# F:\文件\图片\RZJ高亮光.png

# Relative-RZJ使用教程

………………………………………..………………目录……………………….………………………………..

[1前期准备>](#_<前期准备>)

[2新建工程>](#_<新建工程>)

[3打开工程>](#_<打开工程>)

[4工程内部操作>](#_<工程内部操作>)

[4.1新建工程代码>](#_{新建工程})

[4.2打开工程代码>](#_{打开工程})

[5其它>](#_<序>)

[6Relative目录存储详解>](#_<Relative目录存储详解>)

[7Relative进程通信操作>](#_<Relative进程通信操作>)

[8语言库的Dll接口和事件>](#_<语言库的Dll接口和事件>)

[9卸载>](#_<卸载>)

………………………………………..………………目录……………………….………………………………..

## <前期准备>

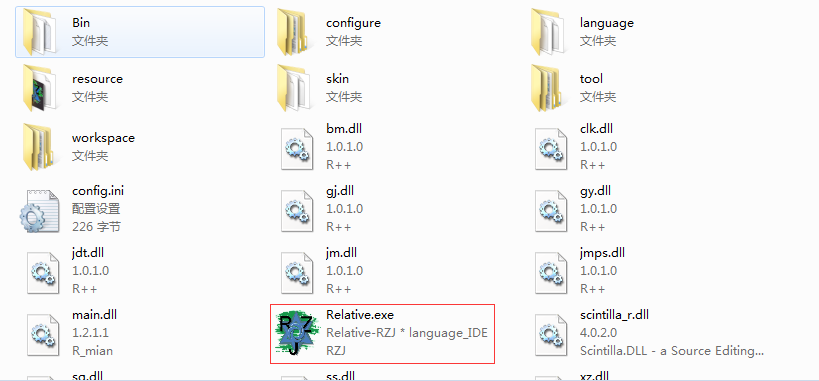
解压下载的程序包



>右键解压到当前目录



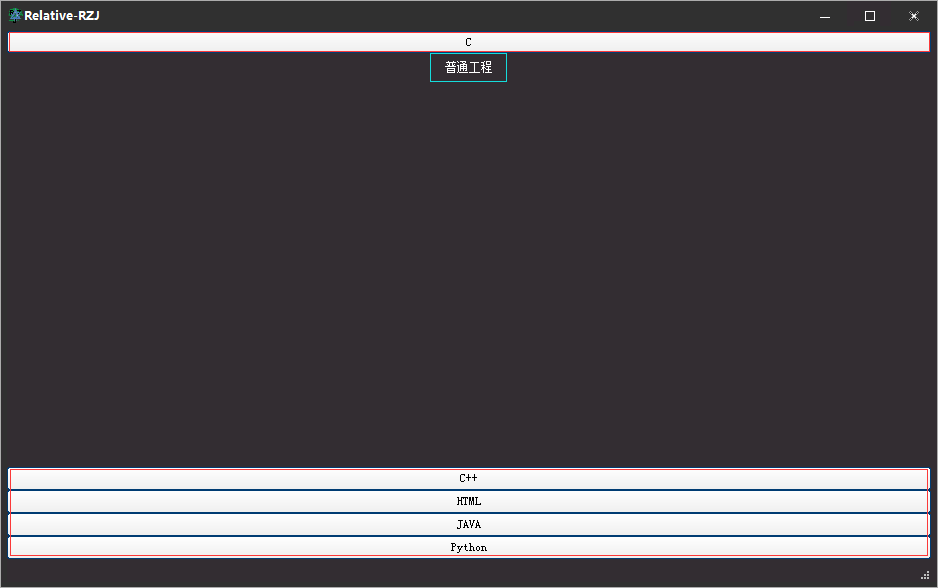
>双击打开目录



>双击打开Relative.exe可执行文件

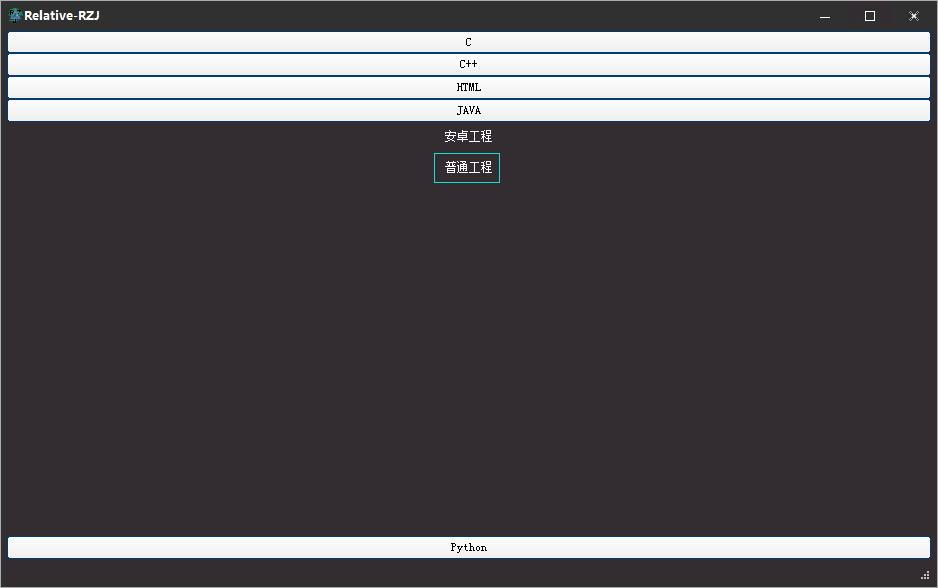


>等待程序加载……



》红标标记的是计算机编程语言

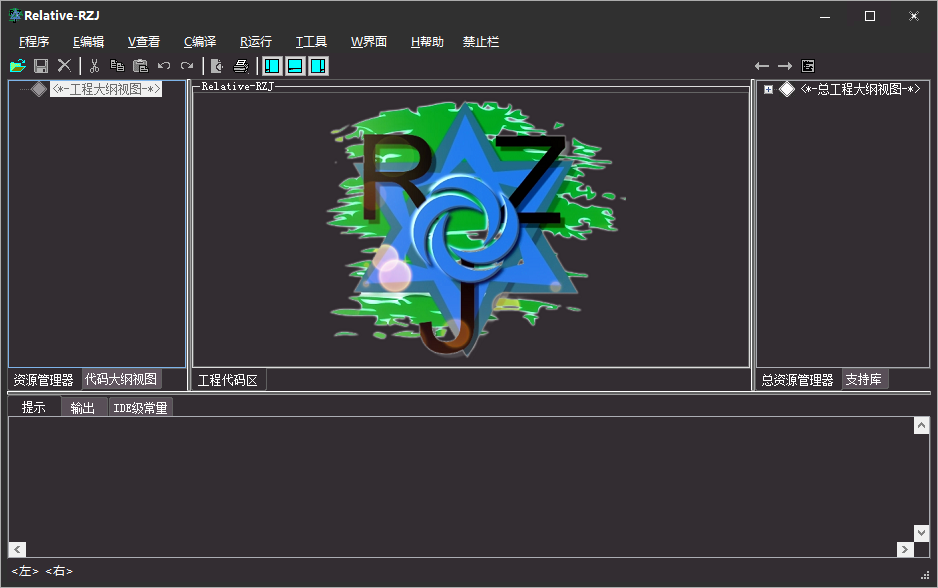
》蓝标标记的是当前语言的工程类型



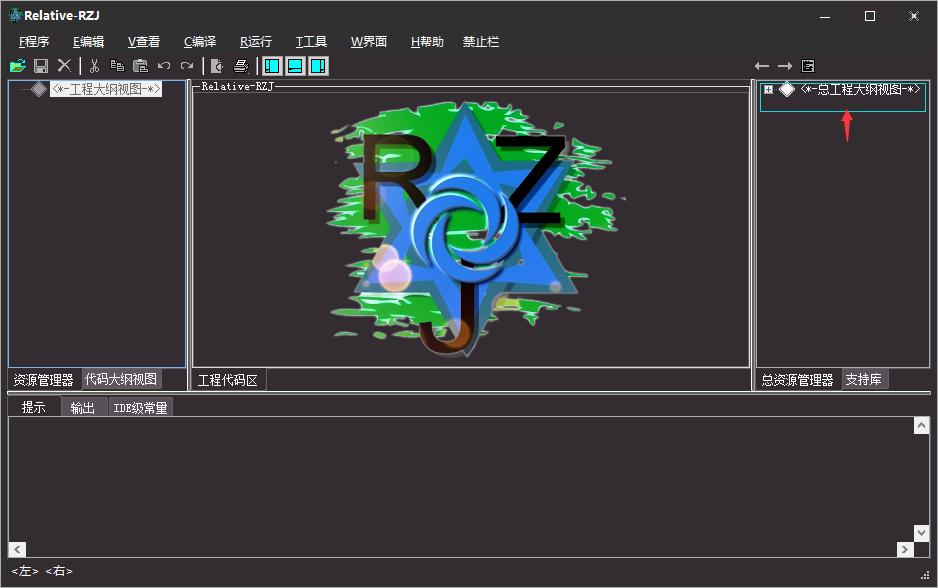
>这里我选择JAVA的普通工程(鼠标单击)

>第一次选择语言库会加载语言库的库数据，相对会慢些。

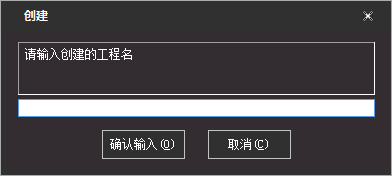
>进入主界面



## <新建工程>

1.右键右边的总资源管理区

在弹出的菜单中选择



在弹出的目录中输入要创建的工程目录名(这里我取名Relative)



展开发现多了（Relative）这个工程就证明工程创建完成！

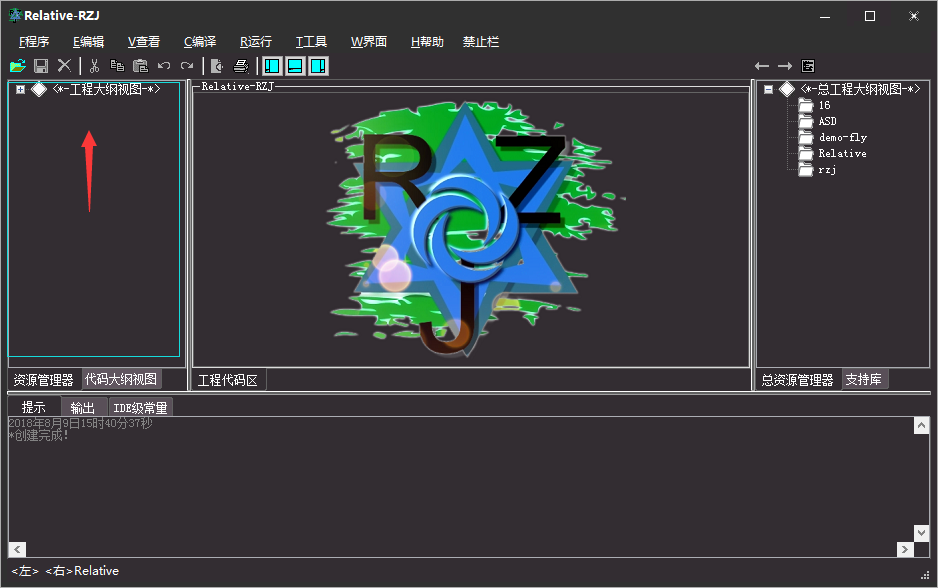
## <打开工程>

1.双击要打开的工程



2.右键在菜单中选择

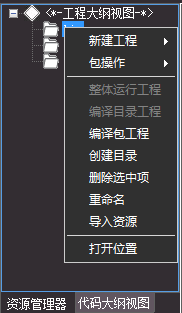
# <工程内部操作>



打开工程后会发现左边的子资源管理器可以操作。

展开后会有默认的操作目录

右键会有操作菜单弹出



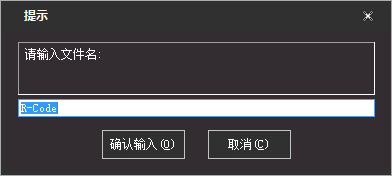
<右键菜单功能是否可用取决于Relative、当前语言库的操作和右键文件类型的影响>

### {新建工程}

值得注意的是Relative的主程序（Relative.exe本身不带有任何语言的操作能力。所以新建代码的位置取决于选择的语言库的影响。这里我们受之前选择的JAVA>普通工程语言库的影响）



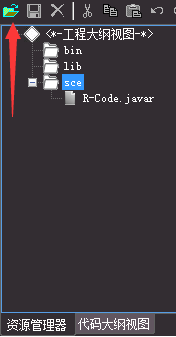
在选项中我们选择新建



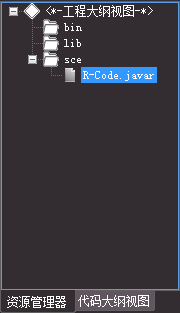
在弹出的目录中设置工程名，默认（R-Code.javar）

值得注意的是（在处文件名不需要打后缀。且生成的文件默认会与选择的语言默认工程后缀有差别。比如JAVA工程默认后缀为\*.java而Relative会主动判断\*.javar为默认文件。这是为了将中文JAVA代码和英文JAVA代码区分开而采用的机制。）

### {打开工程}

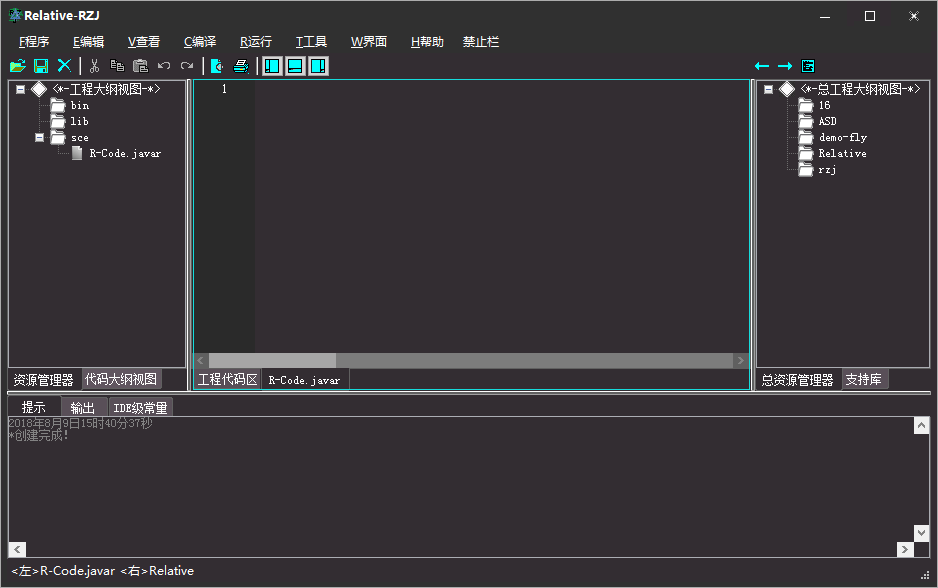
1. 

先选中要打开的工程文件如何点击该按钮

2. 

双击要打开的工程

打开后就可以进行代码的编写了。

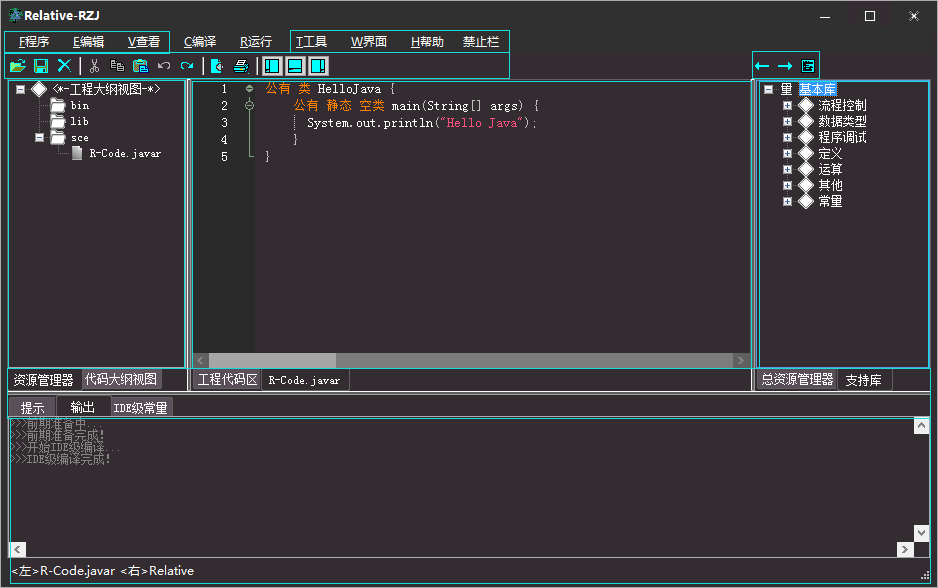


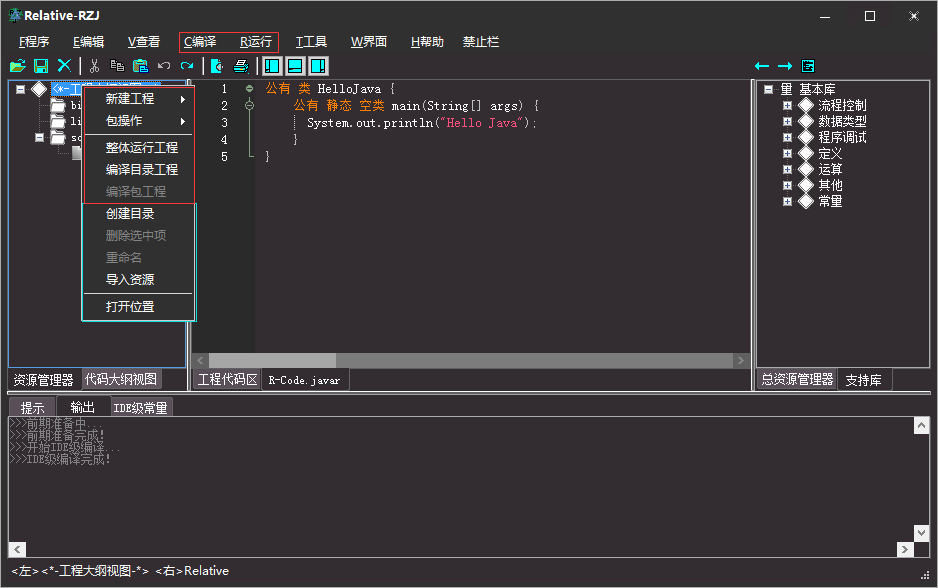
目前中文化的代码只有《基本库》中的内容

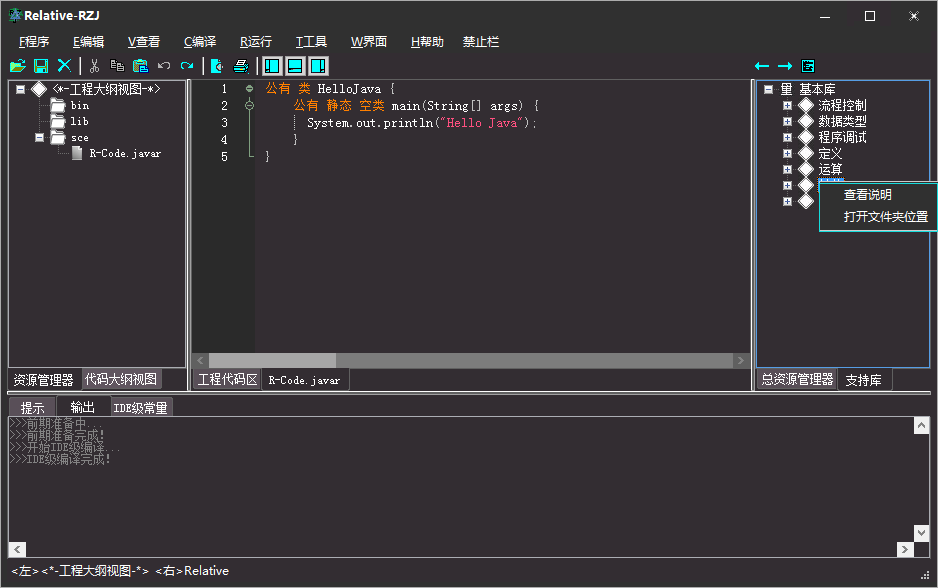


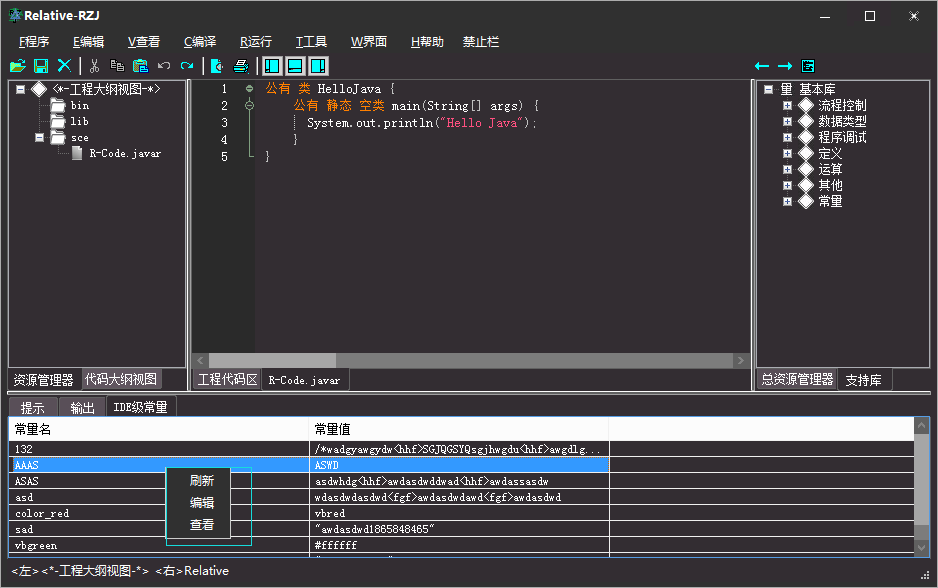
## <序>

1.列出所有Relative原生功能：(蓝色为原生无色或红色为库动态添加的功能)

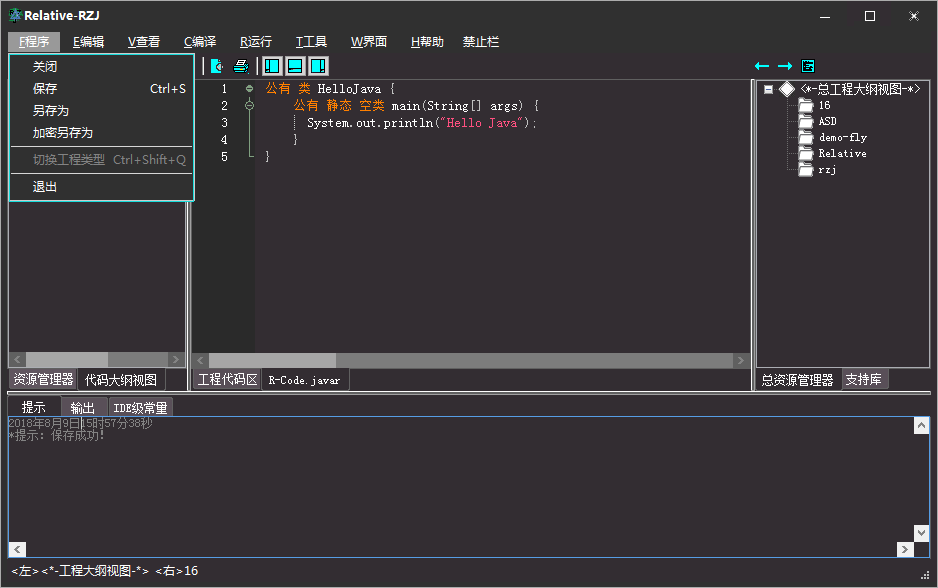


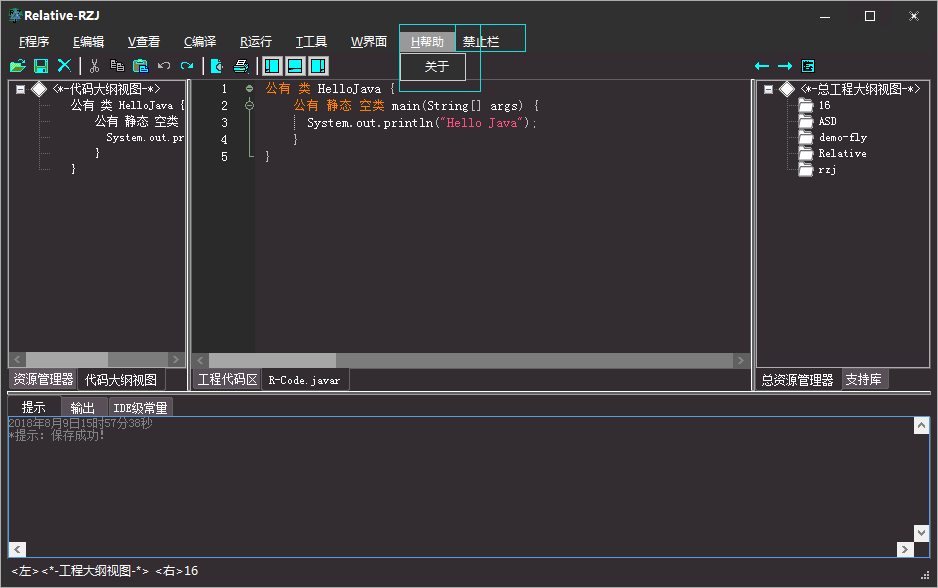
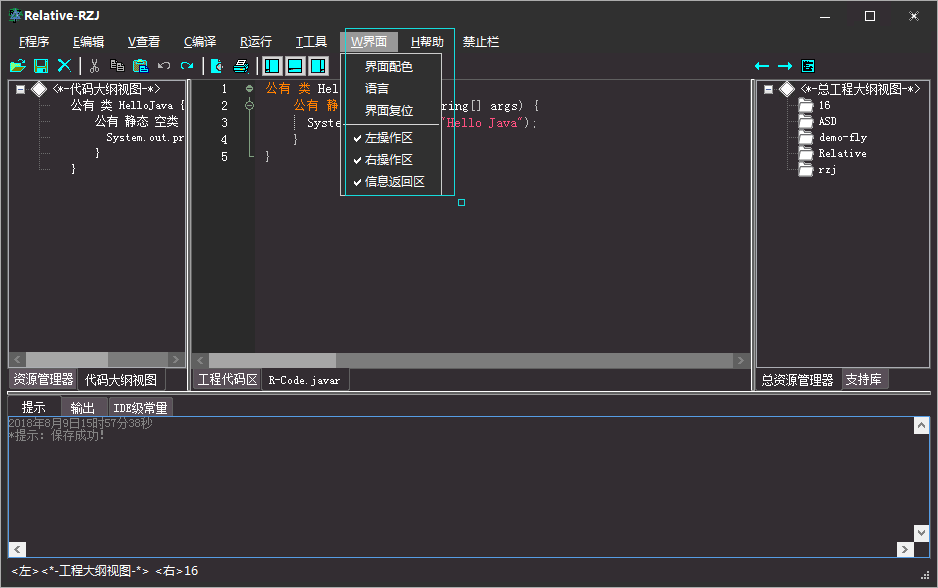
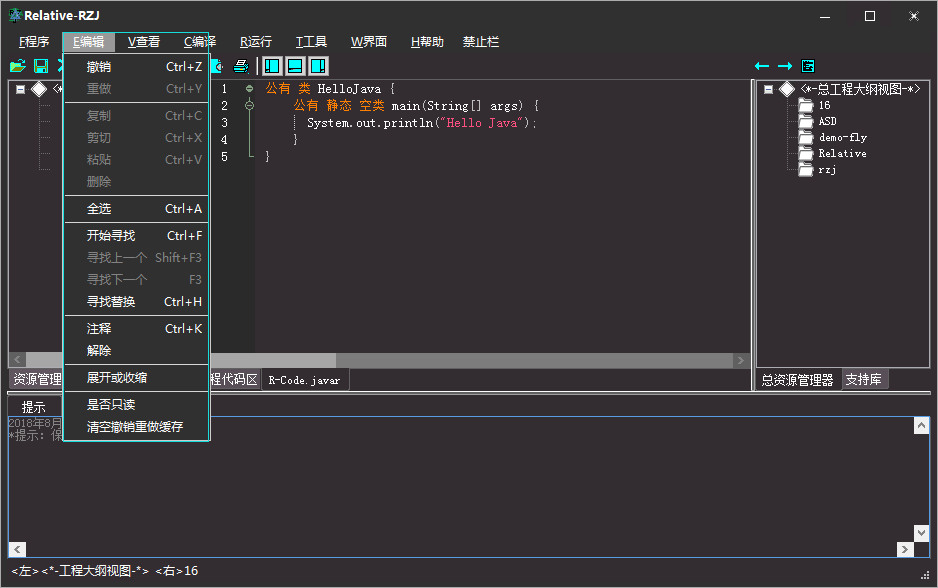
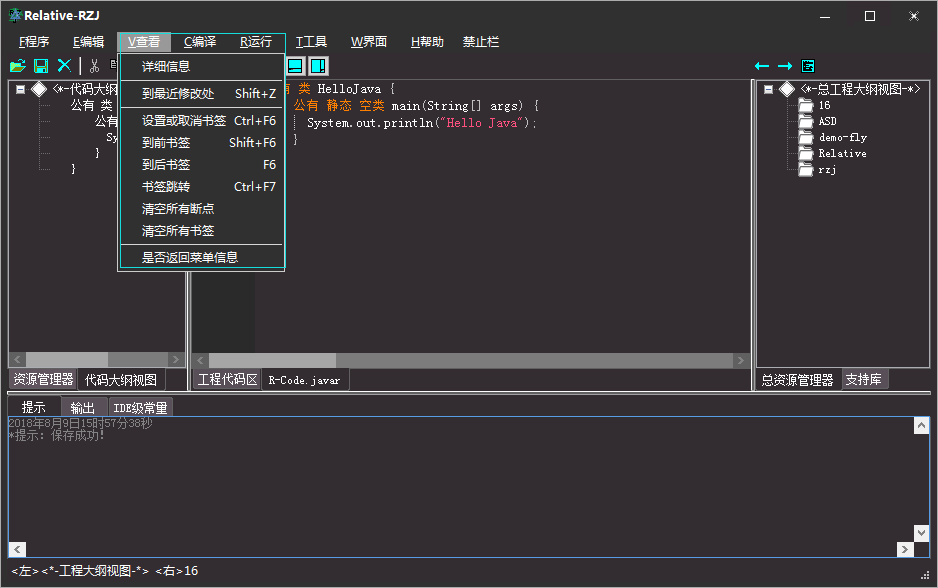




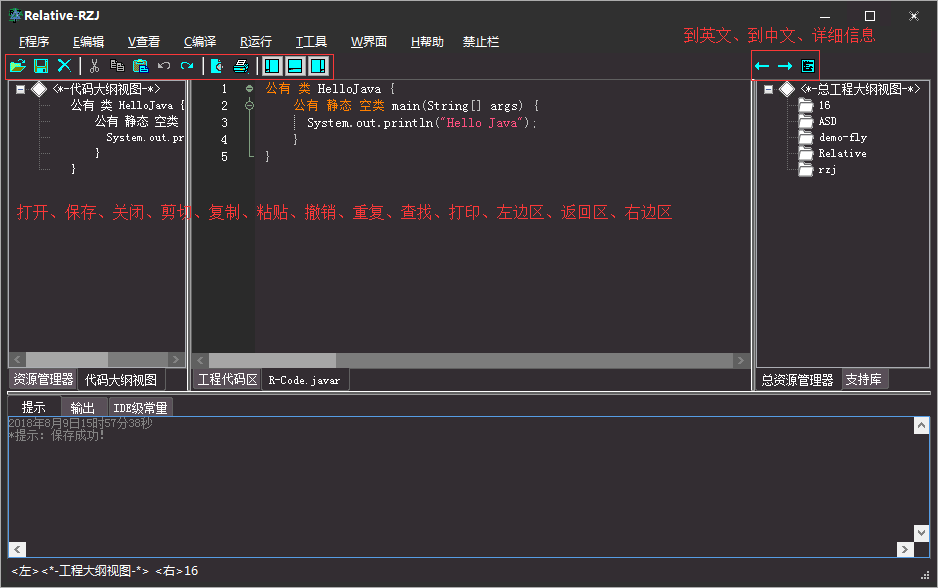




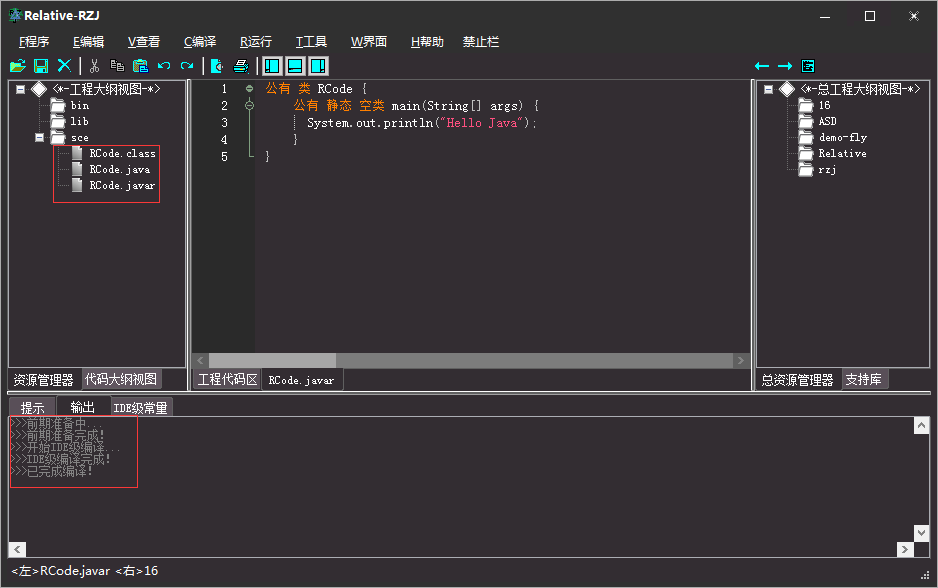




2.工具栏介绍：



3.编译<JAVA库.普通工程>



编译操作后会产生该语言的默认后缀源代码。内部为翻译后的源代码

public class RCode {

public static void main(String[] args) {

System.out.println("Hello Java");

}

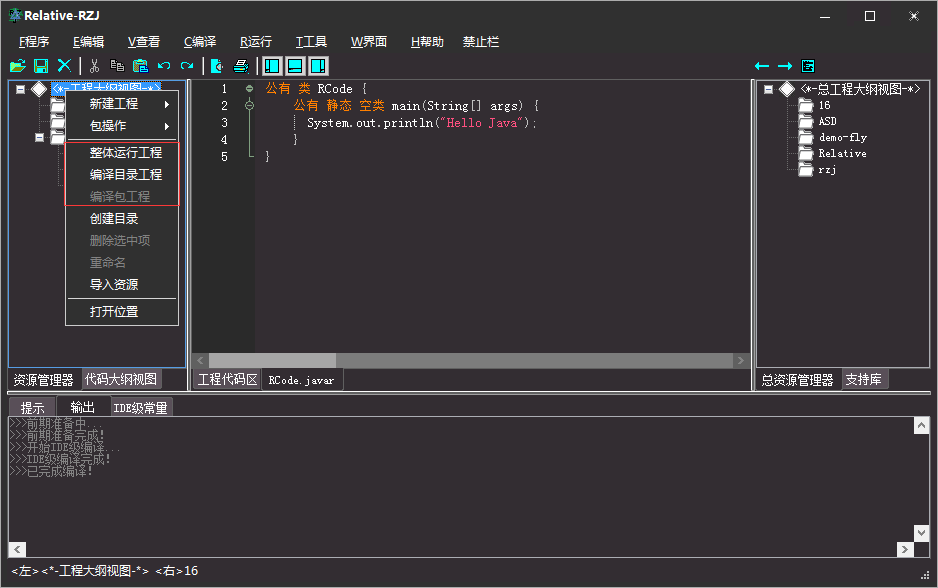
}

和

后缀为\*.class的文件

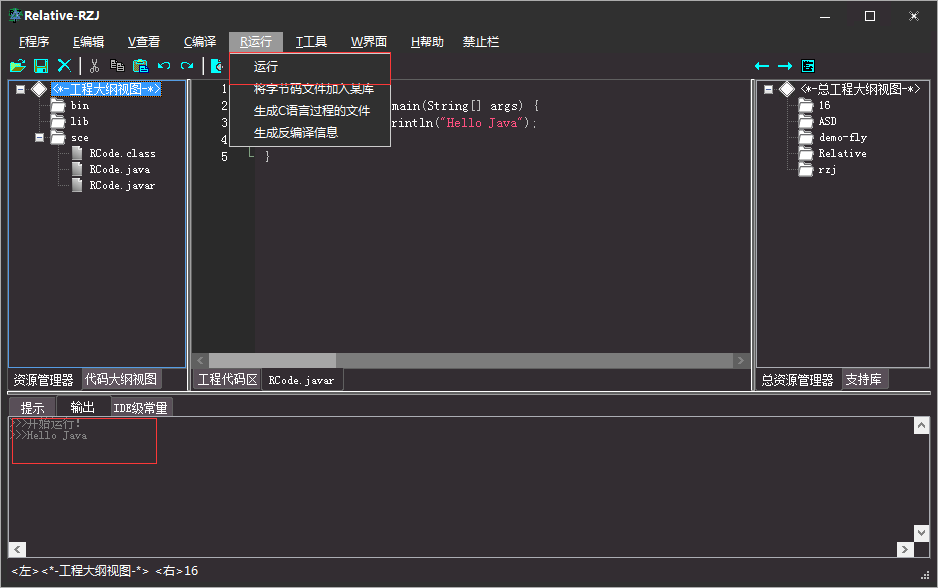
这代表编译完成

JAVA库中提供的编译方式不只有这种。

还有：

4.运行

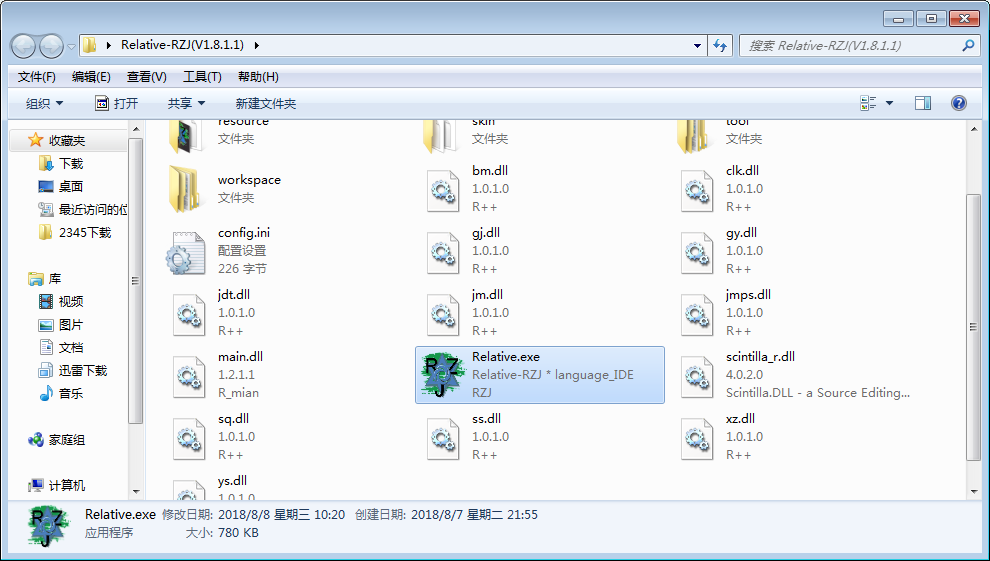
编译后直接点击运行下方会输出运行结果



\*

## <Relative目录存储详解>

1.主目录

主目录下存放的是Relative.exe主程序解密驱动程序和主程序依赖的\*dll以及卸载程序

2.Bin

这里存放运行支持文件。

3. configure

本目录存放语言库文件夹

格式是：语言名文件夹(比如：JAVA)

type文件夹(预留层，为了方便以后扩展)

子工程文件夹(比如：安卓工程、普通工程)

create(存放新建工程时默认的内容)

dlls(存放库Load.dll文件)

lib(存放Load.dll依赖的翻译库文件，如：<基本库>)

config.txt

config\_filter.txt

property.ini

config 存放代码框配置信息

config\_filter 存放编译或翻译时要过滤翻译的指定正则表达式，用<fgf>分隔create目录>存放新建工程时的默认内容

lib目录>存放当前语言库的支持库

文件内部解释：

GL 表示 语言脚本常量

GCLX 表示 当前语言类型 （0 编译语言 或 脚本语言,1 其它）

GCHZ 表示 该语言的代码存放的文件的后缀 （许多地方 优先第一项）

YXWJ 表示 该语言运行文件的后缀 （许多地方 优先第一项）

ZS 表示 该语言的单行语句注释

DZS 表示 该语言的多行语句注释

4.language

存放界面语言文件

翻译文件分三个部分

\*.lan 和 \*.lanX以及\*.lanG

只要将其以文本方式打开并翻译为新的语言。

导出后重命名。即可完成语言的翻译！

5. resource

LOGO.png

rzj.jpg

res.rcl

xmt文件夹

目前有01-07 7个目录分别存放界面依赖的图片组。放入图片以数字顺序为名.bmp为后缀。图片组文件会在Relative启动时自动生成。

内部所有的BMP图片一般为16\*16像素

在修改其中BMP文件后必须删除其中的\*.ril文件

1.【\*.ril文件删除后会在下次启动中生成】

2.【文件夹中的文件名必须从 “0.bmp” 开始】

3.【文件夹内图片必须使用BMP格式】

6.skin

内部存放界面配色文件

分两部分

\*.rskin和\*.she皮肤文件(此文件可有可无)

7. tool

工具存放文件夹

分两层

第一层文件夹（用于分类工具）

第二层为\*.exe可执行文件

8. workspace

存放代码

## <Relative进程通信操作>

通信使用专门提供的通信模块

代码是

R\_专用消息发送 (“PASS”, 到字节集 (“<TS> IDE代码操作器<fgf>启动完成！”))

以下为“PASS”通信的操作命令

执行的代码格式>>

<代码> 代码值1 代码值2 代码值......

其中若代码值中需要用到空格必须用 <fgf> 来代替！

下面是代码及格式>>

{在有工程的前提下才有效！}

<charu> 内容 [用处：向代码区当前位置插入“内容”]

<-> 位移 [用处：调整代码区光标位置，当“位移”为正数则向右移，反之左移]

<SET\_SQ> 行号 [用处：设置“行号”行的书签]

<SET\_DD> 行号 [用处：设置“行号”行的断电]

<XZ\_H> 行号 [用处：选中“行号”行]

<XZ\_WZ> 起点 终点 [用处：选中从“起点”到“终点”范围内的内容]

<XZ\_D> 内容 [用处：将选中区的文本替换为“内容”]

<R> [用处：撤销一步]

<N> [用处：重复一步]

<CD> ID [用处：模拟单机菜单栏中ID为“ID”的菜单项]

<>>> 类型 内容 [用处：当“类型”=0，则将“内容”插入到当前位置，否则插入到最后]

<GET\_HANG> [用处：为语言DLL库提供行文本和行号]

<GET\_ALL> [用处：为语言DLL库提供全部代码文本]

<GET\_DUANDIAN> [用处：将当前全部断点提供给语言库使用]

<GET\_ALL\_E> [用处：将当前全部代码翻译后提供给语言库使用]

<ADD\_BL> 变量应用范围|变量数据类型|变量名|项目数值

[用法：按格式加入变量项]

<ADD\_KU> 库名称|库引用地址|项目数值

[用法：按格式加入引用库]

<SC\_KU> index [用处：删除第“index”项库引用]

<SC\_BL> index [用处：删除第“index”项变量引用]

{在无工程的前提下也有效！}

<Tree> 树型框号 图片组文件位置 [用处：设置“树型框号”(1~5) 的图片组为“图片组文件位置”的文件]

<TS> 内容 [用处：在提示栏中显示“内容”]

<SC\_SC> 内容 [用处：在输出栏中追加显示“内容”]

<SC\_QK> [用处：清空输出栏的内容]

<B\_L> +/- [用处：“+”显示变量夹，“-”隐藏变量夹]

<C\_L> +/- [用处：“+”显示常量夹，“-”隐藏常量夹]

<K\_L\_J> +/- [用处：“+”显示库链接夹，“-”隐藏库链接夹]

<Tree1\_KZ> index [用处：扩展树型框1的第 index 项]

<Tree1\_SS-KZ> index [用处：[\_树型框1\_即将扩展(index)事件]一般跟在<Tree1\_KZ>后面]

<Tree1\_SS> index [用处：收缩树型框1的第 index 项]

<Tree1\_JS> [用处：初始化树型框1内容和状态]

<Tree2\_SX> [用处：树型框2刷新内容]

<ZCX\_TGB> [用处：使主程序可以被关闭]

<ZCX\_FGB> [用处：使主程序不可以被关闭]

“PASS\_tree5”通信没有命令

直接提供树型框.项目，字节集。 是用于给外部提供智能提示的接口

“PASS\_tree3”通信没有命令

直接提供树型框.项目，字节集。 是用于给外部提供代码大纲视图的接口

提供的树型框.项目值的 取项目数值() 结果必须为行数。提供双击跳转。

## <语言库的Dll接口和事件>

事件：

.子程序 load, , 公开, 初始化加载

.参数 窗口句柄, 整数型

.参数 总事件为, 子程序指针

.参数 库位置为, 文本型

.参数 取光标所在位置文本, 子程序指针

这里的 效果类似 进程通信的”PASS”

调用子程序 (总事件为, , , “”)‘无返回值

这里的取光标所在位置文本用于取当前代码框中指定位置的光标所在地

调用子程序 (取光标所在位置文本, ,文本型[返回取到的文本] , 整数型[指定光标位置])

.子程序 loadR\_cdx, , 公开, 编译菜单项

.参数 主窗口菜单句柄, 整数型, , （最高层的菜单句柄）

.参数 程序, 整数型, , （第二层的菜单句柄）1 程序 2 编辑 3 查看 4 工具 5 界面 6帮助 7 禁止栏

.参数 编辑, 整数型

.参数 查看, 整数型

.参数 工具, 整数型

.参数 界面, 整数型

.参数 帮助, 整数型

.参数 禁止栏, 整数型

.参数 左菜单句柄, 整数型, , （第二层的菜单句柄）左菜单句柄

.参数 右菜单句柄, 整数型, , （第二层的菜单句柄）右菜单句柄

.参数 子夹头句柄, 整数型, , （第二层的菜单句柄）存放子夹头菜单项句柄

.参数 右\_库句柄, 整数型

.参数 下\_常亮句柄, 整数型

.子程序 load\_gjsj1, 文本型, 公开, 挂接事件 窗口菜单被选择 [可有可无]

.参数 菜单ID, 整数型

.子程序 load\_gjsj2, 文本型, 公开, 挂接事件 系统菜单被选择 [可有可无]

.参数 菜单ID, 整数型

.子程序 load\_gjsj3, 文本型, 公开, 挂接事件 鼠标经过菜单项 [可有可无]

.参数 菜单ID, 整数型

.子程序 load\_zsxk, , 公开, 判断事件 左树型框右键后执行

.参数 现行选中项文本, 文本型

.参数 是否为文件夹, 逻辑型

.参数 路径, 文本型

.参数 项目索引, 整数型

.参数 项目父索引, 整数型

.参数 总工程路径, 文本型, , 树型框最顶层时的路径

.子程序 load\_ysxk, , 公开, 判断事件 右树型框右键后执行

.参数 现行选中项文本, 文本型

.参数 是否为文件夹, 逻辑型

.参数 路径, 文本型

.参数 项目索引, 整数型

.参数 项目父索引, 整数型

.子程序 load\_zsyjc, 文本型, 公开, 注释与解除注释

.参数 是否为注释, 逻辑型

.参数 是否多行, 逻辑型

.参数 文本, 文本型

.子程序 load\_R\_zfsr, , 公开, 代码框输入字符时触发

.参数 字符代码, 整数型

.子程序 load\_qsxkzjj, 文本型, 公开, 取树形框的项目（支持库）

.子程序 DM\_FG, 文本型, 公开, 设置代码及风格号

.子程序 AI\_tall, 文本型, 公开, 智能提醒

.参数 输入的文本, 文本型, , 当输入的内容为多项继承时会提供分割 如：（form.left）会传递为：“form . left”

.子程序 AI\_retrue, 整数型, 公开, 智能提醒 返回 选中项(返回值为 -1 时自动选择前面的长度替换文本 0插入文本 其它则 选取指定长度替换文本)

.参数 选中项文本, 文本型

.参数 选中项数值, 整数型, ,

.子程序 KU\_SM, , 公开, 取库或命令的说明

.参数 选中项文本, 文本型

.参数 选中项数值, 整数型, ,

.参数 选中项层次, 整数型

.子程序 OPEN\_KU\_path, , 公开, 打开库位置

.参数 选中项文本, 文本型

.参数 选中项数值, 整数型, ,

.子程序 GET\_interpret, 文本型, 公开, 返回鼠标停留事件

.参数 停留获得的内容, 文本型

.子程序 GET\_HANG, , 公开, 提供当前行文本行文本

.参数 行文本, 文本型

.参数 行数, 整数型

.子程序 GET\_ALL, , 公开, 提供当前全部文本

.参数 文本, 文本型

.子程序 DJ\_ZJFK, , 公开, 鼠标左键放开事件

.参数 位置, 整数型

.参数 功能键状态, 整数型

.参数 光标位置, 整数型

.子程序 DJ\_ZJAX, , 公开, 鼠标左键按下事件

.参数 位置, 整数型

.参数 功能键状态, 整数型

.参数 光标位置, 整数型

.子程序 GET\_DUANDIAN, , 公开, 提供当前代码断点

.参数 文本型, 文本型, , 用","分割

.子程序 GET\_GCXX, , 公开, 提供当前工程信息

.参数 工程文件夹, 文本型

.参数 工程项位置, 文本型

.参数 工程项编码, 逻辑型, , 真 则使用编码：UTF-8，假 则使用：ANSI

.参数 工程名, 文本型

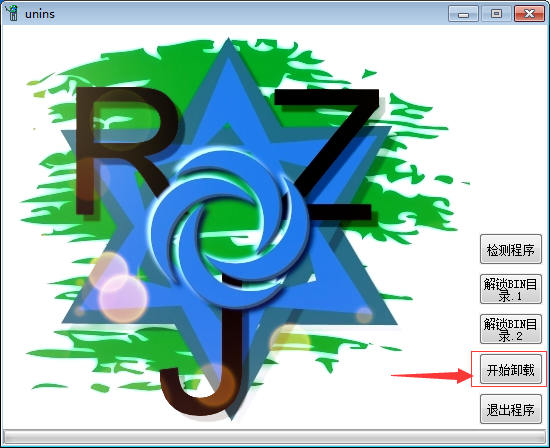
.参数 对应的树型框索引, 整数型

.参数 对应的树型框父索引, 整数型

.子程序 R\_OVER, , 公开, 库程序结束将被释放时执行

## <卸载>

1.双击打开

2.点击”开始卸载”即可

注：累死我啦…..懒得再详细了。语言库的接口和事件详细可以看 我公开的代码。里面是完整的。不过目前这里面的也是完整的。很多东西没介绍。不过对于基本使用已经没有问题了。