selected, the database panel in the left lower corner is connected with *ODBC*.

### 3.5.7. LANGUAGE CONFIGURATION

Shows the available language list of the application:

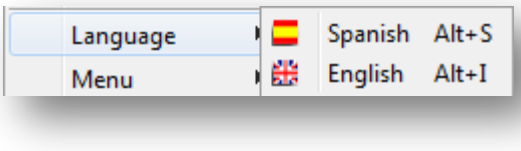



**Figure 38: Language configuration menu**

In this case, the user can choose only between *English* or *Spanish.*

### 3.5.8. MENU CONFIGURATION

It contains the menu item options for the menu configuration management:

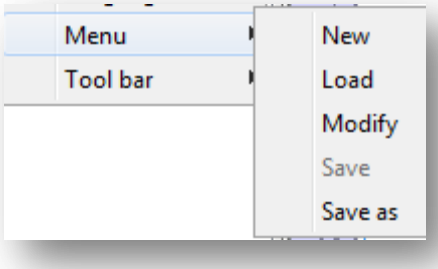



**Figure 39: Menu configuration menu**

We also explain how to configure menu externally with *XML* files in *Chapter 13.3.1.* Next, we further describe each one of the previous menu item options:

#### 3.5.8.1. NEW

Displays the following configuration window:

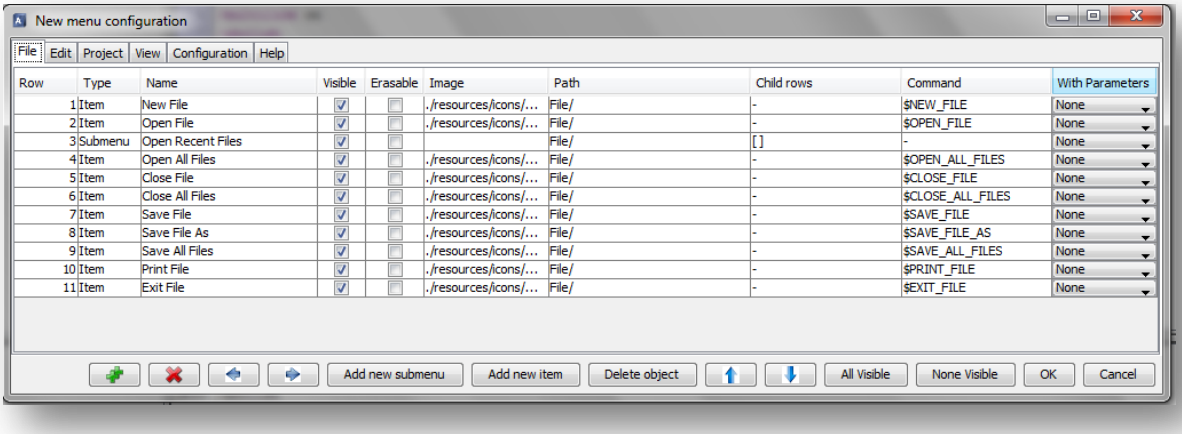



**Figure 40: New menu**

It displays a list of tabs with the names of the menus in the *Menu bar*. For each tab there is a grid with attributes of its menu objects.

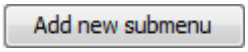
The user can edit directly in the grid the attributes he wants to change, except some that are not editable. The value it is not assigned until user hits *ENTER* or changes to other attribute or object. Next, we further describe each one of the menu objects options:

- **Row:** the number of the row. It is not editable.
- **Type:** the type of the menu object in this row. It can ben *Item* or *Submenu*. It is not editable, this value is assigned when the object is created.
- **Name:** the name of the menu object. It is editable.
- **Visible:** this value sets if the menu object is visible in the *Menu bar* or not.
- **Erasable:** this value indicates if this menu object is a default menu object or not. It is not editable. The menu objects with erasable value to false are default menu objects. These objects have to be always in the *Menu bar* configuration, altough they can be not visible. When the application builds the menu, it checks if all the default menu objects exit in the configuration. If any menu object does not exit, the application creates it at the end of its submenu. It can exits only one of each default menu object. The application will delete the rest.
- **Image:** the path of the image icon of the menu item. The image icons belong only to menu items.

- **Path:** indicates the location of the menu object inside the menu which contains it.
- **Child rows:** it is only for menu submenus. It indicates the number of row of their childern.
- **Command:** it is only for menu items. It sets the command that the menu item will run. The commands that start with a *"$"* sign are intern commands for *ACIDE – A Configurable IDE* and they are explained on *Chapter 12*. Commands that not start with *"$"* are sent to console.
- **With parameters:** it is only for menu items. Indicates the type of parameter which the command needs to run.

#### 3.5.8.1.1. BUTTONS PANEL

Next, we further describe each of the buttons of the configuration window:

-  **Add new menu** : It will display a window where user can type down the name of the new menu he wants to insert. It will be inserted at the end of the menus list.
-  **Delete menu:** It will delete the present menu before a confirmation message. The default menu can be deleted.
-  **Move menu to the left:** moves the present menu to the left in the menus list.
-  **Move menu to the right:** moves the present menu to the left in the menus list.
- 

: adds a new submenu to the menu selected. If there is a menu submenu selected, the new submenu will be inserted inside it. If there is a menu item selected, the new submenu will be inserted after it. In other case, the new submenu will be inserted at the end of the list of the root menu.
- 

: adds a new menu item to the menu selected. If there is a menu submenu selected, the new item will be inserted inside it. If there is a

menu item selected, the new item will be inserted after it. In other case, the new item will be inserted at the end of the list of the root menu.

- Delete object : deletes the selected object after a confirmation message. The objects that are not erasable can not be deleted.
- **Move object to up**:moves to up the selected menu object.
- **Move object to down**: moves down the selected menu object.
- All Visible: sets as visible all the menu objects of the *Menu Bar*.
- None Visible: sets as no visible all the menu objects of the *Menu Bar*. The menu objects related to menu configuration always have to be visible.
- OK : Applies the changes and closes the window.
- Cancel : Closes the window without applying the changes.

#### 3.5.8.1.2. POPUP MENU

The *popup menu* of menu object is as follows:

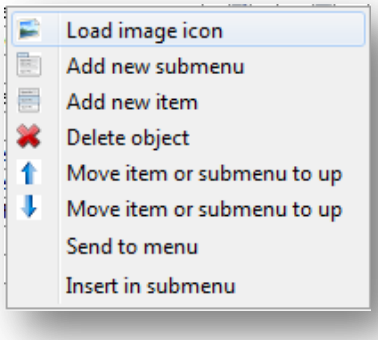



**Figure 41: Object menu popup menu**

Next, we further describe each of the options:

- **Load image icon**: it will display a load window where user can select the image he wants to set as icon of the menu object.
- **Add new submenu**: adds a new submenu to the menu selected. If there is a menu submenu selected, the new submenu will be inserted inside it. If there is a menu item selected, the new submenu will be inserted after it. In

other case, the new submenu will be inserted at the end of the list of the root menu.

- **Add new item**: adds a new menu item to the menu selected. If there is a menu submenu selected, the new item will be inserted inside it. If there is a menu item selected, the new item will be inserted after it. In other case, the new item will be inserted at the end of the list of the root menu.
- **Delete object**: deletes the selected object after a confirmation message. The objects that are not erasable can not be deleted.
- **Move item or submenu to up**: moves to up the selected menu object.
- **Move item or submenu to down**: moves down the selected menu object.
- **Send to menu**: it displays a window with a list of menus where user can send the selected menu object.
- **Insert in submenu**:it displays a window with a list of submenus inside the present menu where user can insert the selected menu object.

#### 3.5.8.1.3. KEY NAVEGATION

- **Up arrow:** selects previous object.
- **Down arrow:** selects next object.
- **Ctrl + Home:** selects the first object.
- **Ctrl + End:** selects the last object.
- **Tab:** selects next attribute.
- **Tab + Shift:** selects previous attribute.
- **Esc:** deselects the selected object.

### 3.5.8.2. LOAD

Loads a menu configuration with **XML** extension.

### 3.5.8.3. MODIFY

Selecting this option displays the following configuration window, similar to creating a new configuration window, but with corresponding options of the loaded menu and with the name of the configuration in the window title:

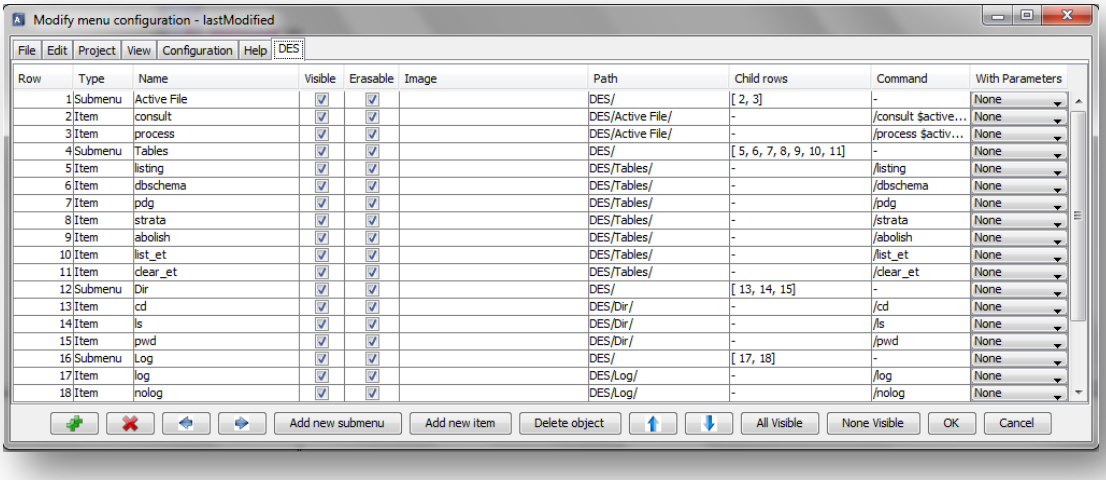



**Figure 42: Modify menu**

Its performance is equal to the new menu window explained on *Chapter 3.5.8.1*.

### 3.5.8.4. SAVE

Saves the current menu configuration into a menu configuration file with **XML** extension.

### 3.5.8.5. SAVE AS

Saves the current menu configuration into a menu configuration file with **XML** extension in a different path.

### 3.5.9. TOOLBAR CONFIGURATION

It contains the menu item options for the tool bar configuration management:

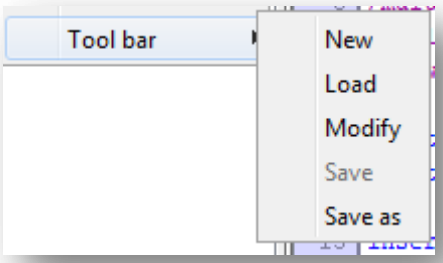



**Figure 43: Tool Bar configuration menu**

We also explain how to configure toolbar externally with *.toolbarConfig* files in *Chapter 13.3.2*

#### 3.5.9.1. NEW

It displays the following configuration window:

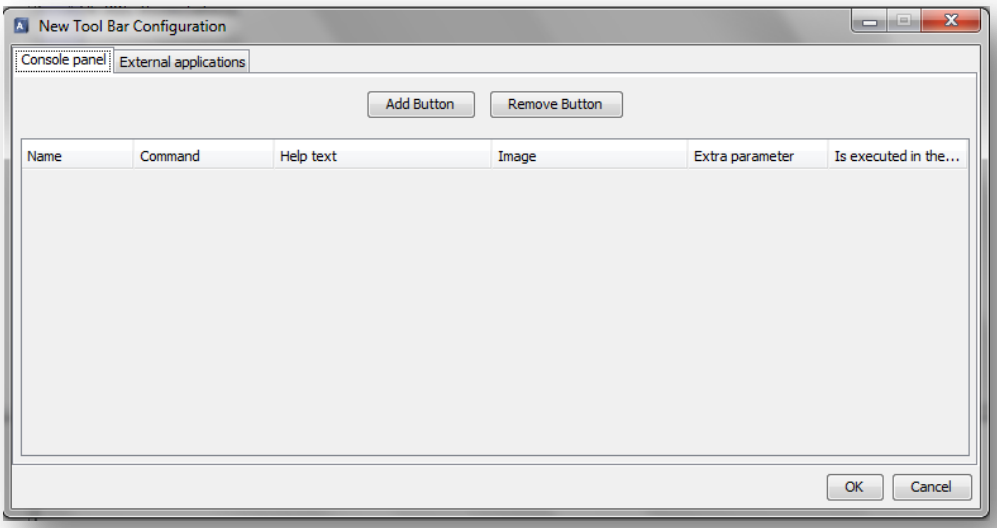



**Figure 44: New tool bar**

The window has two different panels:

- **Console panel:** defines the commands related to the console panel tool bar that are executed in the console panel.
- **External applications panel:** defines the commands related to the external applications tool bar that are executed out of the aplication.

In each one of the panels, the user can do the following operations:

- **Add button:** adds a new command to the command list in the table.
- **Remove button:** removes the selected command from the command list.
- **Direct edition on the tables:** the user can modify the commands by editing directly on the table. However, the changes will not be applied until the focus changes or the user presses down the *ENTER* key.

In the *console panel* tab the table contains the following parameters:

- **Name:** text to display in the button. If this field is empty the application will assign it a number as name by default.
- **Command:** command itself. It admits the insertion of *ACIDE – A Configurable IDE special variables* that are further detailed in the *Chapter 11* of the present document.
- **Help text:** hint text of the button.
- **Image:** image for the button which can be selected by the option available in the *popupmenu* of the column.
- **Extra parameter:** shows a combo box with the following options: *NONE, TEXT, FILE, DIRECTORY.* Each one of the previous options will ask the user for the selected type with different dialog windows.
- **Is executed in the OS shell:** indicates if the command is executed in the Operative System shell or in the loaded shell in the console panel.

In the *external applications panel* tab the table contains the following parameters:

- **Name:** text to display in the button. If this field is empty the application will assign it a number as name by default.
- **Executable path:** executable path of the command to execute. It admits the insertion of *ACIDE – A Configurable IDE special variables* that are further detailed in *Chapter 11* of the present document.
- **Help text:** hint text of the button.
- **Image:** image for the button which can be selected by the option available in the *popup menu* of the column.

The tool bar configuration files have *toolbarConfig* extension.

#### 3.5.9.2. LOAD

Loads a tool bar configuration with *toolbarConfig* extension.

#### 3.5.9.3. MODIFY

It displays the following configuration window:

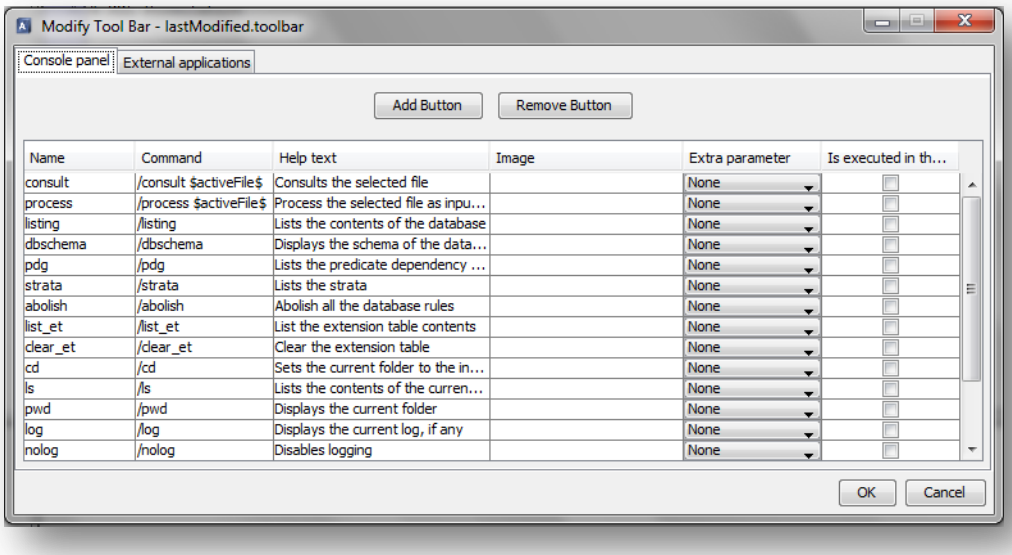



**Figure 45: Modify tool bar**

It contains the same options than the configuration window displayed by the *Configuration/Menu/New.*

In this case, the window displays the current tool bar configuration loaded in the tables and also with a different window title which contains the name of the current configuration to modify.

#### 3.5.9.4. SAVE

Saves the current tool bar configuration into a tool bar configuration file with ***toolbarConfig*** extension.

#### 3.5.9.5. SAVE AS

Saves the current tool bar configuration into a tool bar configuration file with ***toolbarConfig*** extension and with a different path.

## 3.6. HELP MENU

Contains the following menu items:

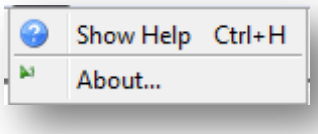



**Figure 46: Help menu**

Next, the previous menu options are further explained:

### 3.6.1. SHOW HELP

Links directly to the present document.

### 3.6.2. ABOUT US

Displays the following window with some extra information about the application:

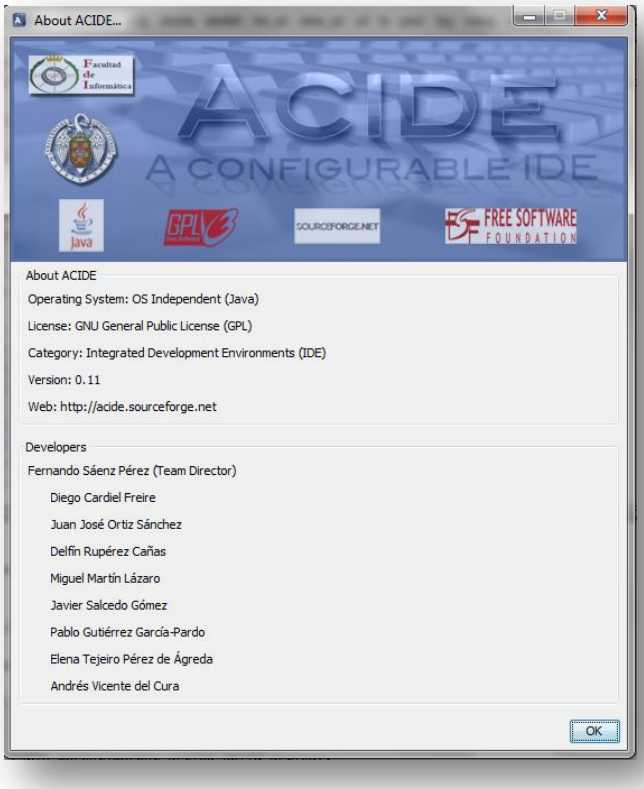



**Figure 47: About us window**

## 3.7. INSERTED SUBMENUS

As explained in *chapters 3.5.8* and *13.3.1* , user can insert new submenus in the tool bar. Then, inside these submenus new inserted submenus and menu items can be defined. For each inserted submenu the attributes are:

- **Name:** the name of the submenu.
- **Visible:** define if the submenu is visible or not in the application.
- **Erasable:** define if the submenu is a default submenu (not erasable) or not (is erasable). The value of this attribute can not be edited.
- **List:** list of all the submenus and menu items that the submenu contains.
- **Image:** for submenus the value of this attribute is empty.

An example of an inserted submenu is:

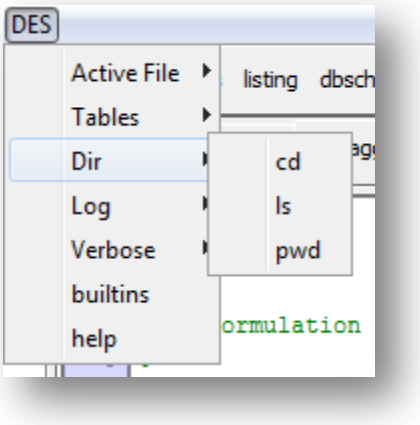



**Figure 48: Example of inserted submenu**

In this example we can see an inserted submenu called *DES* and defined in the menu bar.

## 3.8. INSERTED MENU ITEMS

As explained in *chapters 3.5.8* and *13.3.1 ,* user can insert new menu items in the tool bar. For each inserted menu item the attributes are:

- **Name:** the name of the menu item.
- **Visible:** define if the menu item is visible or not in the application.
- **Erasable:** define if the menu item is a default menu item (not erasable) or not (is erasable). The value of this attribute can not be edited.
- **Image:** defines the path of the image which is the icon of the menu item.
- **Command:** defines the command that is sent to console when this menu item is clicked.
- **Parameter:** defines the type of parameter that the command of this menu item needs: *None*, *Text*, *File* or *Directory.*

A example of inserted menu items can be seen in *Chapter 3.7* of the present document.

# 4. PROJECT BROWSER PANEL

In the project browser panel are displayed the folders and files of the active project. The *main files* appear with a blue circle beside, the *compilable files* with a green circle and the rest with a grey circle:

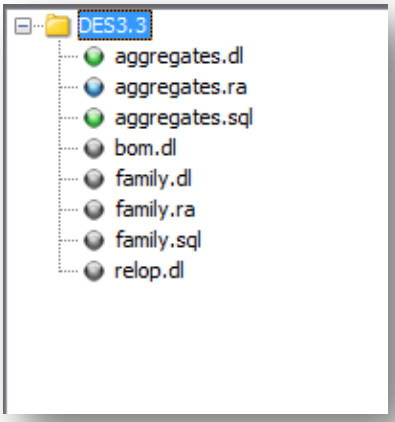



**Figure 49: Project browser panel**

The *popup menu* of each file and folder is as follow:

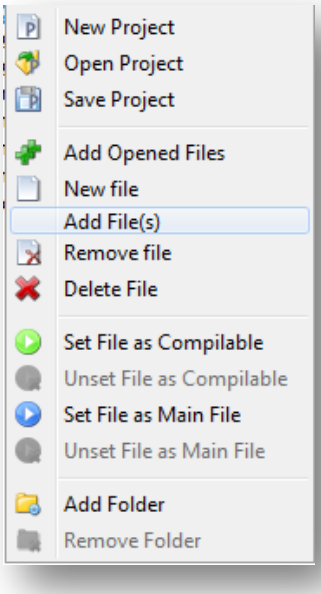



**Figure 50: Project browser popup menu**

All these options have been explained before on *Chapter 3.3.*

# 5. FILE EDITOR PANEL

In the file editor panel are displayed all the opened files by tabs. Each tab is named by the name of the file it contains:

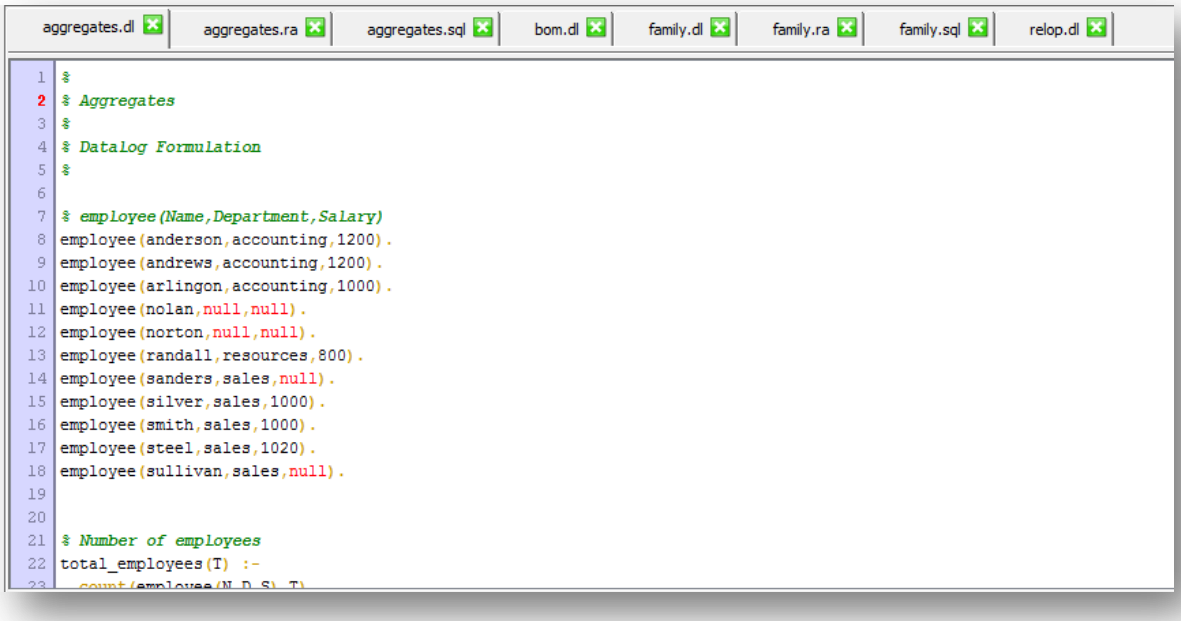



**Figure 51: File editor panel**

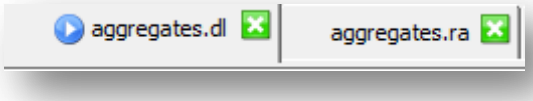
When a file is modified and it is not saved yet, its tab is as follows:


with a red cross beside the title of the tab.

When a file is set as compilable file, its tab is as follows:


with a green play sign beside the title of the tab.

And finally, when a file is set as main file, its tab is as follows:




With a blue play sign beside the title of the tab.

The *popup menu* is as follow:

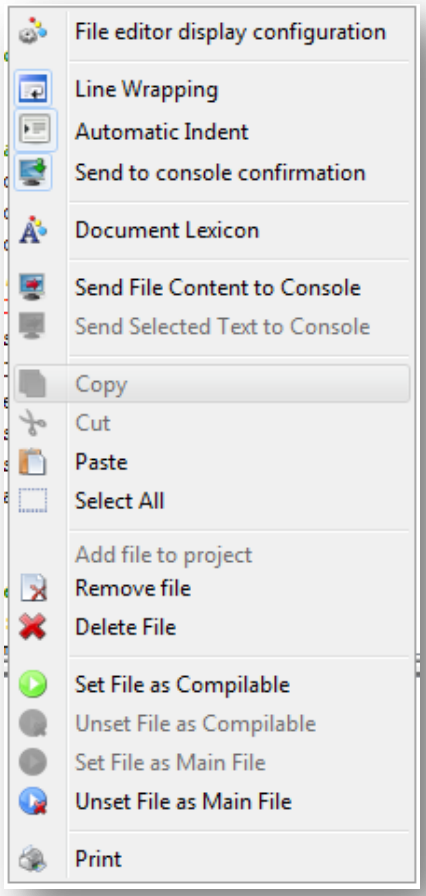



**Figure 52: File editor popup menu**

All these options have been explained before on *Chapter* 3.

The available accessibility shortcuts for File Editor will be further explained in *Chapter 10*.

# 6. TOOL BAR

In the toolbar are displayed some items related with files and projects, commands defined by user to be run in console and commands defined by user to run external applications:

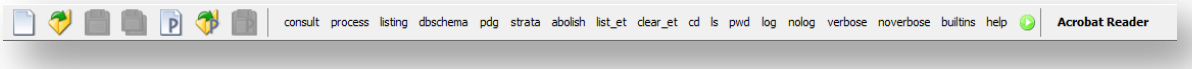



**Figure 53: Tool bar**

Next, we further describe each one of the previous components:

-  : Creates a new file.
-  : Opens a file.
-  : Saves current file.
-  : Saves all opened files.
-  : Creates a new project.
-  :Opens a project.
-  : Saves current project.
- The following items are commands configured by user that run commands on shell (explained on *chapters 3.5.9* and *13.3.2).*
-  : Sends file content to console.
- The following items are commands configured by user that run external applications (explained on *chapters 3.5.9* and *13.3.2).*

# 7. CONSOLE PANEL

At console panel the user can work with the shell he connects to *ACIDE – A Configurable IDE* (explained on *Chapter 3.5.5*). An example of console panel connected with *DES*:

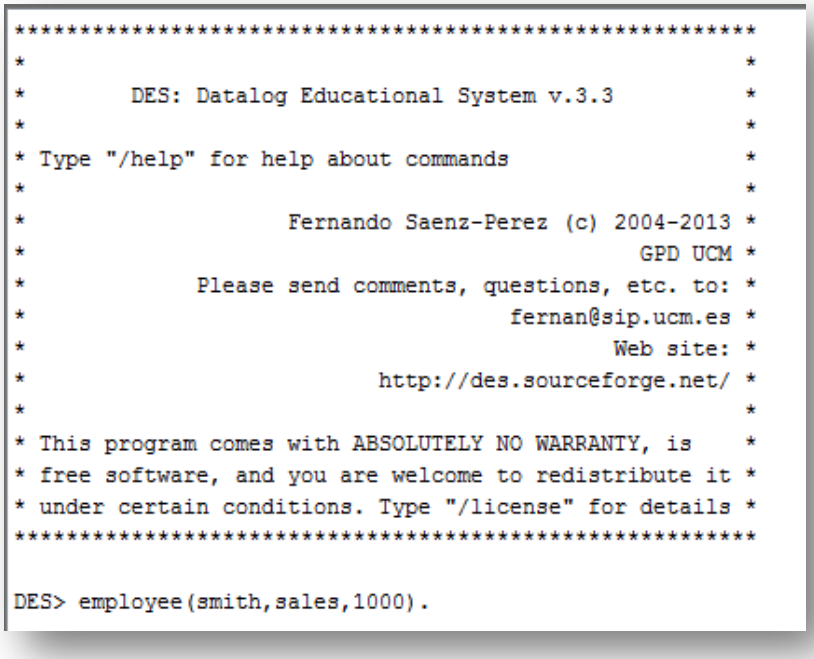



**Figure 54: Console panel**

The *popup menu* is as follows:

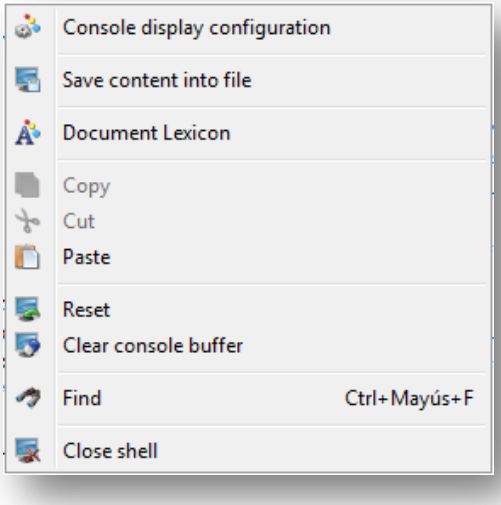



**Figure 55: Console panel popup menu**

All the options have been explained before in *Chapter 3.*

The user can send commands to the shell in different ways. As explained before, user can send the selected text or the content of a file to the sell. Also he can configure the toolbar with buttons which send commands to shell. A new performance of this version is that user can configure the *Menu Bar* to build buttons that send commands to shell in the same way that the toolbar buttons. The default buttons of *ACIDE – A Configurable IDE* send special commands that will be further explained in *Chapter 12*.

# 8. DATABASE PANEL

The database panel shows the metadata of your computer's databases on the lower left corner of the screen.

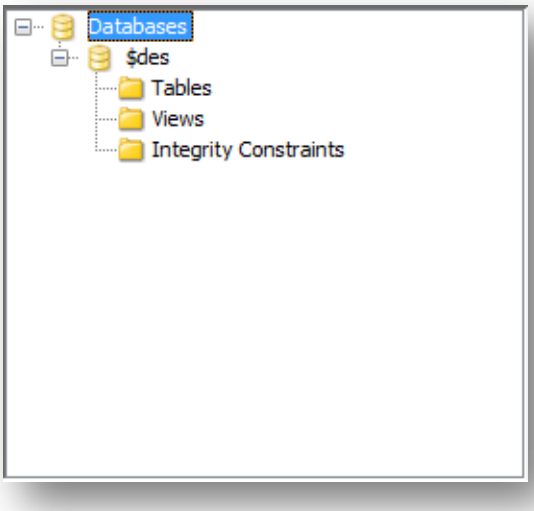



**Figure 56: Database panel**

This panel can be connected with the *DES* or *ODBC* connections of your computer. The user can choose the connection in the *configuration* menu, submenu *database panel*. Nodes can be expanded with double click or with one click on the node and one more on the *"+"* button.

## 8.1. DATABASES NODE

This is the root node of the database panel, below it all the databases connected will be showed.

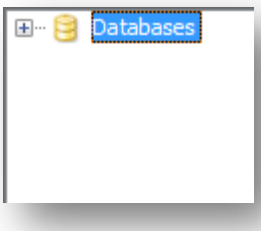



**Figure 57: Databases node**

The *popup menu* of this node is the next:

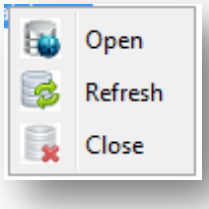



Figure 58: Databases node popup menu

Next we further detail each one of the components of the *popup menu:*

### 8.1.1. OPEN

With this option user can connect the panel with other database. The following window is displayed:

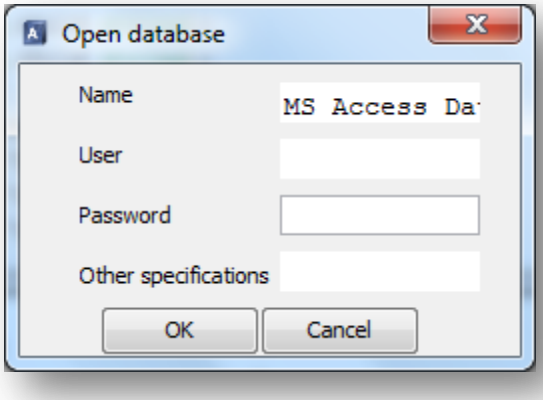



Figure 59: Open database

### 8.1.2. REFRESH

All the *database panel* will be refreshed and user will see all the modifications made with it.

### 8.1.3. CLOSE

The *database panel* will be closed.

## 8.2. DATABASE NODE

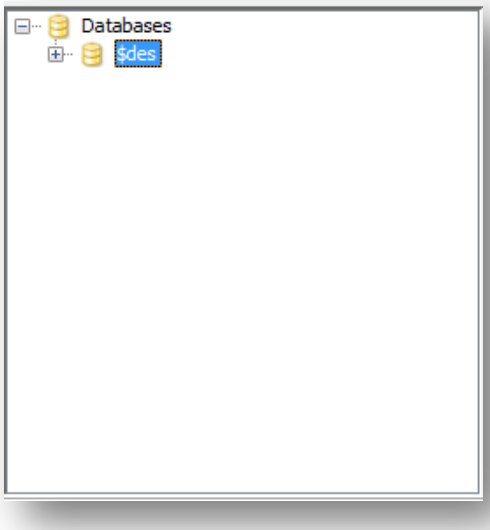



Figure 60: Database node

All the databases opened on this panel are showed in this level of the tree. With the contextual menu user can do the following actions:

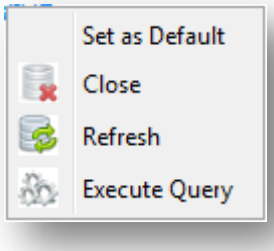



Figure 61: Database node popup menu

### 8.2.1. SET AS DEFAULT

If the shell is connected to *DES*, this option will set this database as the database in use for the following commands.

### 8.2.2. CLOSE

It will close the connection with the database.

### 8.2.3. REFRESH

It will refresh the database node.

### 8.2.4. EXECUTE QUERY

It will displays a window with a text field to input queries in *SQL* in the database.

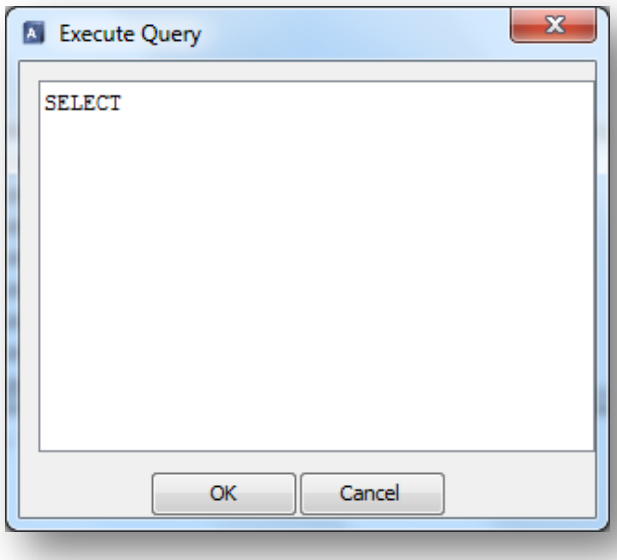



**Figure 62: Execute query**

When user clicks on *"OK"* button the results are showed on the *Data View. Data View* will be further explained in *Chapter 8.4.5*.

Expanding this node there will be three folders below it: *tables, views,* and *integrity constraints.*

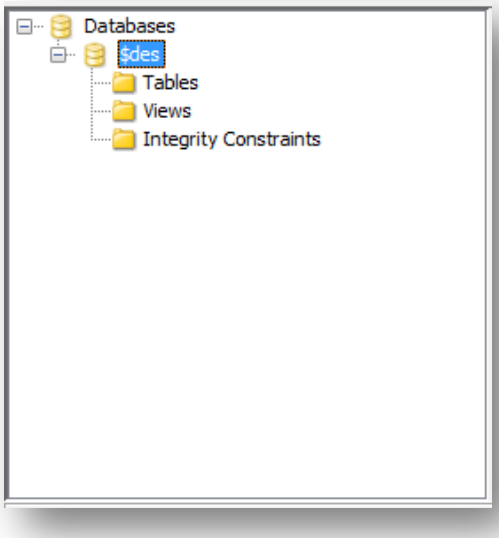



**Figure 63: Expanding database node**

## 8.3. TABLES NODE

The childrens of this node will be all the tables of this database. Its *popup menu* is:

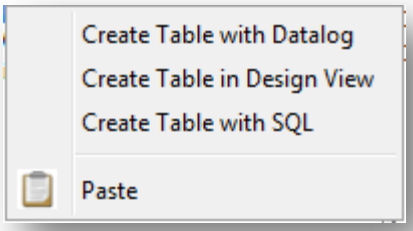



**Figure 64: Tables node popup menu**

### 8.3.1. CREATE TABLE WITH DATALOG

This option is only enabled if the panel is connected with *DES.* The user can create a table with a *Datalog* command in a window similar to the window of *Execute query* action (*Chapter 8.2.4*).

### 8.3.2. CREATE TABLE WITH DESIGN VIEW

With this option the user can create a new table usign a design table with four columns to choose: *name of the field, type, primary key* and *not null:*

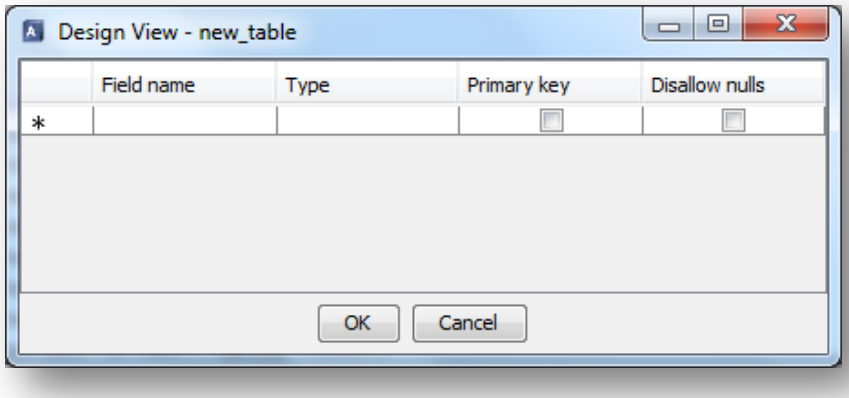



**Figure 65: New table**

With the *"*"* new rows can be inserted in the design table. If you want to make a field part of the primary key you have to mark the checkbox of that column. This option make impossible to mark the *Disallow nulls* option.

### 8.3.3. CREATE TABLE WITH SQL

It displays a window like the *"Execute query"* (*Chapter 8.2.4*) window where the use can create a table with *SQL* commands.

### 8.3.4. PASTE

This option will create a new table with the schema or with the schema and data that the user has copied before from another table of the panel.

## 8.4. TABLE NODE

If the panel is connected with *DES* nodes of this type will show the name of the table and all the information of the fields. However, if the panel is connected with *ODBC*, will only show the name of the table.

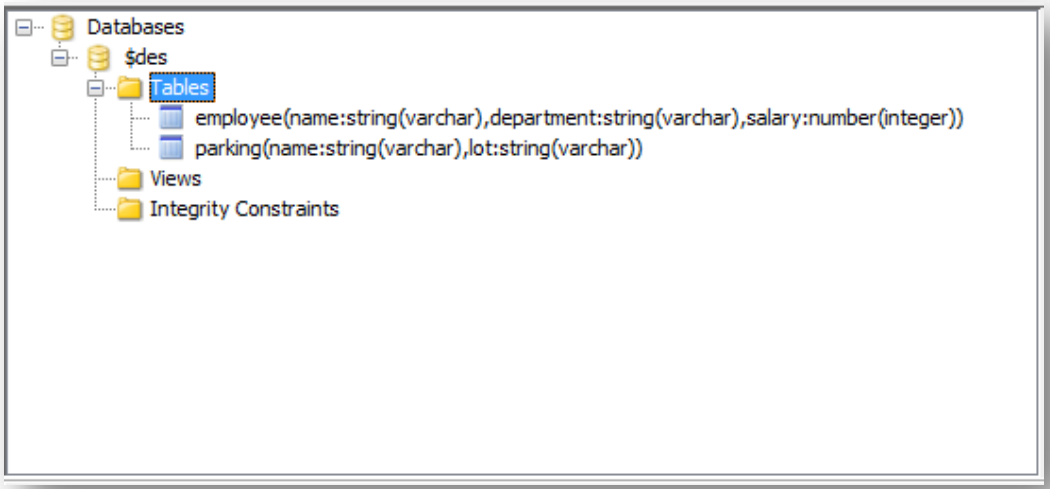



**Figure 66: Table node**

With the contextual menu of this node you can make the following actions:

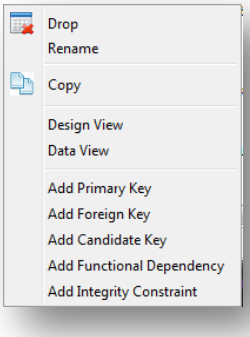



**Figure 67: Table node popup menu**

### 8.4.1. DROP

This action will drop the table.

### 8.4.2. RENAME

The user can change the name of the table with this menu item.

### 8.4.3. COPY

With this option the user can choose between copying only the schema or copying the schema and the data.

### 8.4.4. DESIGN VIEW

It will display the *Design view* of the selected table where the user can make changes on it, add columns, change the primary key and go on.

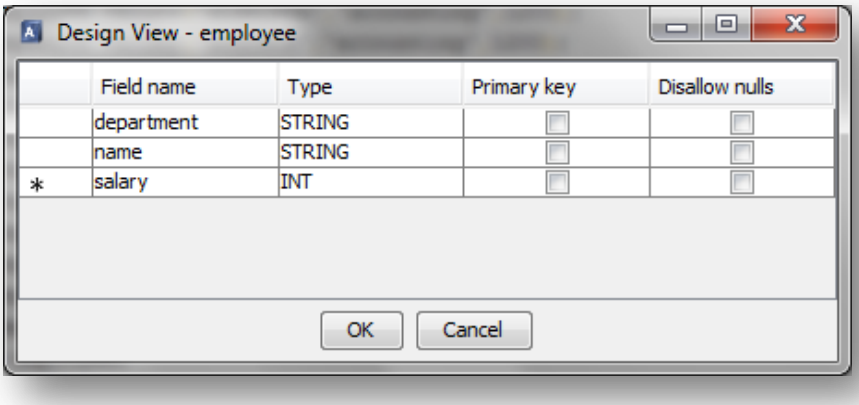



**Figure 68: Design view**

Clicking on *"OK"* button, changes will be applied. If an error occurs, the table will be restored to its previous schema.

### 8.4.5. DATA VIEW

Displays the following window which shows the data contained in the selected table or view, where the simbol "►" indicates the current record.

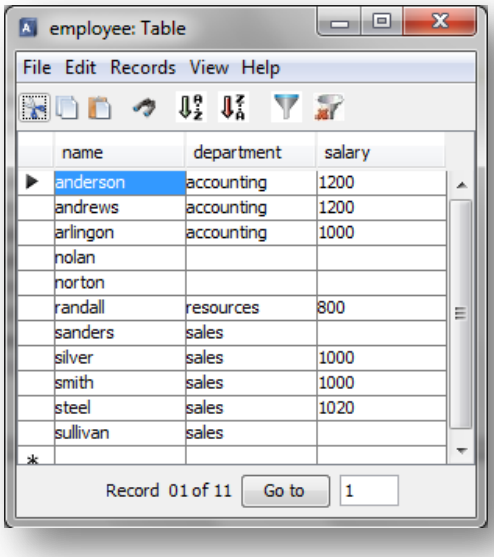



**Figure 69: Data view**

If it is opened for a view, the modification is not allowed. Also, if a data view for the selected table is already opened, only another read-only data view can be opened.

#### 8.4.5.1. ACTIONS PERMITTED ON THE GRID

- **Key navegation:**
  - **Up arrow:** selects previous record.
  - **Down arrow:** selects next record.
  - **Ctrl + Home:** selects the first record.
  - **Ctrl + End:** selects the last record.
  - **Tab:** selects next field.
  - **Tab + Shift:** selects previous field.
- **Sort:**
  - Clicking on the column header, rows will be sorted ascending: the first record displayed will be the record with the lowest value for

this field. Pressing successively on the same field will change the sorting direction.

- Presentation:
    - The user is able to move the columns by clicking on the column name and dragging it to its new location.

#### 8.4.5.2. MENU BAR

##### 8.4.5.2.1. FILE MENU

It contains the following menu items for the files management:

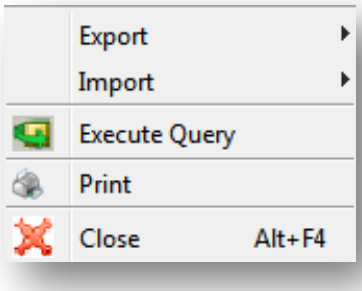
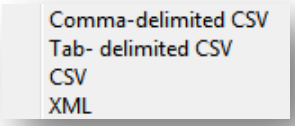



**Figure 70: Data view file menu**

Next, all the previous menu items will be further explained:

###### 8.4.5.2.1.1. EXPORT

It contains the following menu items for the files management:




- **Comma-delimited CSV:** It opens a dialog box to select a file. A text file will be created with all the records of the grid and all their fields in the order they appear in the grid, separated by commas.
- **Tab-delimited CSV:** Same as comma-delimited CSV, but the separator character between fields is the *tab.*
- **CSV:** This opens a dialog box where user can write the separator character and proceed to select the file and save the data.

- **XML:** This opens a dialog box to select a file. A *XML* file will be created with the following structure:

```
<DATA>
<ROW>
<col>value</col>
</ROW>
</DATA>
```

#### 8.4.5.2.1.2. IMPORT

It contains the same menu items as *"Export"* menu item:

- **Comma-delimited CSV:** This opens a dialog box to select a file. For each line of the text file the value that corresponds to the field appears in the grid. Each line will be inserted in the table as described before.
- **Tab-delimited CSV:** same as comma-delimited CSV but the separator character between fields is the *tab.*
- **CSV:** this opens a dialog box where user can write the separator character and proceed to select the file and load the data.
- **XML:** This opens a dialog box to select a file. It will read the *XML* file with the structure indicated above. Each row of data of the *XML* file will be inserted in the table.

#### 8.4.5.2.1.3. EXECUTE QUERY

It displays a dialog in which the user will type down the query he wants to perform:

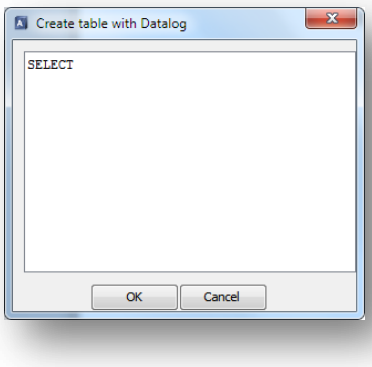



**Figure 71: Execute query**

#### 8.4.5.2.1.4. PRINT

It displays the print window to print the grid.

#### 8.4.5.2.1.5. CLOSE

Close the data view window. It also can be closed with the key combination *"Alt + F4".*

### 8.4.5.2.2. EDIT MENU

It contains the following menu items for the common grid editor management:

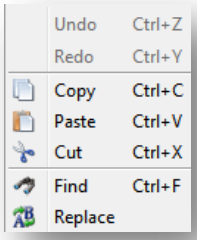



**Figure 72: Data view edit menu**

#### 8.4.5.2.2.1. UNDO

Undoes the updates in the grid. It also can be done with the key combination *"Ctrl + Z".*

#### 8.4.5.2.2.2. REDO

Redoes the updates in the grid. It also can be done with the key combination *"Ctrl + Y".*

#### 8.4.5.2.2.3. COPY

Copies the selected text active field from the grid and put it into the System clipboard. It also can be done with the key combination *"Ctrl+ C"* or with the icon  of the icon bar.

#### 8.4.5.2.2.4. PASTE

Pastes the text stored in the System clipboard in the current position of the active field in the grid. It also can be done with the key combination *"Ctrl + V"* or with the icon  of the icon bar.

#### 8.4.5.2.2.5. CUT

Cuts the selected text active field from the grid and put it into the System clipboard. It also can be done with the key combination *"Ctrl + X"* or with the icon  of the icon bar.

#### 8.4.5.2.2.6. FIND

Displays the search text window for the *data view*.

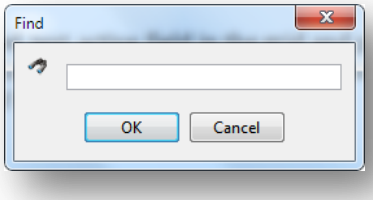



**Figure 73: Data view search window**

It also can be done with the key combination *"Ctrl + F"* or with the icon  of the icon bar.

#### 8.4.5.2.2.7. REPLACE

Displays the replace text window of the *data view:*

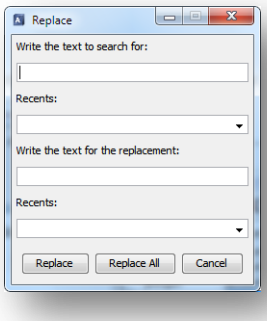



**Figure 74: Data view replace window**

When a general replacement is performed, it displays the following dialog to the user informing of the *number of replacements:*

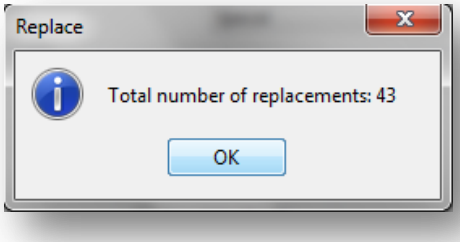



**Figure 75: Data view number of replacements**

#### 8.4.5.2.3. RECORDS MENU

It contains the following menu items for the common grid editor management:

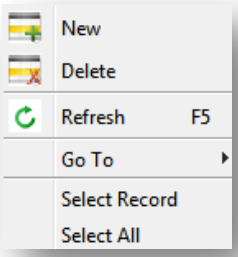



**Figure 76: Data view records menu**

Next, all the previous menu items will be further explained:

##### 8.4.5.2.3.1. NEW

Inserts a new record in the grid. The values of the new record must be written at the last row of the grid. It also can be done clicking in the cell with the "*" icon.

##### 8.4.5.2.3.2. DELETE

Deletes the selecetd record from the grid.

##### 8.4.5.2.3.3. REFRESH

Updates the view of the grid.

##### 8.4.5.2.3.4. GO TO

It contains the following menu items for the common grid editor management:

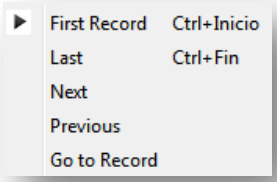
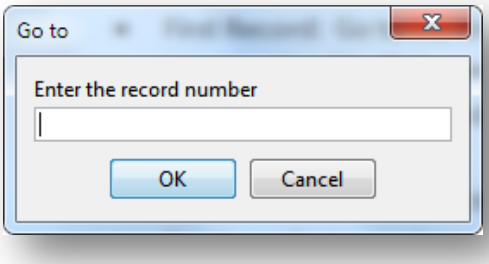



Figure 77: Data view go to menu

- **First record:** Goes to the first record. It also can be done with the key combination *"Ctrl+ home"*.
- **Last:** Goes to the last record. It also can be done with the key combination *"Ctrl + end".*
- **Next:** Goes to next record. It also can be done with the *up arrow key*.
- **Previous:** Goes to previous record. It also can be done with the *down arrow key*.
- **Go to record:** It displays a dialog window where the user will type down the row number he wants go to.




#### 8.4.5.2.3.5. SELECT RECORD

Selects the current record from the grid.

#### 8.4.5.2.3.6. SELECT ALL

Selects all the records from the grid.

### 8.4.5.2.4. VIEW MENU

It contains the following menu items for the common grid editor management:

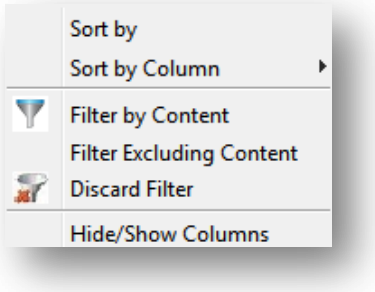



**Figure 78: Data view view menu**

Next, all the previous menu items will be further explained:

#### 8.4.5.2.4.1. SORT BY

It displays a window with a grid to select the table field by which sort the table, and the criteria *"Ascending"* or *"Descending".*

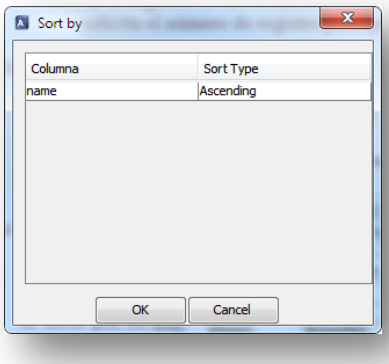
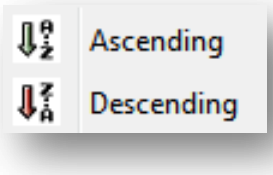



**Figure 79: Data view sort by window**

#### 8.4.5.2.4.2. SORT BY COLUMN

It contains the following menu items for the common grid editor management:




- **Ascending:** it will order the grid ascending by the selected column. It also can be done with the icon  of the icon bar.

- **Descending:** it will order the grid ascending by the selected column. It also can be done with the icon  of the icon bar.

#### 8.4.5.2.4.3. FILTER BY CONTENT

Filters the grid by the content of the selected field. It also can be done with the icon  of the icon bar.

#### 8.4.5.2.4.4. FILTER EXCLUDING CONTENT

Filters records that do not contain the content of the selected field.

#### 8.4.5.2.4.5. DISCARD FILTER

Removes the filter. It also can be done with the icon  of the icon bar.

#### 8.4.5.2.4.6. HIDE/SHOW COLUMNS

It will display a window with a grid to select the columns visibility:

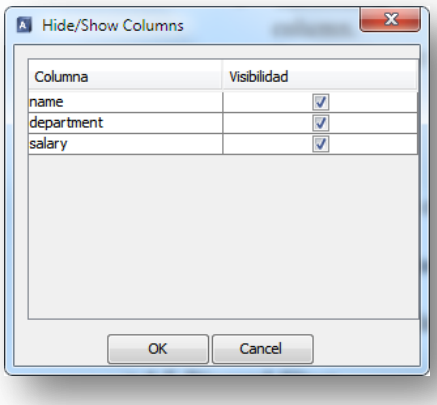



**Figure 80: Data view hide/show columns**

### 8.4.5.2.5. HELP MENU

Contains the following menu items:

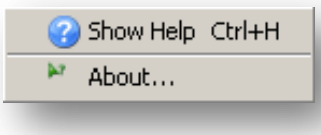



**Figure 81: Data view help menu**

Next, the previous menu options are further explained

#### 8.4.5.2.5.1. SHOW HELP

Links directly to the user's manual of *DES-ACIDE*.

#### 8.4.5.2.5.2. ABOUT US

Displays the following window with some extra information about the application:

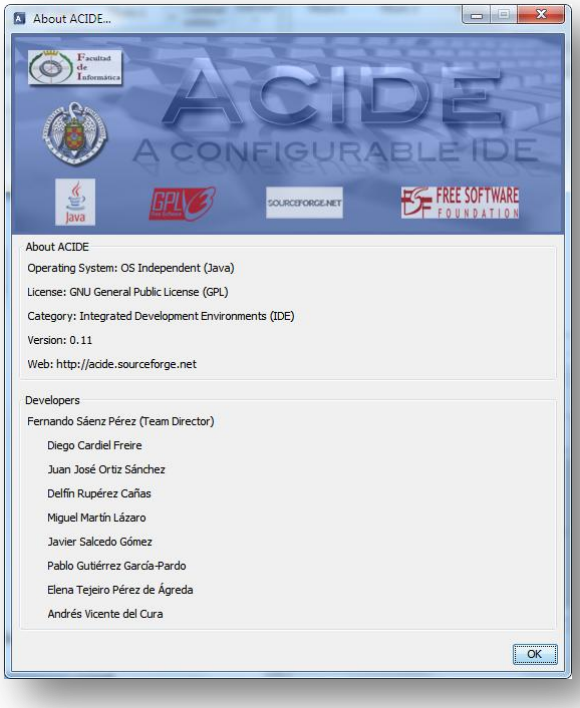



**Figure 82: Data view about us window**

### 8.4.6. ADD PRIMARY KEY

A primary key constraint specifies that no two tuples have the same values for a given set of columns. To define a primary key constraint user has to type *:-pk(*name_of_the_relation, *[*column_name_list*]).*

This menu option displays a window where user can write the *datalog* command to create a primary key in the selected table:

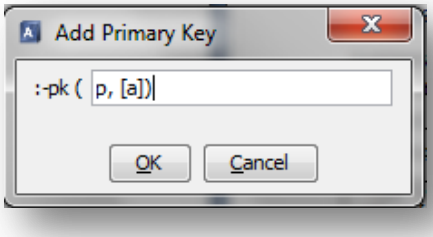



**Figure 83: Add primary key**

### 8.4.7. ADD FOREIGN KEY

A foreign key constraint specifies that the values in a given set of columns of a relation must exist already in the columns declared in the primary key constraint of another relation. To define a foreign key constraint in a relation the user has to type *:-fk(*name_of_the_target_relation,*[*name_of_the_column_foreign_key*]*,name_of_source_relation,*[*name_of_source_column*]).*

This menu option displays a window where user can write the *datalog* command to create a foreign key in the selected table:

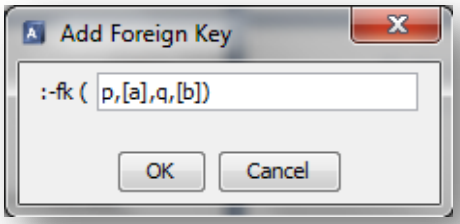



**Figure 84: Add foreign key**

### 8.4.8. ADD CANDIDATE KEY

As a primary key, a candidate key constraint specifies that no two tuples have the same values for a given set of columns. To define a candidate key constraint user has to type *:-ck(*name_of_the_relation, *[*column_name_list*]).*

This menu option displays a window where user can write the *datalog* command to create a candidate key in the selected table:

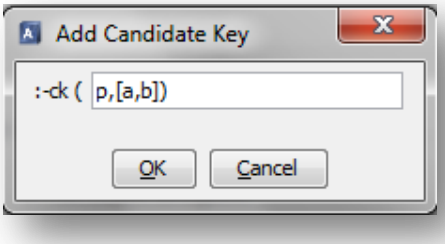



**Figure 85: Add candidate key**

### 8.4.9. ADD FUNCTIONAL DEPENDENCY

A functional dependency constraint specifies that, given a set of attributes *A1,* of a relation *R*, they functionally determine another set *A2*, i.e., each tuple of values of *A1* in *R* is associated with precisely one tuple of values *A2* in the same tuple of *R.* To define a functional dependency constraint user has to type *:-fd (*name_of_the_relation, *[*A1*], [*A2*]).*

This menu option displays a window where you can write the *datalog* command to create a functional dependency in the selected table:

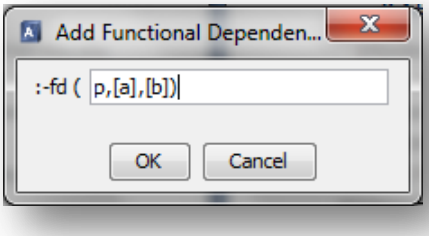



**Figure 86: Add functional dependency**

### 8.4.10. ADD INTEGRITY CONSTRAINT

A integrity constraint is represented with a rule without head. The rule body is an assertion that specifies incosistent data, i.e., should this body can be proved, an inconsistency is detected and reported to the user.

Declaring such integrity constraints implies to change your mind w.r.t usual consistency constraints as domain constraints in *SQL*. For instance, to specify that a column **c** of a table **t** can take values between two integers one can use the *SQL* clause **CHECK** in the creation of the table as follows:

```
CREATE TABLE t(c INT CHECK (c BETWEEN 0 AND 10));
```

In contrast, in *Datalog* user can submit the following constraints:

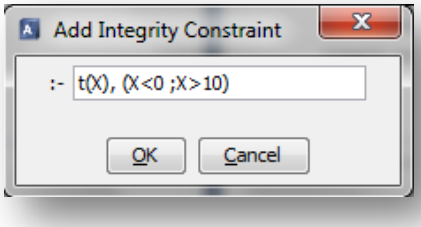



**Figure 87: Add integrity constraint**

Notice that the rule body succeeds for values in **t** out of the interval **[0,10]**. So, an integrity constraint specifies *unfeasible* values rather than feasible.

## 8.5. CHILDREN OF TABLE NODES

Under the table node the user can see the information of the columns and all the constraints like primary key, foreign key and go on.

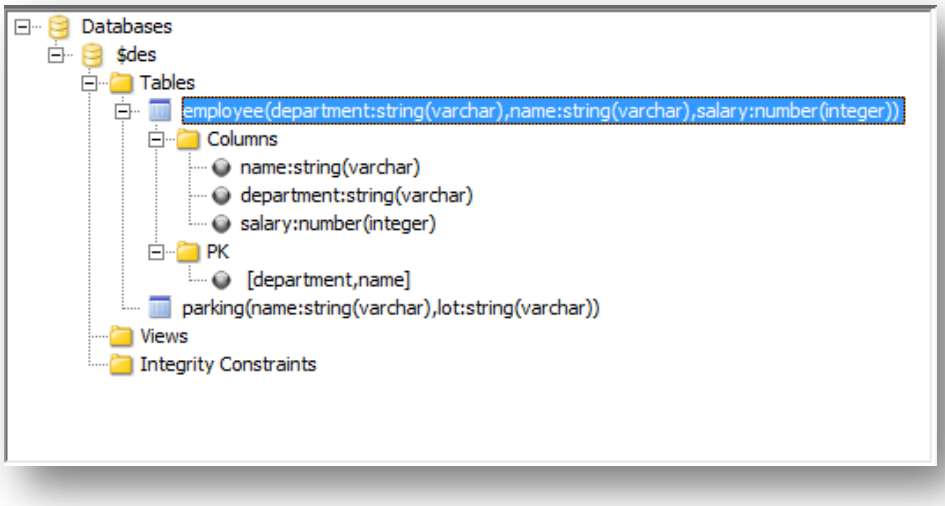
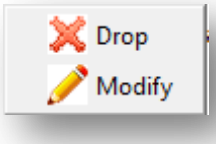



**Figure 88: Children of table nodes**

In the primary key node, and in all the nodes which define a constraint (in the figure the node [department, name]), the user has these options in the *popup menu:*




### 8.5.1. DROP

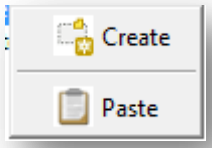
It will drop the restriction.

### 8.5.2. MODIFY

The user can modify the restriction with this action.

## 8.6. VIEWS NODE

The children of this node are all the views of the selected database. Its *popup menu* is the following:




### 8.6.1. CREATE

With the next window the user can create a view, defining it with an *SQL* command:

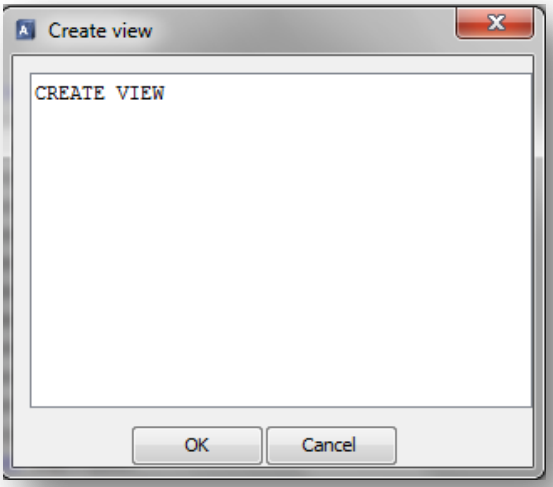



**Figure 89: Create view window**

### 8.6.2. PASTE

A new view will be created with the same schema than the view that had been copied before.

## 8.7. VIEW NODE

This node relates the name and the fields informatino of one view of the selected database.

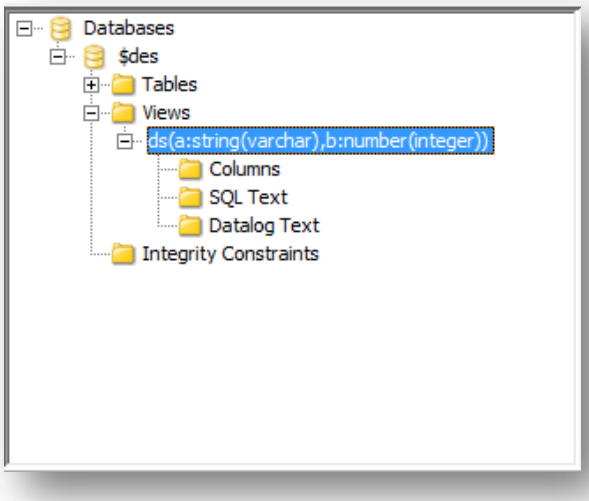



**Figure 90: View node**

Its *popup menu* is as follows:

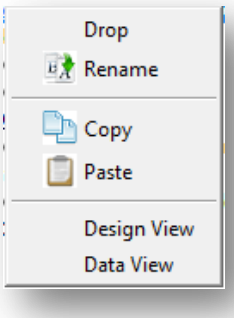



**Figure 91: View node popup menu**

### 8.7.1. DROP

The view will be deleted from the database.

### 8.7.2. RENAME

The user can change the name of the view with this action.

### 8.7.3. COPY

The schema of the selected view will be copied to the clipboard.

### 8.7.4. PASTE

A new view with the schema of the view copied in the clipboard before will be created.

### 8.7.5. DESIGN VIEW

A window with the *SQL* text of the view will be showed.

### 8.7.6. DATA VIEW

It is almost identical to *Data view* for Tables, explained before on *Chapter 8.4.5*.

## 8.8. COLUMNS NODES

The children of this node are all the columns of the selected view.

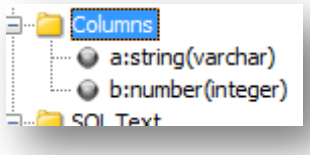



**Figure 92: Columns nodes**

## 8.9. SQL TEXT AND DATALOG TEXT NODES

These nodes show the *SQL* and *Datalog* commands of the view definition.

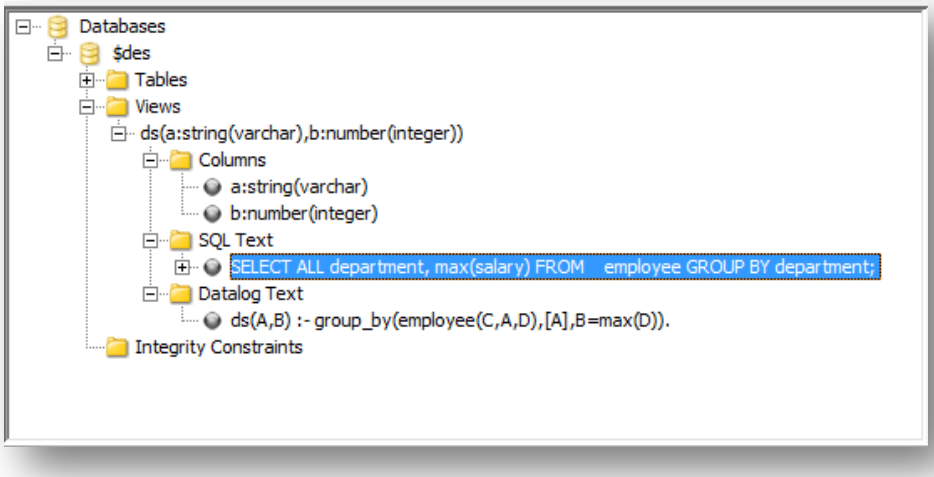



**Figure 93: SQL and Datalog text nodes**

The user can make double click in this node and a window with this text will appear where user can modify it.

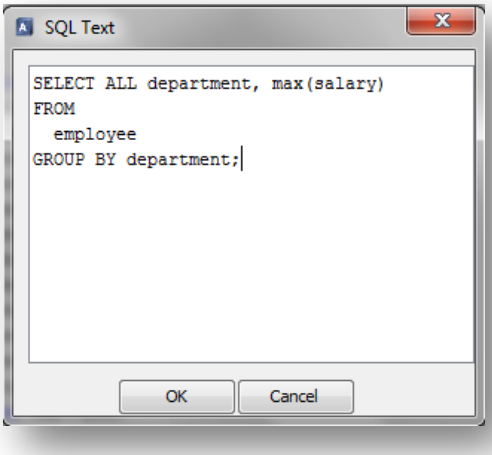



**Figure 94: SQL text**

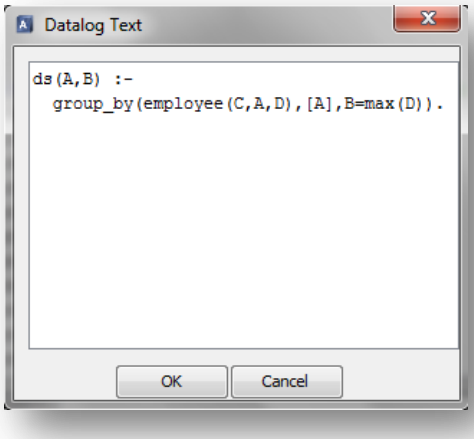
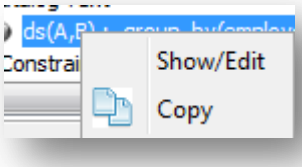



**Figure 95: Datalog text**

With the *popup menu* user can show and edit too, and copy the definition text.



# 9. STATUS BAR

The *Status Bar* contains some information about the active file, the current project and go on. It is as follows:

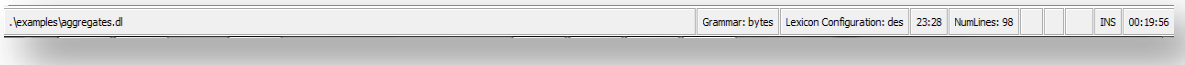



**Figure 96: Status bar**

Next, we further describe all the components:

- **Panel 1:** the *status message* is displayed. It shows the path and name of the active file in the *File Editor*.
- **Panel 2:** the *syntactic configuration* shows the name of the grammar applied to the current project.
- **Panel 3:** the *lexicon configuration* shows the name of the lexicon applied to the current project.
- **Panel 4:** shows the line and column where the caret is.
- **Panel 5:** *BLOQ MAYUS* status.
- **Panel 5:** *BLOQ NUM* status.
- **Panel 6:** *BLOQ SCROLL* status.
- **Panel 7:** writing mode: *INSERT* or *OVERWRITE.*
- **Panel 8:** the *System* clock.

# 10. ACCESSIBILITY SHORTCUTS

The application offers some accessibility shortcuts to wrapper common user actions such as:

- **F3 + Selected text:** performs the *forward* text search with the selected text in the file editor, in the console panel or in the data view window.
- **F3 + Shift + Selected text:** performs the *backward* text search with the selected text in the file editor, in the console panel or in the data view window.
- **Mouse wheel:** performs the vertical scroll line by line in the file editor and console panel.
- **Control + mouse wheel:** performs the zoom effect for the font size in the file editor and console panel.

Others accessibility shortcuts depends on the language of the application.

## 10.1. ACCESSIBILITY SHORTCUTS IN ENGLISH

Shortcuts in *File menu:*

- **Ctrl + N:** Creates new file.
- **Ctrl + O:** Opens a file.
- **Ctrl + S:** Saves active file in the file editor.
- **Ctrl + Shift + S:** Saves all files in the file editor.
- **Ctrl + P :** Prints active file in the file editor.
- **Alt + X:** Closes the application.

Shortcuts in *Edit menu:*

- **Ctrl + Z:** Undo the last action.
- **Ctrl + Y:** Redo the last change.
- **Ctrl + C:** Copy the selected text to the System clipboard.
- **Ctrl + X:** Cuts the selected text to the System clipboard.
- **Ctrl + V:** Paste the test in the System clipboard.

- **Ctrl + E:** Select all the text in the active file of the file editor.
- **Ctrl + F:** Opens the search window.
- **Ctrl + R:** Opens the replace window.

Shortcuts in *Project menu:*

- **Alt + Shift + N:** Creates a new project.
- **Alt + Shift + O:** Opens a project.
- **Alt + Shift + S:** Save the opened project.
- **Alt + Shift + A:** Adds a file to the opened project.
- **Alt + C:** Compiles the opened project.
- **Alt + E:** Executes the opened project.

Shortcuts in *View menu:*

- **Alt + Shift + L:** Shows the log tab.

Shortcuts in Configuration menu:

- **Ctrl + Shift + L:** Documents lexicon.
- **Ctrl + Shift + X:** Modifies the lexicon.
- **Ctrl + Shift + T:** Creates a new grammar.
- **Ctrl + Shift + A:** Actives line wrapping.
- **Ctrl + Shift + F:** Opens the search in console window.
- **Alt + S:** Changes language to *Spanish.*
- **Alt + E:** Changes language to *English*.

Shortcuts in *Help menu:*

- **Ctrl + H:** Shows this document.

Shortcuts in *Data view:*

- **Up arrow:** goes to previous record.
- **Down arrow:** goes to next record.
- **Tab:** goes to next field.
- **Shift + Tab:** goes to previous field.

- **Alt + F4:** closes the *Data view* window.
- **Ctrl + Z:** undoes the updates in the grid.
- **Ctrl + Y:** redoes the last undo in the grid.
- **Ctrl + C:** copies the selected text active field from the grid to the System clipboard.
- **Ctrl + V:** pastes the text stored in the System clipboard in the current position of the active field in the grid.
- **Ctrl + X:** cuts the selected text active field from the grid to the System clipboard.
- **Ctrl + F:** shows the search text window for the *Data view.*
- **F5:** refresh the view of the grid.
- **Ctrl + home:** goes to the first record.
- **Ctrl + end:** goes to the last record.
- **Ctrl + H:** links directly to the present document.

Shortcuts in *Menu configuration:*

- **Up arrow:** selects previous object.
- **Down arrow:** selects next object.
- **Ctrl + Home:** selects the first object.
- **Ctrl + End:** selects the last object.
- **Tab:** selects next attribute.
- **Tab + Shift:** selects previous attribute.
- **Esc:** deselects the selected object.

## 10.2. ACCESSIBILITY SHORTCUTS IN SPANISH

Shortcuts in *File menu:*

- **Ctrl + N:** Creates new file.
- **Ctrl + O:** Opens a file.
- **Ctrl + G:** Saves active file in the file editor.
- **Ctrl + Shift + G:** Saves all files in the file editor.
- **Ctrl + P :** Prints active file in the file editor.

- **Alt + X:** Closes the application.

Shortcuts in *Edit menu:*

- **Ctrl + Z:** Undo the last action.
- **Ctrl + Y:** Redo the last change.
- **Ctrl + C:** Copy the selected text to the System clipboard.
- **Ctrl + X:** Cuts the selected text to the System clipboard.
- **Ctrl + V:** Paste the test in the System clipboard.
- **Ctrl + E:** Select all the text in the active file of the file editor.
- **Ctrl + B:** Opens the search window.
- **Ctrl + L:** Opens the replace window.

Shortcuts in *Project menu:*

- **Alt + Shift + N:** Creates a new project.
- **Alt + Shift + O:** Opens a project.
- **Alt + Shift + S:** Save the opened project.
- **Alt + Shift + A:** Adds a file to the opened project.
- **Alt + C:** Compiles the opened project.
- **Alt + E:** Executes the opened project.

Shortcuts in *View menu:*

- **Alt + Shift + L:** Shows the log tab.

Shortcuts in Configuration menu:

- **Ctrl + Shift + L:** Documents lexicon.
- **Ctrl + Shift + X:** Modifies the lexicon.
- **Ctrl + Shift + T:** Creates a new grammar.
- **Ctrl + Shift + A:** Actives line wrapping.
- **Ctrl + Shift + F:** Opens the search in console window.
- **Alt + S:** Changes language to *Spanish.*
- **Alt + E:** Changes language to *English*.

Shortcuts in *Help menu:*

- **Ctrl + H:** Shows this document.

*Shortcuts in Data view:*

- **Up arrow:** goes to previous record.
- **Down arrow:** goes to next record.
- **Tab:** goes to next field.
- **Shift + Tab:** goes to previous field.
- **Alt + F4:** closes the *Data view* window.
- **Ctrl + Z:** undoes the updates in the grid.
- **Ctrl + Y:** redoes the last undo in the grid.
- **Ctrl + C:** copies the selected text active field from the grid to the System clipboard.
- **Ctrl + V:** pastes the text stored in the System clipboard in the current position of the active field in the grid.
- **Ctrl + X:** cuts the selected text active field from the grid to the System clipboard.
- **Ctrl + F:** shows the search text window for the *Data view.*
- **F5:** refresh the view of the grid.
- **Ctrl + home:** goes to the first record.
- **Ctrl + end:** goes to the last record.
- **Ctrl + H:** links directly to the present document.

Shortcuts in *Menu configuration:*

- **Up arrow:** selects previous object.
- **Down arrow:** selects next object.
- **Ctrl + Home:** selects the first object.
- **Ctrl + End:** selects the last object.
- **Tab:** selects next attribute.
- **Tab + Shift:** selects previous attribute.
- **Esc:** deselects the selected object.

# 11.ACIDE VARIABLES

The application supports some variables in the *Console Panel, External Applications Tool Bar* and the shell loaded in the *console panel* such as:

- **$activeFile$:** references the current active file in the file editor panel.
- **$activeFileName$:** references just the current active name file in the file editor panel.
- **$activeFilePath$:** references just the current active path file in the file editor panel without including neither file name nor file extension.
- **$activeFileExt$:** references just the current active extension file in the file editor panel.
- **$mainFile$:** references the file in the file editor panel that has been marked as *MAIN* file.
- **$mainFilePath$:** references the just file path in the file editor panel that has been marked has *MAIN* file without including neither file name nor file extension.
- **$mainFileExt$:** references just the file extension in the file editor panel that has been marked as *MAIN* file.

# 12.ACIDE DEFAULT COMMANDS

As explained in *Chapter 3.5.8,* with the menu configuration the user can assign to the application the actions that will be executed when menu items are pressed down. All these commands start with *"$".* The commands assigned by default to *ACIDE – A Configurable IDE* menu items are:

- *File menu:*
  - **$NEW_FILE:** Creates a new file in the file editor.
  - **$OPEN_FILE:** Opens a file in the file editor.
  - **$OPEN_ALL_FILES:** Opens all the files of the active project.
  - **$CLOSE_FILE:** Closes the active file in the file editor.
  - **$CLOSE_ALL_FILES:** Closes all files in the file editor.
  - **$SAVE_FILE:** Saves the active file.
  - **$SAVE_FILE_AS:** Saves the active file in a different path.
  - **$SAVE_ALL_FILES:** Saves all the opened files in the file editor.
  - **$PRINT_FILE:** Prints the active file in the file editor.
  - **$EXIT_FILE:** Closes the application.
- *Edit menu:*
  - **$UNDO:** Undoes the last action.
  - **$REDO:** Redoes the last undone action.
  - **$COPY:** Copies the selected text to the System clipboard.
  - **$PASTE:** Pastes the text in the System clipboard.
  - **$CUT:** Cuts the selected text to the System clipboard.
  - **$SELECT_ALL:** Select all the text in the active file of the file editor.
  - **$GO_TO_LINE:** Opens a window where user can type down the number of line where he wants to go.
  - **$SEARCH:** Opens the search window.
  - **$REPLACE:** Opens the replace window.
- *Project menu:*
  - **$NEW_PROJECT:** Creates a new project.
  - **$OPEN_PROJECT:** Opens a project in the application.
  - **$CLOSE_PROJECT:** Closes the opened project in the application.

- **$SAVE_PROJECT:** Saves the active project in the application.
- **$SAVE_PROJECT_AS:** Saves the active project in the application with a different path.
- **$ADD_OPENED_FILES:** Adds all opened files in the application to the active project.
- **$NEW_PROJECT_FILE:** Creates a new file and adds it to the active project.
- **$ADD_FILE:** Adds the active file in the file editor to current project.
- **$REMOVE_FILE:** Removes the active file in the file editor from the current project.
- **$DELETE_FILE:** Deletes the active file from the current project and from disk.
- **$ADD_FOLDER:** Adds a folder to the current project.
- **$REMOVE_FOLDER:** Removes the selected folder from the current project.
- **$COMPILE:** Compiles the current project.
- **$EXECUTE:** Executes the current project.
- **$SET_COMPILABLE_FILE:** Sets compilable the selected file.
- **$UNSET_COMPILABLE_FILE:** Unsets compilable the selected file.
- **$SET_MAIN_FILE:** Sets as main file the selected file.
- **$UNSET_MAIN_FILE:** Unsets as main file the selected file.

- *View menu:*
  - **$SHOW_LOG_TAB:** Shows the log tab.
  - **$SHOW_EXPLORER_PANEL:** Shows or hides the explorer panel.
  - **$SHOW_CONSOLE_PANEL:** Shows or hides the console panel.
  - **$SHOW_DATABASE_PANEL:** Shows or hides the database panel.
- *Configuration menu:*
  - *Lexicon submenu:*
    - **$NEW_LEXICON:** Opens a window where user type down the name for the new lexicon.
    - **$DOCUMENT_LEXICON:** Loads the lexicon configuration file in the active file of the file editor.
    - **$MODIFY_LEXICON:** Open the lexicon configuration window.

        - **$DEFAULT_LEXICON:** Shows the default lexicon configuration window.
    - *Grammar submenu:*
        - **$NEW_GRAMMAR:** Opens the new grammar configuration window.
        - **$LOAD_GRAMMAR:** Loads a grammar configuration.
        - **$MODIFY_GRAMMAR:** Displays the modify grammar configuration window.
        - **$SAVE_GRAMMAR:** Saves the current grammar configuration into a file.
        - **$SAVE_GRAMMAR_AS:** Saves the current grammar configuration into a file with a different path.
        - **$SET_PATHS:** Displays the set paths window.
    - **$COMPILER:** Displays the compiler configuration window.
    - *File editor submenu:*
        - **$FILE_EDITOR_DISPLAY_OPTIONS:** Displays the file editor display configuration window.
        - **$AUTOMATIC_INDENT:** Enables or disables the automatic indent in the file editor.
        - **$LINE_WRAPPING:** Enables or disables the line wrapping in the file editor.
        - **$MAXIMUM_LINES:** Asks to the user for the maximum number of lines to send to the console panel.
        - **$SEND_CONSOLE_CONFIRMATION:** Enables or disables the confirmation request when user sends contents to console panel.
    - *Console submenu:*
        - **$CONFIGURE_CONSOLE:** Opens the console configuration window.
        - **$CONSOLE_DISPLAY_OPTIONS:** Displays the console display configuration window.
        - **$SAVE_CONSOLE_CONTENT:** Saves the console content into a file.

        - **$DOCUMENT_CONSOLE:** Loads a lexicon configuration into the console panel.
        - **$SEARCH_CONSOLE:** Opens the search in console window.
        - **$CLOSE_CONSOLE:** Closes the console.
    - *Database panel submenu:*
        - **$DES_PANEL:** Selects the *DES* conection in database panel.
        - **$ODBC_PANEL:** Selects the *ODBC* conection in database panel.
    - *Language submenu:*
        - **$SPANISH:** Selects *Spanish* as language.
        - **$ENGLISH:** Selects *English* as language.
    - *Menu submenu:*
        - **$NEW_MENU:** Displays the new menu configuration window.
        - **$LOAD_MENU:** Loads a menu configuration and applies it to application.
        - **$MODIFY_MENU:** Displays the menu configuration window for modifying.
        - **$SAVE_MENU:** Saves the current menu configuration.
        - **$SAVE_MENU_AS:** Saves the current menu configuration with a different path.
    - *Tool bar submenu:*
        - **$NEW_TOOLBAR:** Displays the new toolbar configuration window.
        - **$LOAD_TOOLBAR:** Loads a tool bar configuration and applies it to application.
        - **$MODIFY_TOOLBAR:** Displays the tool bar configuration window for modifying.
        - **$SAVE_TOOLBAR:** Saves the current tool bar configuration.
        - **$SAVE_TOOLBAR_AS:** Saves the current tool bar configuration with a different path.
- *Help menu:*
    - **$SHOW_HELP:** Opens this document.
    - **$SHOW_ABOUT_US:** Displays the *About Us* window.

These commands can be assigned to other menu items than are not the default menu items.

# 13. CONFIGURATION OF ACIDE BY CONFIGURATION DOCUMENTS

## 13.1. MANAGERS IN XML FILES

Frequently in *XML* configuration files we found labels of the form *"...Manager"*. These labels contain a type of object called *Manager* that is responsible for handling lists of different types of objects. Inside the labels of a *Manager* there is another label called *"_list"* and that in turn holds another label also called *"_list"*. It is inside this label where user introduces the labels of the objects that make up the list he wants to handle with the *Manager* (could be a list of *String, AcideLexiconTokenGroup,* etc.).

There are java classes for each of the *Managers* in *XML* files, which provide methods to manipulate the lists as adding, removing or getting items from them. *Manager Java classes* have an *ObjectList* type field, which in turn has an *ArrayList* field (where user stores the list of objects) and methods for manipulating that list.

To introduce, delete, reorder, etc. elements of the list, just manually edit the *XML* document and operate on the labels of each object.

## 13.2. PROPERTIES CONFIGURATION

To configure several properties of *ACIDE – A Configurable IDE* there is a file called **configuration.properties** stored in *./configuration*. In this file are stored properties that are not specified in other files. The structure of this file is fixed; user can only edit the values for each field, but he is not able to add new properties or delete any of the existing.

The first line of the properties configuration file is:

```
#ACIDE Configuration
```

The following line shows the date of the last time the user ran this issue of *ACIDE – A Configurable IDE*. Displays the following format:

```
#Mon May 27 18:16:32 CEST 2013
```

The following lines show the property name followed by a *"="* and the value assigned to that property with the following structure:

- **consolePanel.fontName**=name of the font of console panel.
- **workbenchConfiguration**=path to *XML* file that configures the workbench (*Chapter 13.3*)
- **lastOpenedFileDirectory**=the folder was last opened. Used to locate the user in the same folder next time.
- **javacPath**=path of *javac.exe*.
- **jarPath**=path of *jar.exe.*
- **consolePanel.exitCommand**=exit command for console.
- **ed**=
- **consolePanel.fontStyle**=style of the font of console panel.
- **consolePanel.bufferSize**=size of buffer of console panel.
- **previousMenuNewConfiguration**=path to *XML* file that previously configured *ACIDE – A Configurable IDE* menu with the new configuration of version 0.11 (*Chapter 13.3.1*)
- **consolePanel.backgroundColor=**console panel background color (numeric valor).
- **currentMenuConfiguration**=path to *.menuConfig* file that was configurating *ACIDE – A Configurable IDE* menu with the configuration of older versions.
- **consolePanel.shellDirectory**=path to the folder where the *.exe* of console shell is stored.
- **console Panel.shellPath**=path to the *.exe* file of console shell.
- **consolePanel.fontSize**=size of the font of console.
- **previousToolbarConfiguration**=path to *.toolbarConfig* file that previously configured *ACIDE – A Configurable IDE* toolbar (*Chapter 13.3.2*).
- **currentMenuNewConfiguration**= path to *XML* file that configures *ACIDE – A Configurable IDE* menu with the new configuration of version 0.11 (*Chapter 13.3.1*).
- **consolePanel.isechoCommand**=true or false to define the behaviour of echo command at console panel.

- **language**=can be English or Spanish.
- **currentToolbarConfiguration**=path to *.toolbarConfig* file that configures *ACIDE – A Configurable IDE* toolbar (*Chapter 13.3.2*)
- **previousMenConfiguration=**path to *.menuConfig* file that previously configured *ACIDE – A Configurable IDE* menu with the configuration of older versions.
- **lastOpenedProjectDirectory**=the folder of project was last opened. Used to locate the user in the same folder the next time he displays a load or save project dialog.
- **javaPath**=path of *java.exe.*
- **projectConfiguration=**path to the *.acideProject* file used to configure opened project (explained in *Chapter 13.4)*.
- **consolePanel.foregroundColor**=foreground color for console panel (numeric valor).
- **consolePanel.parameters**=parameters that *console panel* needs.

## 13.3. WORKBENCH CONFIGURATION

The workbench is all the space where user works with files. It contains the *Menu Bar*, the *Tool Bar*, the *Explorer Panel*, the *File Editor*, the *Console Panel* and the *Database Panel*.

The *XML* file that configures the workbench must be saved in the path *./configuration/workbench.*

The root label of this file is:

```
<acide.configuration.workbench.AcideWorkbenchConfiguration>
```

to reference the Java class *AcideWorkBenchConfiguration*.

Inside this root label there are six basic labels:

- **<_workbenchLoaded>:** with true value identifies if the configuration *XML* file has been loaded.
- **<fileEditorConfiguration>:** inside this label there are others nested labels with the configuration of the file editor (explained in *Chapter 13.3.3)*.

- **<_consolePanelConfiguration>:** inside this label there are others nested labels with the configuration of the console panel (explained in *Chapter 13.3.4*.
- **<_lexiconAssignerConfiguration>:** inside this label there are others nested labels with the configuration of lexicons for different extensions and lexicon applied to console (*lexiconAssignerConfiguration* explained in *Chapter 13.3.5)*.
- **<_recentFilesConfiguration>:** inside this label there is a list (inside a *_list* label) of *Strings* with the paths of files opened recently.
- **<_recentProjectsConfiguration>:** inside this label there is a list (inside a *_list* label) of *Strings* with the paths of projects opened recently.

### 13.3.1. MENU CONFIGURATION

The *Menu Bar* is the element situated at the top of *Workbench*. It contains as default the submenus *File, Edit, Project, View, Configuration* and *Help.* The *Menu Bar* provides user to do the most of actions that are provided in *ACIDE – A Configurable IDE.*

The root label of this file is:

```
<acide.configuration.menu.AcideMenuItemsConfiguration>
```

to reference the Java class *AcideMenuItemsConfiguration.*

Inside this label there is only one basic label, *_itemsManager*. This label has two nested *_list* labels. Inside of the most nested there are the *acide.configuration.menu.AcideMenuSubmenuConfiguration* objects that define the basic menus that exit on *Menu Bar.* They have the following nested labels:

- **<_name>:** the name of the submenu.
- **<_visible>:** with true or false value. It sets if submenu is visible or not.
- **<_erasable>:** with true or false value. It sets if submenu is erasable or not erasable (it is a default submenu).
- **<_image>:** for submenus this label is empty.
- **<_itemsManager>:** it is equal to root *_itemsManager* label. It contains all the menu objects that are inside the submenu.

For the menu items the label is *AcideMenuItemConfiguration.* They have the following nested label:

- • **<_name>:** the name of the item.
- **<_visible>:** with true or false value. It sets if item is visible or not.
- **<_erasable>:** with true or false value. It sets if item is erasable or not erasable (it is a default item).
- **<_image>:** the path of its image icon.
- **<_command>:** the command will be run when user click on this menu item.
- **<_parameter>:** the type of parameter that command needs to run. It can be *NONE, TEXT, FILE,* or *DIRECTORY.*

User can insert, delete, reorder, etc. *AcideMenuObjectConfiguration* labels (*AcideMenuSubmenuConfiguration* and *AcideMenuItemConfiguration* both are subclasses of *AcideMenuObjectConfiguration)* inside the root label to manage the configuration of the *Menu Bar.*

### 13.3.2. TOOLBAR CONFIGURATION

The *Tool Bar* is situated below the *Menu Bar*. In the *Tool Bar* appear several buttons for typical actions with files and projects. It also contains buttons that user configures to send commands to shell and to launch external applications.

Toolbar configuration is done in *.toolbarConfig* files. These files are divided in two parts, one part that stores settings of buttons for the toolbar that paste code on the console to be run, and other part for configuration of buttons to launch external applications.

To configure buttons to send commands to the console, each configuration of a button should be headed by a comment line (starting with //) and consists of six lines with the following structure:

- **name** = name displayed.
- **action** = command to run on the console.
- **hintText** = help text displayed when user puts mouse over the button.
- **icon** = path of the image for the button.

- **parameterType** = type of the parameter that the command uses on console. It can be:
    - **NONE**
    - **TEXT**
    - **FILE**
    - **DIRECTORY**
- **isExecutedInSystemShell** = if is executed in the system or not.

Once the list of command buttons is ended, user has to enter the following line in the file, in order to indicate that the following settings are for buttons that launch external applications:

```
//End of Console Panel Tool Bar Button Configuration
```

Configurations of buttons that launch applications must be headed by a comment line (starting with //) followed by four lines of properties:

- **name** = name displayed.
- **path** = path to run.
- **hintText** = help text displayed when user puts the mouse over the button.
- **icon** = path of the image for the button.

Once the list of command buttons is ended, user has to enter the following line in the file, in order to indicate that configuration of buttons that launch external applications is ended:

```
//End of External Applications Tool Bar Button Configuration
```

### 13.3.3. FILE EDITOR CONFIGURATION

The *File Editor* is where user can edit the content of the files. It contains a tab pane where the opened files are displayed.

The *File Editor* is configured by a label in the *XML* file that configures the *Workbench* (explained on *Chapter 13.3)*. Inside this label the user can find the information needed to configure *File Editor*. The labels are:

- **_fileEditorConfigurationList**: acts like a *Manager* (explained on *Chapter 13.1)* including two nested *_list* labels with *AcideFileEditorPanelConfiguration* objects. These objects store information about files which must be shown opened at *File Editor* next time the application will be opened.
- **_selectedFileEditorPanelName**: the name of the file which is shown at the *File Editor*.
- **_fontName**: the font name of the text of *File Editor*.
- **_fontStyle:** font style of the text of *File Editor.*
- **_fontSize:** font size of the text of *File Editor*.
- **_foregroundColor, backgroundColor:** RGB components of font color and background color.
- **_editionMode:** with false value, edition mode is *INSERT*, with true value it is *OVERWRITE.*
- **_automaticIndent:** with true value, automatic indent, with false value, manual indent.
- **_maximumLinesToConsole:** the maximum number of lines that can be sent to the console at the same time.
- **_lineWrapping:** with true value, sets on line wrapping, with false value, sets off line wrapping.
- **_sendToConsoleConfirmation:** with true value, system needs confirmation to send content of file to console. With false value, file content is sent without confirmation.

### 13.3.4. CONSOLE PANEL CONFIGURATION

At *Console Panel* content of shell connected with the application is displayed.

It has two labels:

- **_lexiconConfiguration:** path of lexicon which is used at console.
- **_commandsConfiguration:** path of *XML* file that contains commands history with which we want to start the console (explained in *Chapter 13.6).*

Sdvsvs

### 13.3.5. LEXICONASSIGNER CONFIGURATION

It has three basic labels:

- **_list:** acts like a *Manager*, inside there is a *_list* label with another *_list* label nested. It is a list of *AcideLexiconAssigner* objects. These objects describe possible lexicons to use at console. They have the following nested labels:
  - **_description:** name of lexicon.
  - **_extensionList:** it has a group of nested String labels with the possible extensions for the lexicons.
  - **_lexiconConfiguration:** path of *XML* file that configures lexicon.
- **_consoleLexiconConfiguration:** path of *XML* file that configures lexicon which is currently in use.
- **_applyLexiconToConsole:** with true value lexicon is applied to console, with false value it is not applied.

## 13.4. PROJECT CONFIGURATION

Project configuration is edited in *.acideProject* files. In this type of files are arranged in separate lines different project properties. These are, line by line, the following:

```
1. Project Name
2. Project Path
3. Compiler Path
4. Compiler Arguments
5. Compiler All Files
6. File separator
```

```
7. File extensión
8. Executable path
9. Executable arguments
10. Console panel Shell path
11. Console panel Shell directory
12. Console panel exit command
13. Console panel is echo command
14. Console panel parameters
15. Console panel foreground color
16. Console panel background color
17. Console panel Font name
18. Console panel Font style
19. Console panel Font size
20. Console panel buffer size
21. Is explorer panel showed flag
22. Is console panel showed flag
23. ACIDE - A Configurable IDE main window width
24. ACIDE - A Configurable IDE main window height
25. ACIDE - A Configurable IDE main window x coordinate
26. ACIDE - A Configurable IDE main window y coordinate
27. ACIDE - A Configurable IDE main window vertical split
28. ACIDE - A Configurable IDE main window horizontal split
29. Language configuration
30. Database panel configuration
31. Menu configuration
32. Menu new configuration
33. Tool bar configuration
34. Number of files of the project
```

The following lines show the properties of the project files. For each file there are seven lines of text storing the file properties. Therefore, there will be many groups of seven lines in the configuration file as indicated at line number 34. The properties are as follows:

- Absolute path.
- Name.
- Parent.
- Directory flag.
- Compilable flag.
- Main flag.
- Opened flag.

## 13.5. CONFIGURATION OF LEXICONS

Lexicons can be configured by manually editing *XML* files that define them.

A *XML* file that defines a lexicon begins with the root label:

```
<acide.configuration.lexicon.AcideLexiconConfiguration>
```

to reference the class *AcideLexiconConfiguration.* Inside this root label there are seven basic tags:

- **_name:** defines the name of the lexicon.
- **_path:** indicates the relative path of this file.
- **_isCompiledOrInterpreted:** a false value indicates that the lexicon is compiled and true indicates that it is interpreted.
- **_tokenTypeManager:** it is a *Manager* (explained on *Chapter 13.1)* of the types of token there are in the lexicon. It consists of a list of objects *AcideLexiconTokenGroup.*
- **_validExtensionsManager:** it is a *Manager* of valid extensions of files at the lexicon defined in the *XML* document. The extensions are *String* objects.
- **_delimitersManager:** it is a *Manager* of valid delimiters at the lexicon defined in the *XML* document. The delimiters are *String* objects.
- **_remarksManager:** it is not a common *Manager.* It defines the symbol to mark a line as a comment in the lexicon. It has four nested labels:
  - **_symbol:** defines the symbol to use to begin a comment line.
  - **_isCaseSensitive:** defines (true or false value) if it is case sensitive.
  - **_color:** defines color of the comments. It has four nested tags (*red, blue, green, alpha*) that define the RGB components and the degree of opacity of the comments.
  - **_fontStyle:** defines the font style of comments.

### 13.5.1. TOKENTYPE MANAGER

This label has two nested *_list* labels. Inside of the most nested there are the *AcideLexiconTokenGroup* objects that define the token types in the lexicon. The *AcideLexiconTokenGroup* objects have five nested labels:

- **_name:** it is a summary of the properties defined by the remaining labels. It has the following form:
  - Color: [R: _, G: _, B: _], Font Style: __, Case Sensitive: _

- For color will take the values defines in the label *_color*. In Font Style appears the name that corresponds to the number defined on the label *_fontStyle.* In Case Sensitive value yes appears if the label - *_IsCaseSensitive* is true and value not if the label *_IsCaseSensitive* has value false.

- **_color:** same structure as explained for *_color* label above.
- **_fontStyle:** it defines with a number the font style.
- **_isCaseSensitive:** it defines by true or false value if it is case sensitive.
- **_tokenList:** contains a label called *_list* where appears the list of *String* objects which define the tokens with the properties user has described for this token group. Adding, removing and editing these strings the user will get the list of tokens.

### 13.5.2. VALIDEXTENSION MANAGER

As a *Manager*, it has two nested *_list* labels. Inside the last the user can find String objects labels where he can define extensions valid for the lexicon.

### 13.5.3. DELIMITERS MANAGER

It is a *Manager* whose list contains String objects. With the strings the user defines the valid delimiters for the lexicon.

## 13.6. COMMANDS HISTORY

In *ACIDE – A Configurable IDE* is possible to configure a commands history so that when user starts the application already exits this history, similar to when he gets commands entered earlier in the same run.

The *XML* file that contains the commands history must be saved in the path *./configuration/console.*

The root label of this file is:

```
<acide.configuration.console.AcideConsoleCommandsConfiguration>
```

to reference the AcideConsoleCommandsConfiguration class.

Inside this label user has to define another label of *Manager* type called *_commandsManager.*

As usual at *Managers*, there are two nested *_list*. Inside the last the user defines by String labels the commands he wants to introduce in the commands history. The first command at the list acts like the less recently entered at the console.

# 14. REGULAR EXPRESSIONS

A regular expression, often called pattern, is an expression that describes a set of strings without listing their elements. Most formalizations provide the following constructors: a regular expression is a way of representing regular languages (finite or infinite) and is constructed using alphabet characters on which the language is defined. Regular expressions provide a flexible way to search or recognize strings.

## 14.1. CONSTRUCTION OF REGULAR EXPRESSIONS

Specifically, regular expressions are built using the operators union, concatenation and Kleene closure.

- **Alternation**: A vertical bar separates alternatives. For example, "red|brown"joins with red or brown.
- **Quantification**: A quantifier after a character specifies the frequency with which this can occur. The most common quantifiers +, ? and *:
    - **+**: The plus sign indicates that the preceding character must appear at least once. For example, hello+ joins hello, helloo, hellooo, etc.
    - ?: The question mark indicates that the preceding character can appear at most once. For example, S?pain joins Spain and pain.
    - **: The asterisk indicates that the preceding character can appear zero, one, or more times. For example 10** joins 1, 10, 100, 1000, etc.
- **Grouping:** Parentheses may be used to define the scope and precedence of other operators. For example, "(m|h)ouse" is the same as "mouse | house" and "(in)?sensitive" joins with insensitive and sensitive.

Builders can be freely combined within the same expression, so "H (ae? | ä) del" is the same as "H (a |ae | ä) del".

Its most obvious use is to describe a set of strings, which is useful in text editors and applications for searching and manipulating text.

## 14.2. DESCRIPTION OF REGULAR EXPRESSIONS

### 14.2.1. THE DOT "."

The dot is interpreted by the search engine as "any character", looking for any character NOT including line breaks.

The dot is used as follows: If we search "g.t" in the string "gat get got goot" the search engine will find "gat" "get" "got". Note that the search engine don't find "goot", this is because the dot represents a single character and only one.

### 14.2.2. THE BACKSLASH "\"

It is used to "tag" the next character in the search expression so that it acquires a special meaning or stop having him. The backslash is never used by itself, but in combination with other characters. Used for example in combination with the point "\.", this has not its normal meaning and behaves as a literal character.

In the same way, placing a backslash followed by any of the special characters discussed below, these do not have special meaning and become literal search characters.

As mentioned previously, the backslash can also give special meaning to characters that do not. Below is a list of some of these combinations:

- **\t:** represents a tab.
- **\r:** represents the *carriage return* or *return to top*, the place where the line starts again.
- **\n:** represents the *new line* character through which a line begins. Remember that in *Windows* is needed a combination of \r\n to start a new line, while *Unix* uses only \n and classic *Mac OS* uses only \r.
- **\a:** represents a *bell* or *beep* that occurs when you print this character.
- **\e:** represents the *Esc* or *Escape.*
- **\f:** represents a page break.
- **\v:** represents a vertical tab.
- **\x:** is used to represent *ASCII* or *ANSII* code.
- **\u:** is used to represent *UNICODE* characters with its code.
- **\d:** represents a digit from 0 to 9.

- **\w:** represents any alphanumeric character.
- **\s:** represents a blank space.
- **\D:** any character other than a digit from 0 to 9.
- **\W:** represents any non-alphanumeric character.
- **\S:** any character other than a blank.
- **\A:** represents the beginning of the string. Not a character but a position.
- **\Z:** represents the end of the string. Not a character but a position.
- **\b:** marks the beginning and end of a word.
- **\B:** marks position between two alphanumeric or non-alphanumeric characters.

### 14.2.3. THE BRACKETS "[]"

The function of the brackets in regular expressions is to represent "character class", grouping characters into groups or classes. They are useful when is needed to find one of a group of characters. Within the brackets you can use the "-" to specify ranges of characters. Additionally, the metacharacters lose their meaning and become literal when they are inside the brackets. For example, as mentioned previously, "\d" is useful to find any character that represents a digit. However, this name does not include the "." dividing the decimal part of a number. To search for any character that represents a digit or a point we can use the regular expression "[\d.]". As noted above, within the brackets, the point represents a literal character, not a metacharacter, so it is not necessary to precede the backslash. The only character that must be preceded by the backslash inside the brackets is the backslash.

### 14.2.4. THE BAR "|"

Used to indicate one of several options. For example, the regular expression "a|e" find all "a" or "e" in the text. The regular expression "East|West|North|South" will find any of the names of the cardinal points. The bar is commonly used in conjunction with other special characters.

### 14.2.5. THE DOLLAR SIGN "$"

Represents the end of the string or the end of the line when using the multi-line mode. There is not a special character, but a position. Using the regular expression

"\.$" the engine will find all the places where a line ends with a dot, which is useful for moving between paragraphs.

### 14.2.6. THE CARET "^"

This character has a dual function, which differs when used alone and when used in conjunction with other special characters. Firstly its functionality as an individual character: the character "^" represents the beginning of the chain (in the same way that the dollar sign "$" represents the end of the string). Therefore, using the regular expression "^[a-z]" the engine will find all paragraphs beginning with a lowercase letter. When used in conjunction with the brackets, for example with the form "[^\w]", is useful to find any character that is not in the indicated group. The above expression can found any character that is not alphanumeric or a space, all punctuation and other special characters.

### 14.2.7. PARENTHESES "()"

Similarly to the brackets, parentheses are used to group characters. However, there are several differences between groups established by brackets and groups established by parentheses:

- Special characters keep their meaning within the parentheses.
- Groups established by parentheses make a *label* for the search engine that can be used later as denoted below.
- Used in conjunction with bar "|" enables optional searches. For example, the regular expression "to (East | West | North | South) of" searches texts giving instructions through cardinal points, while the regular expression "East | West | North| South" find "east" in the word "beast", failing to fulfill this purpose.
- Used in conjunction with other special characters listed below provide additional functionality.

### 14.2.8. THE QUESTION MARK "?"

The question mark has several features in regular expressions. The first is to specify which part of the search is optional. For example, the regular expression "S?pain" can find both "pain" and "Spain". In conjunction with parentheses specifies

that a larger set of characters is optional, for example, “Nov(\.|ember|iembre)?” finds both “Nov.”, “November” and “Noviembre”. Similarly, you can use the question mark with another meaning. Parentheses define groups “anonymous”, but the question mark in conjunction with triangular brackets “<>” give name to such groups as follows: “^(?<Day>\d\d)/(?<Month>\d\d)/(?<Year>\d\d\d\d)$” Whereupon it specifies to the search engine that the first two digits found will be labeled “Day”, the second will be labeled “Month” and the last four digits will be labeled “Year”.

### 14.2.9. THE BRACES “{}”

Usually the braces are literal characters which are used separately in a regular expression. To be used as metacharacters they have to enclose one or more numbers separated by commas and to be placed to the right of another regular expression as follows: “\d{2}”. This expression will find two adjacent digits. Using this formula, the example “^\d\d/\d\d/\d\d\d\d$” that served to validate a date format will become to “^\d{2}/\d{2}/\d{4}$” for clarity in reading the expression.

### 14.2.10. THE ASTERISK “*”

The asterisk is used to find something that is repeated 0 or more times. For example, using the expression “[a-zA-Z]\d*” will be possible to find both “H” and “H1”, “H01”, “H100” and “H1000”, a letter followed by a indefinite number of digits.

### 14.2.11. THE PLUS SIGN “+”

It is used to find a string that is repeated one or more times. The expression “[a-zA-Z]\d+” will find “H1” but will not find “H”.