Basic Manual

Omega IDE

In this manual, you will learn basic things about how to use Omega IDE for java development.

You will learn how to create projects and files, how the project structure is created, how to compile and run the project and at last the keybindings.

First of all, remember that Omega IDE strictly follows traditional java conventions that are used from creating files to accessing content assist properly.

What are Java Conventions?

Well, it follows the identifier naming system but with some conditions that simplifies the look of a java program.

Only four concepts are required to be learned.

Lets start!

In Java, the name of a class should begin with an upper case character.
Example: String, System or Graphics.

In Java, the name of a final variable should only contain Upper Case Character (any digit or underscore(_) is allowed if required).
Example:

```
final int VAL2 = 0;
final String HIS_NAME = “myName”;
```

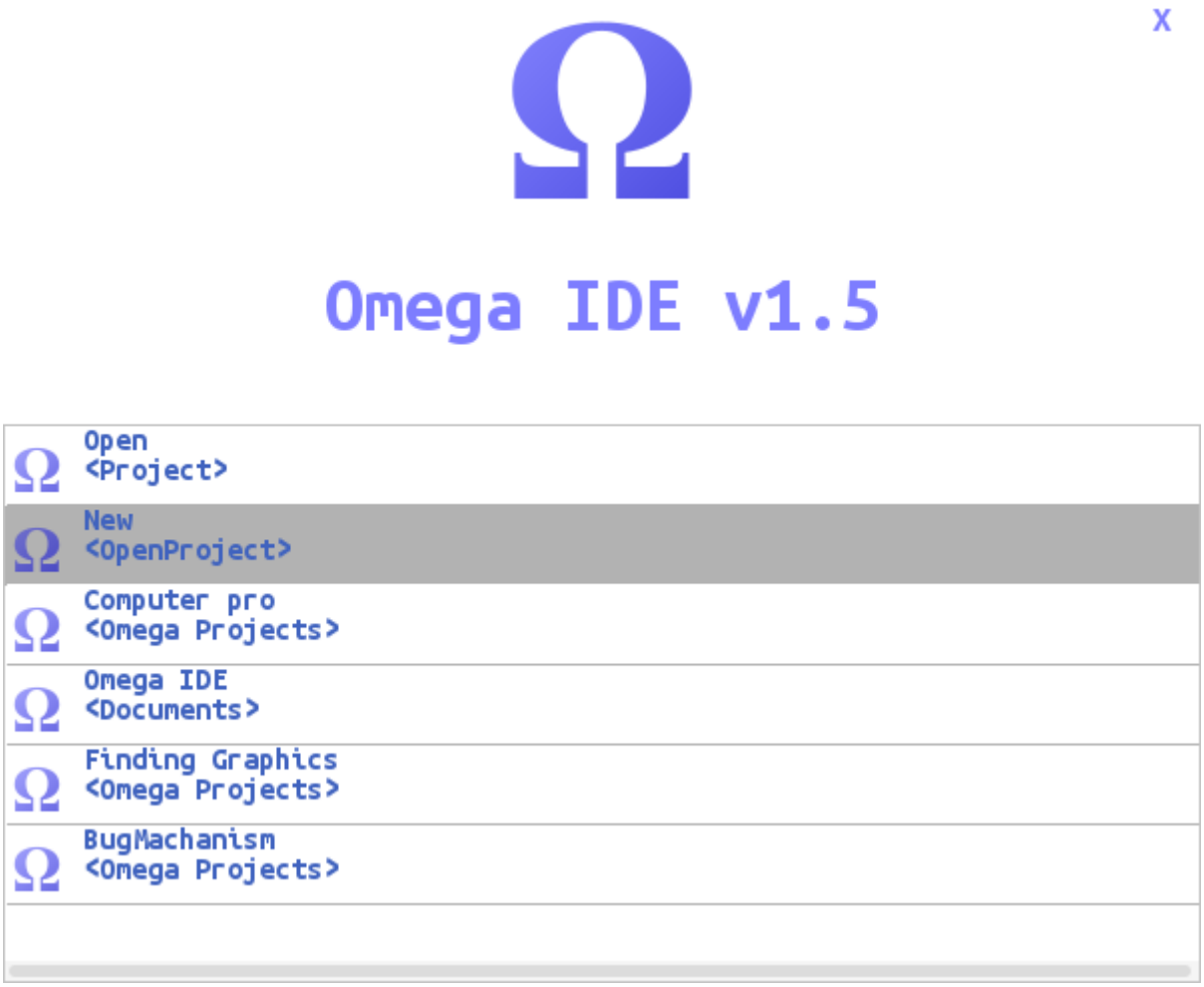
In Java, the name of a package should only be comprised of Lower Case characters.

Example: `java.util` or `say.swing`

In Java, the name of a non-final variable or method must start with a Lower Case character.

Example: `readData()` or `int num;`

Lets Move on to how to use Omega IDE!



Lets start with creating a new project click on the ‘New’ Button

This will popup this window up.

Now, specify the name of your project let say MyFirstProject.

Choose the JDK environment by clicking on the ‘:’ button. If you have multiple JDKs installed on your system else leave it as it is but remember the auto imports and content assist will not work.

Now you will see a text area with some instructions written on it.

Let me tell you If you still cannot make out what it says. Omega IDE eases you when creating new projects with this let see how.

That text area is simply demanding only some source

files names that you want to create.

This auto generates the sources when the project is created.

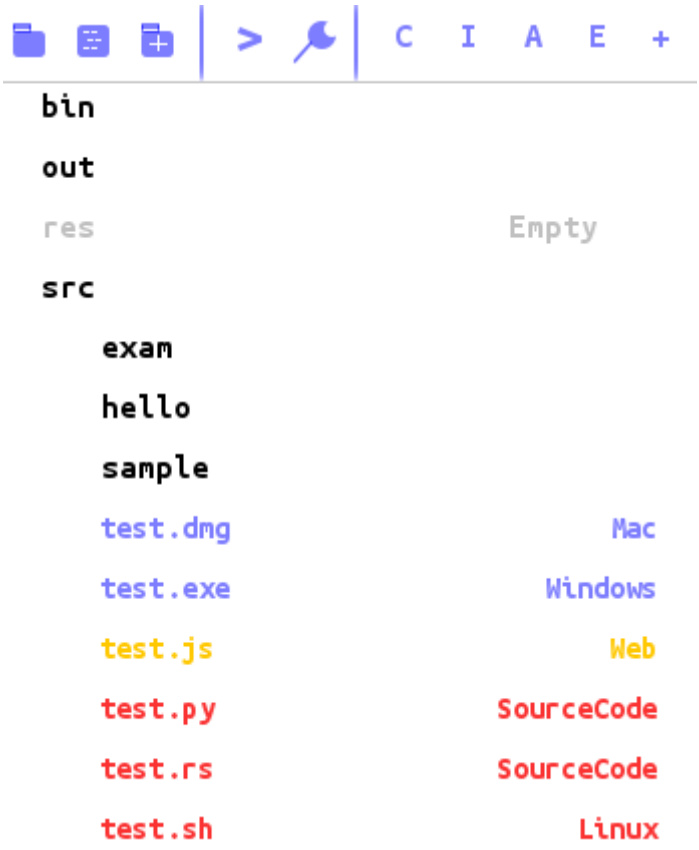
Want to try it out?
Type in the text area

```
pack.Main -class
pack.Intx -interface
pack.Ann -@interface
pack.Enu -enum
```

This will first generate a package called pack and then,

four source files (One Class, One Interface, One Annotation and One Enum) as specified.

Now from below this text area you will see another non-editable text area, that is your dependency area if you know what are dependencies then you will simply make out what this is.



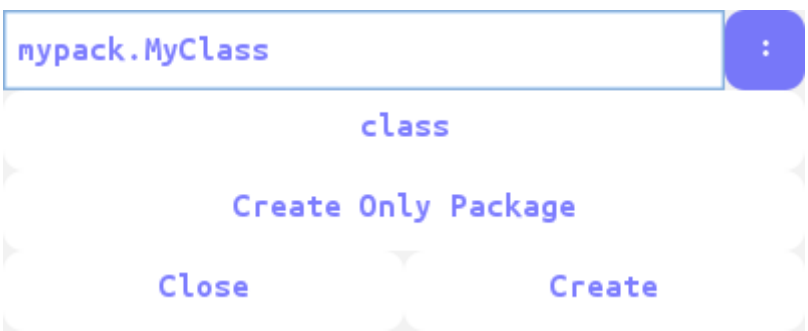
Remember, all the compiled sources are placed in the bin directory of the project, all the output of the IDE (the editor images, etc) are stored in the out directory, the res folder is a default resource root of the project but if you want to use it then you need

to add it to the class path from the IDE and all the sources files are generated in the src directory of the project.

Lets start with creating a class.

In Omega IDE the use of default package is not allowed, that is, a class without a package cannot be created in Omega IDE.

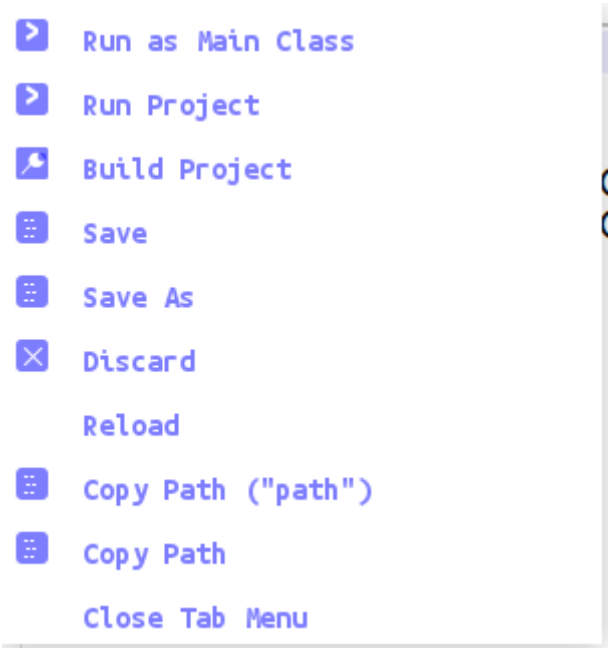
So, lets start with creating a class, click on the button ‘C’ in the ToolBar and type mypack.MyClass and click create.



This will auto generate the package mypack and will create a raw class named MyClass in that package and will open it in the editor for editing.

The ‘:’ symbol on the right side of the text field define the root of the files that is the directory where the files are to be created, by default it is set to ‘src’ directory of current project, however you can change it anytime.

Running the Class as Main :
To run a class as open it in the editor and click on the ‘J’ button and then click ‘Run As Main Class’.



This will launch the specified class as Main-Class of the project.

You need not to do this every time to run the project. Just

click on the run button in the toolbar to do so.

Compiling the project :

Click on the build button located next to the run button to compile the project.

Key Bindings

Control + R → Run Project
Control + B → Build Project
Control + S → Save Current Editor
Control + Shift + O → Auto Import
Control + Shift + F → Find&Replace
Control + Shift + R → Run as Main
Control + Shift + C → Click Editor Image
Control + Shift + I → Override/Implement Methods from parent
Control + Shift + G → Generate Getters/Setters