**1.StarUML**

StarUML(简称SU)，是一种创建UML类图，生成类图和其他类型的统一建模语言(UML)图表的工具。StarUML是一个开源项目之一发展快、灵活、可扩展性强。我们一般都是使用这个软件画一个时序图和软件图等，特别是软件图是我们常用的，用来反映各个模块各个类之间的关系。StarUML是一款很全面，很好用的UML画图工具。相比PlantUML那种使用代码画图方式，StarUML的拖拽式更简单易用。唯一不足之处，在反向工程时，只能生成类图，不能生成类之间的关系。

▪完全免费：StarUML是一套开放源码的软件，不仅免费自由下载，连代码都免费开放。

▪多种格式影像文件：可导出JPG、JPEG、BMP、EMF和WMF等格式的影像文件。

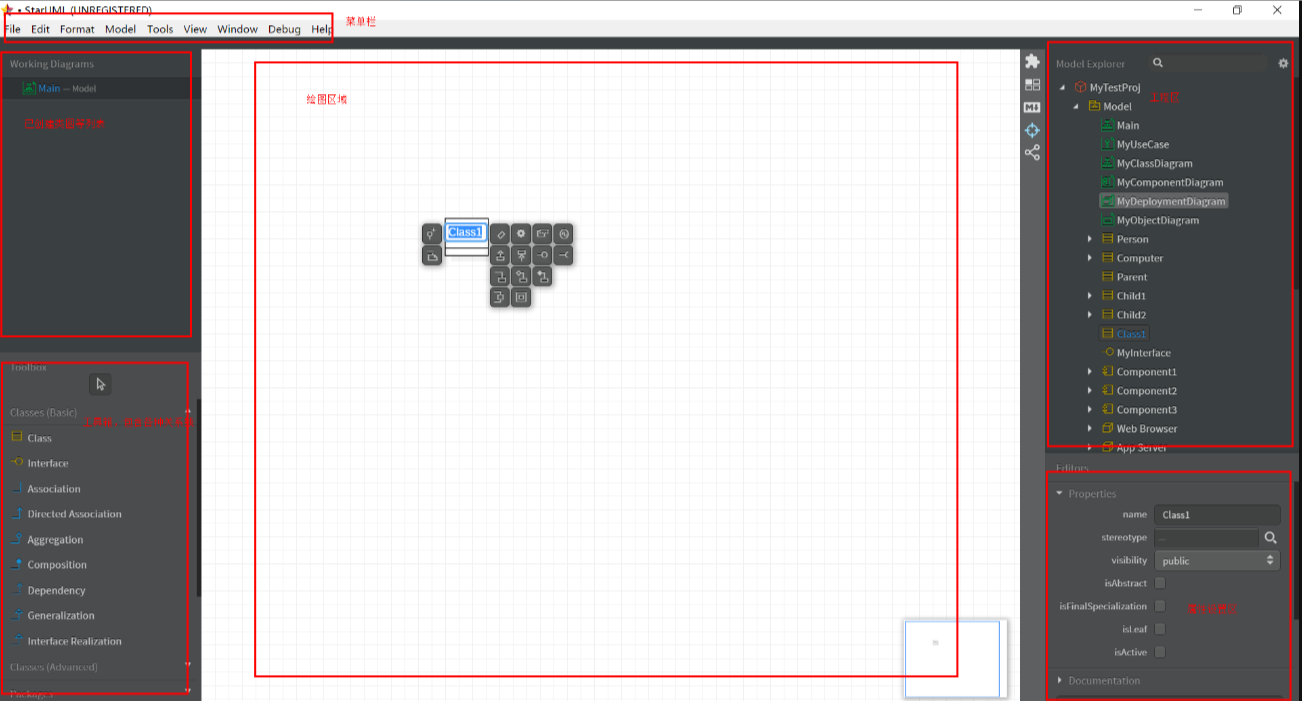
▪语法检验：StarUML遵守UML的语法规则，不支持违反语法的动作。

▪正反向工程：StarUML可以依据类图的内容生成Java、C++、C#代码，也能够读取Java、C++、C#代码反向生成类图。

▪支持XMI：StarUML接受XMI 1.1、1.2和1.3版的导入导出。

▪支持模式：支持23种GoF模式(Pattern)，以及3种EJB模式。

1>界面说明



说明：默认打开starUML后，会默认进入类图模式，各模块区域功能如下：

▪菜单栏： 最上方是菜单栏，新建工程啥的，具体不详细介绍；

▪已经建类图列表：左上方列表显示已经创建的图，比如类图，时序图等；

▪工具箱：左下方工具箱，显示当前类型的图可以使用的工具，主要是各种连线或者图形，是最常用的区域；

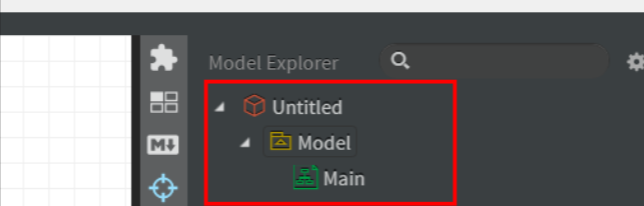
▪绘图区域：中间白色格子区域，就是绘图区域；

▪工程区：右上方区域是工程区，显示工程，model和各种已经绘出的图形；

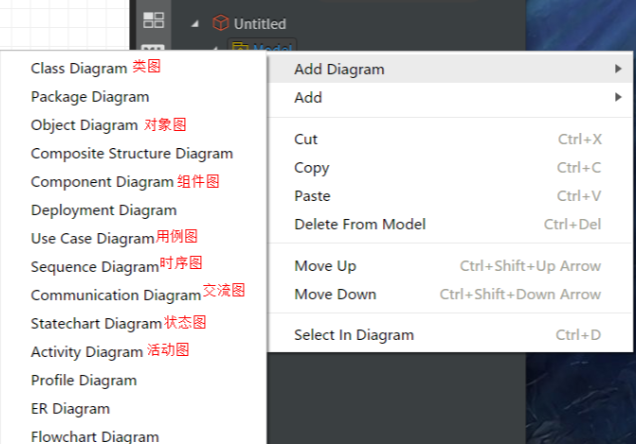
▪属性编辑区域：当画出来一个图形或者一根线时，这个区域会显示这个图形的各种属性，可以修改；

1. 类图

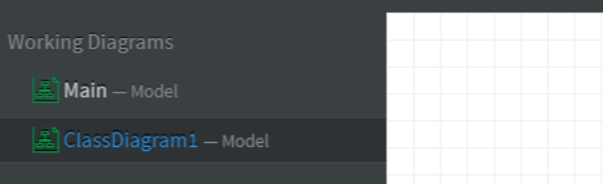
打开starUML工具时，默认进入的就是类图模式，右上角工程区如下显示：



▪创建类图:工程区->选择Model->右键->选择【Add Diagram】，然后如下图：



可以发现，StarUML支持的图类型有十几种，选择【Class Diagram】，表示类图，这时会默认创建一个名字为“ClassDiagram1”的类图（名字自随便修改），显示在左上角的图列表区



▪画类图（Class Diagram）

接下来我们，就可以画类图了，现在左下角工具箱显示了在类图模式下，可以使用的工具列表：

▪用例图（Use Case Diagram）：

工程区->选择Model->右键->选择【Add Diagram】->选择【Use Case Diagram】，然后左下角工具箱变成了用例图的模式

▪时序图（Sequence Diagram）

工程区->选择Model->右键->选择【Add Diagram】->选择【Sequence Diagram】，然后左下角工具箱变成了时序图的模式：

▪组件图（Component Diagram）

工程区->选择Model->右键->选择【Add Diagram】->选择【Component Diagram】，然后左下角工具箱变成了组件图的模式

▪部署图（Deployment Diagram)

工程区->选择Model->右键->选择【Add Diagram】->选择【Deployment Diagram】，然后左下角工具箱变成了部署图的模式

▪协作图（Comunication Diagram）

工程区->选择Model->右键->选择【Add Diagram】->选择【Comunication Diagram】，然后左下角工具箱变成了协作图的模式

▪对象图（Object Diagram）

工程区->选择Model->右键->选择【Add Diagram】->选择【Object Diagram】，然后左下角工具箱变成了对象图的模式

▪活动图（Activity Diagram）

工程区->选择Model->右键->选择【Add Diagram】->选择【Activity Diagram】，然后左下角工具箱变成了活动图的模式

**2.PlantUML**

· PlantUML是开源的，有一个相对比较成熟的网站，通过简单的UML语言来实现想要的效果，有点极客范

· PlantUML有比较详细的各类语言的guide文档

· PlantUML做了很多的适配，比如常用的编译器eclipse，IDEA intelliJ都有对应的插件，同时还和Maven和JQuery都做了集成，还提供了war包形式，可以在本地的JavaEE容器（比如Tomcat）中运行起来

·PlantUML语言简单直接易学，就好像Markdown语法一样，简单高效

PlantUML是一个开源项目，允许用户使用纯文本语言创建UML图表。PlantUML的语言是特定于域的语言的示例。

支持快速绘制UML图

·类图

·时序图

·用例图

·活动图

·组件图

·状态图

·对象图

·部署图

·定时图

同时还支持以下非UML图

·线框图形界面

·架构图

·规范和描述语言 (SDL)

·Ditaa diagram

·甘特图

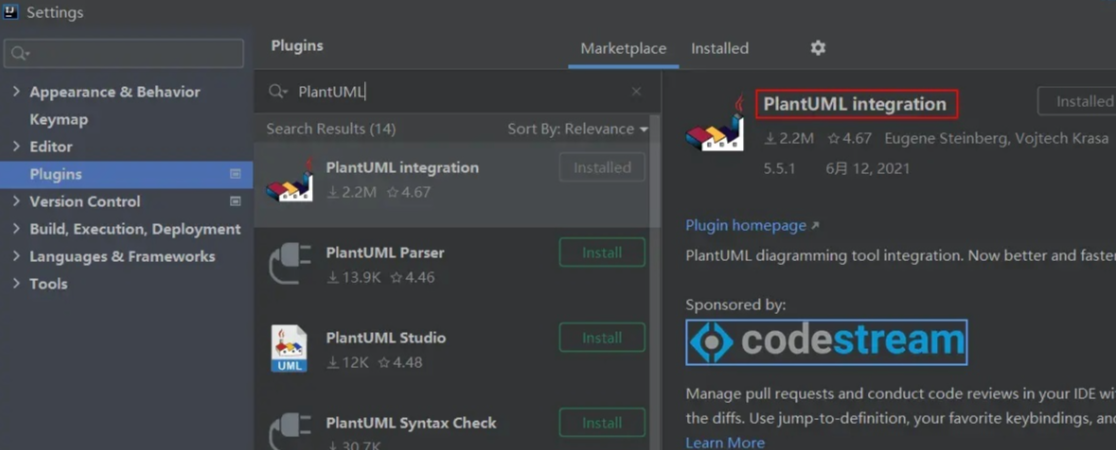
·思维导图

·Work Breakdown Structure diagram

·以 AsciiMath 或 JLaTeXMath 符号的数学公式

·Entity Relationship diagram

在IDEA中以PlantUML插件的方式安装



简要说明如何使用PlantUML插件绘制时序图、用例图、类图、活动图

·时序图：选择New->选择PlantUML File->选择Sequwnce

通过PlantUML提供的语法来生成时序图，其重要的参数说明如下：

- title可以用于指定UML图的标题；

- actor可以声明人形的参与者；

- participant可以声明普通类型的参与者；

- as可以给参与者取别名；

- ->可以绘制参与者之间的关系，虚线箭头可以使用-->；

- 在每个参与者关系后面，可以使用:给关系添加说明；

- autonumber我们可以给参与者关系自动添加序号；

- activate和deactivate可以指定参与者的生命线。

且右键时序图时还可以生成一个在线访问的链接，直接访问这个链接就可 以在线访问UML时序图并进行编辑

·用例图：选择New->选择PlantUML File->选择Use Case

其重要的参数说明如下：

- left to right direction表示按从左到右的顺序绘制用例图，默认从上到下；

- package可以对角色和用例进行分组；

- actor可以定义用户；

- usecase可以定义用例；

- 角色和用例之间的关系可以使用-->来表示。

·类图：选择New->选择PlantUML File->选择Class

其重要的参数说明如下：

- class可以定义类；

- 在属性和方法左边加符号可以定义可见性；

-表示private，#表示protected，+表示public；

- 通过<|--表示类之间的继承关系

·活动图：选择New->选择PlantUML File->选择Activity

其重要的参数说明如下：

- start和stop可以表示流程的开始和结束；

- :和;中间添加文字来定义活动流程节点；

- if+then+endif定义条件判断；

- switch+case+endswitch定义switch判断。

**3.ModelMake**

ModelMaker简称为MM，是一个与Delphi紧密绑定的CASE建模工具。MM弥补了Delphi IDE在建模设计上的不足。MM与其他 CASE工具最大的不同就是其强大的实时同步引擎，你的设计可以直接映射成代码，在代码上的修改可以自动逆向反映到设计模型。以“设计－编码－精练”方式代替了传统的“设计－编码”的工作方式。MM是一个强大的UML建模工具，她还支持时下流行的“设计模式”。

1>MM的默认布局（菜单条下的工具栏上有一个下拉框，选中“basic”）分四大块。

·左上是【树状视图区】包括：【图】（Diagrams），【类视图】（Classes）和【单元视图】（Units）。

·左下方是【类成员区】。

·右上是【编辑区】包括：【图编辑页】（Diagram Editor），【实现编辑页】（Implementation）和【单元代码编辑页】（Unit Code）。

·右下是【消息】和【文档】区。

2>MM中的工程（Project）

先创建一个新工程（或模型),选择菜单“File|New”。你将得到一个空白的工程，该工程包含一个默认的祖先TObject 和 IUnknown。

然后选择菜单“File|Save As...”命名为FirstMM，你将得到FirstMM.mpd文件。

.mpd是以下的文件集合的打包：

· <model>.mpr; 包含工程设置；

· <model>.mma; 包含与工程相关的宏；

· <model>.mmb; 包含代码模型数据；

· <model>.mmc; 包含模型中的文档；

· <model>.mmd; 包含图；

· <model>.mme; 包含事件类型定义；

· <model>.mmf; 包含工程信息。

3>新建类图

首先点击【编辑区】的【图编辑页】（Diagram Editor），把注意力都放在这个区域哦。

· 点击类似“＋”的“add Diagram”按钮（将鼠标移到按钮上，稍停就会有该按钮的名称。以下提到的按钮我们都以MM提示名称来表示）。

· MM弹出对话框要我们输入图名（name）输入：testClass。

· 在下拉框（type）中选择 Classes Diagram后点击“ok”。我们那就得到了一张名为testClass的空白类图。

· 这时你发先图左侧的一派按钮都可用了，点击“add Class”按钮（应该是第一个）。

· 这时鼠标带了一个“＋”在图上找个合适的位置点击一下，MM弹出了对话框要求输入类名class name等类的基本信息。输入class name：TMMobject。选择祖先类Ancestor：TObject。现在我们已经得到了一个类并且看到了他的图形。

· 重复步骤4和步骤5，我们加上另外两个类TMMFace和TMMBoday，他们的祖先类都是TMMobject。我们的图上有四个类TMMobject、TMMFace、TMMBoday和Tkk。那么如何展现他们之间的关系呢？

· 当然我们可以自己来画箭头,但是这个办法好好像很笨。一个更好的办法是用MM提供的向导。

步骤如下：先用鼠标圈住（或用键盘Ctrl＋A）以上四个类。点击右键“Wizards|visualize class relations...”这是MM自动的生成了TMMFace和TMMBoday指向TMMobject的箭头。或者点击“Wizards|visualize related classes ...”这样MM会把关联到的类全部显示出来，我们的类图上就会多了TObject。

4>添加类成员

· 选中TMMFace在右键弹出菜单“class|add property”。

· name输入：color，根据需要设置其他各项（相信会delphi都应该明白是什么意思，我们在这里就用默认的选项）。

· ok后会发现在左下方的【类成员区】出现两成员Fcolor:integer和color:integer。在图的空白区域上点右键“visual style|Diagram properties”,选“symbol style”页,将“Project member type filter”前的勾去掉。把“fields、properties”等根据需要选中，即可在图中显示

· 如上所述，给TMMFace加入方法WashFace。给TMMBoday加入属性：face:TMMFace和方法bath。

1. 实现类的方法

在MM中我们可以直接用Delphi代码实现类的方法,在【类成员区】选中要实现的方法，点击【编辑区】，【实现编辑页】（Implementation）写代码。

6>生成Delphi代码

· 在界面左上【树状视图区】的【单元视图】（Units），点“add new unit”按钮。

· 在path Alias选择路径。Relative File name输入“MMUnit”单元名，ok。

· 工具栏按钮“Unlock Code Generation”按钮解锁。

· 在【单元视图】（Units）点击按钮“Generate”。

· 在你本地的文件夹（刚才设置的path Alias）下找找，一个MMUnit.PAS出现了，打开看看Unit的开头都有了，却没有发现TMM们的身影？为什么？

· 在【单元视图】（Units）把“Class not assigned to units”下的类都拖到“MMUnit”下。

· 点击按钮“Generate”。

7>逆向到模型

· 我们给类TMMFace增加一个成员（str:string）使其代码如下：

TMMFace = class(TMMobject)

private

Fcolor: Integer;

str:string;

public

procedure WashFace;

property color: Integer read Fcolor write Fcolor;

end;

· 在Delphi中点菜单“ModelMaker|Jump to ModelMaker ”回到MM中。

· 在【单元视图】（Units）选“refresh Import”按钮。

· 查看TMMFace发现【类成员视图】果然有了str:string。

8>完全的逆向工程

· 【单元视图】（Units）选“Import Unit..”

· 选择你要导入的.pas文件。

· ok后你在【单元视图】（Units）可以看见多了一个新的单元。

· 【图编辑页】（Diagram Editor），右键菜单“Wizards|visualiztion Wizards”选刚导入的单元，单元下的类选到右边。next，finish，我们的图上多了刚才选中类图。

**4.Visual Paradigm**

Visual Paradigm for UML (VP-UML)是一款功能强大、跨平台、使用最便捷、直观的UML建模和CASE工具，它可以整合在其他CASE工具或者其他IDE工具中，这种“一站购物式”的软件解决方案要远远优于常规的Model-Code-Deploy的开发进程。

Visual Paradigm for UML支持UML建模、数据库建模、对象关系映射、逆向工程和Java双向工程等等，功能异常强大。

· Visual Paradigm的入坞环境Dockable Environment

VP-UML采用了入坞环境，它允许你拖动用户界面组件来完成定制你喜欢的工作环境。你能存储此环境作为一个视图文件，且以后再次打开使用。VP-UML允许你使用不同的视图来实现不同的目标。

· 使用入坞环境

入坞环境由一系列的窗口所组成，通过调用入坞框架。入坞框架既可以是单个的浮动窗口，也可以是多个窗口的嵌套。

你可以用鼠标点击标题栏title bar，拖动任何入坞框架到你想要的地方。

要注意你拖动框架时的灰色线，它指示了你现在拖动到的位置。

· 拖动入坞框架到另一个容器Contianer中

如果你拖动入坞框架到另一个容器Contianer时，灰色线将改变其形状以适应容器的区域。

· 在制表窗格中拖动制表

在制表窗格中每一个tab都是一个入坞框架，能够被拖动和放入。你可以在外面拖动一个tab，使它成为一个浮动窗口。

· 拖动一个入坞框架进入制表窗格,跟6正好相反

你可以方向操作6。

· 入坞框架的自动隐藏

每一个入坞框架都能被设置为“自动隐藏”auto hide。这意味着当它处于非活动状态时，它可以自动隐藏。

· 方框或窗格

VP-UML提供了多种方框，有：

1）图表导航方框Diagram Navigator Pane

2）模型方框Model Pane

3）类库方框Class Repository Pane

4）逻辑视图方框Logical View Pane

5）ORM方框ORM Pane：它从指定的类路径和数据库中显示了一系列类和数据库表。

6）模板方框Stencil Pane

7）属性方框Property Pane

8）图表视图方框Diagram Overview Pane

9）文档方框Documentation Pane

10）消息框Message Pane

· 布局和视图

VP-UML提供了多种视图，有：

1）存储视图Saving Perspective

2）打开视图Opening Perspective

3）管理视图Managing Perspective

4）重设视图Resetting Perspective

· 打开指定的对话框

打开指定的对话框属于顶层窗口，能让你浏览和编辑模型元素和图表的细节。它能通过按鼠标右键弹出窗口选择Open Specification来实现调用。

VP-UML的按钮一般有五种：重设Reset，同意OK，取消Cancel，应用Apply，帮助Help。

· 在图表元素之间导航

你能通过打开指定的对话框在图表元素之间实现导航。有两种方式来实现：一是通过图表元素之间的关系（如父子关系）来导航；另一种是打开指定的对话框在图表元素之间实现导航。

**5.Junit**

JUnit是一个Java语言的单元测试框架。它由Kent Beck和Erich Gamma建立，逐渐成为源于Kent Beck的sUnit的xUnit家族中最为成功的一个JUnit有它自己的JUnit扩展生态圈。多数Java的开发环境都已经集成了JUnit作为单元测试的工具。

注意：Junit 测试也是程序员测试，即所谓的白盒测试，它需要程序员知道被测试的代码如何完成功能，以及完成什么样的功能

使用 Junit 能让我们快速的完成单元测试。通常我们写完代码想要测试这段代码的正确性，那么必须新建一个类，然后创建一个 main() 方法，然后编写测试代码。如果需要测试的代码很多,那么要么就会建很多main() 方法来测试，要么将其全部写在一个 main() 方法里面。这也会大大的增加测试的复杂度，降低程序员的测试积极性。而 Junit 能很好的解决这个问题，简化单元测试，写一点测一点，在编写以后的代码中如果发现问题可以较快的追踪到问题的原因，减小回归错误的纠错难度。

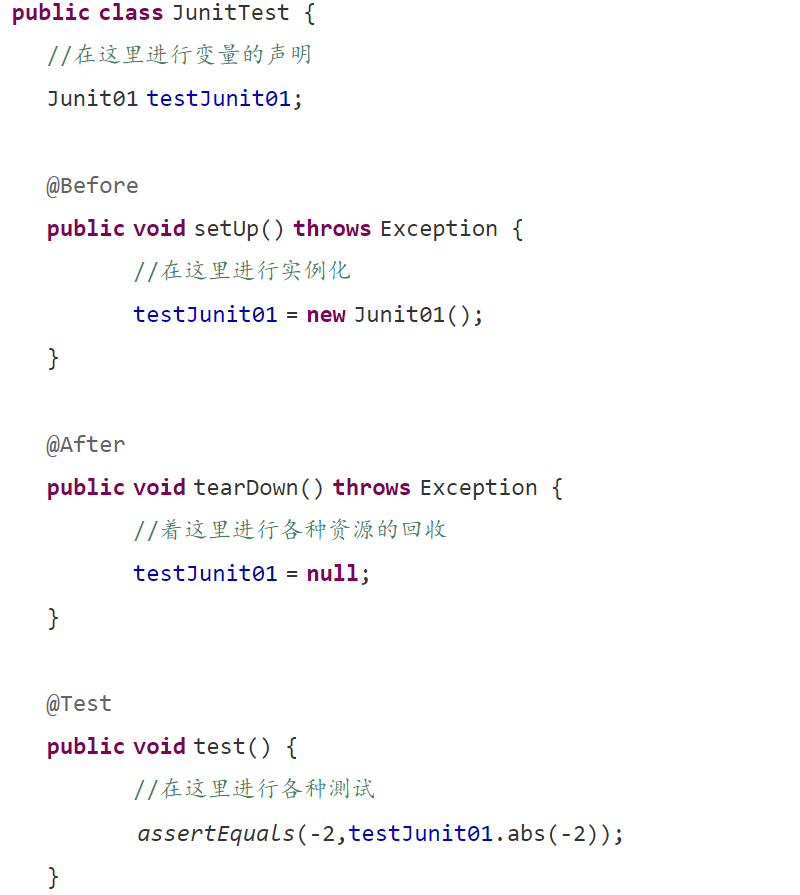
1>Junit的使用

使用Junit的一般步骤为：

· 准备測试条件并未測试分配资源

· 设置断言并进行測试

· 清除回收各种资源。



单击右键 -> Run As -> JUnit Test.系统将自己主动为我们准备@Before中的环境，并执行@Test中的測试。进度条为绿，则说明測试通过。进度条为红，则可能为Failure，也可能为Error。前者则说明assertEquals设置的断言有误，代码本身没错，我们仅仅须要检查逻辑。后者则说明代码本身有错误，存在异常。

2>断言(assert的使用方法)

JUnit中提供了大量断言，它表示方法执行后的值是否满足我们的期望。断言的使用方法大同小异，以下我们列举一些经常使用的断言：

assertEquals 要注意对应类型的精度范围

assertNull，assertSame 是否引用的同一个对象

assertTrue , assertFail 经常使用在不可能到达的地方，如捕获异常后

1. Test

在Junit 4之前，Junit一直是运用反射机制如testXXX来寻找须要測试的方法。Junit 4引入了注解模式，仅仅须要在方法名称前加上@Test注解即表示此为測试方法而无需在意方法的名称。

这里的@Test还有两种使用方法，即能够传递两个參数:

A>expected

表示期望捕获的异常，假设没有捕获异常或者捕获的异常并非期望的，都会产生Failure。

B>timeout

表示方法运行的毫秒数，超过设置的值就会产生Failure。

4>Ignore

被忽略的测试方法：加上之后，暂时不运行此段代码

5>Before

每一个测试方法之前运行

6>After

每一个测试方法之后运行

7>BeforeClass

方法必须必须要是静态方法（static 声明），所有测试开始之前运行，注意区分before，是所有测试方法

8>AfterClass

方法必须要是静态方法（static 声明），所有测试结束之后运行，注意区分 After

**6.Umbrello**

Umbrello UML 建模工具是一个跨平台的 UML 绘制的自由软件，运行在类Unix系统和Windows上,其优点是免费，跨平台，可正向逆向工程，缺点是Windows下安装稍微有点复杂。

Umbrello 能够处理所有标准的UML图表类型。它可以对 C++、IDL、Pascal、Ada、Python和Java编写的代码进行反向工程；以及引入 XMI 文件，由来自 PHP或 Perl 和各种编程语言的外部工具生成。

Umbrello 允许模型内容由输出成DocBook 和 XHTML格式发布。

如果用户愿意，他可以 在单个 XMI 文件中组合多个相关图表. 这些将被组织在不同的视图中 (逻辑、用例、组件等。)，反过来又可以包含图表或文件夹，用于进一步分类。

一种类型图的创建仅限于某种类型的视图. 在一个视图中，图表可以在文件夹之间自由移动。

1>支持输出的编程语言

· ActionScript

· Ada

· C#

· C++

· D语言

· IDL

· Java

· JavaScript

· Pascal

· Perl

· PHP

· Python

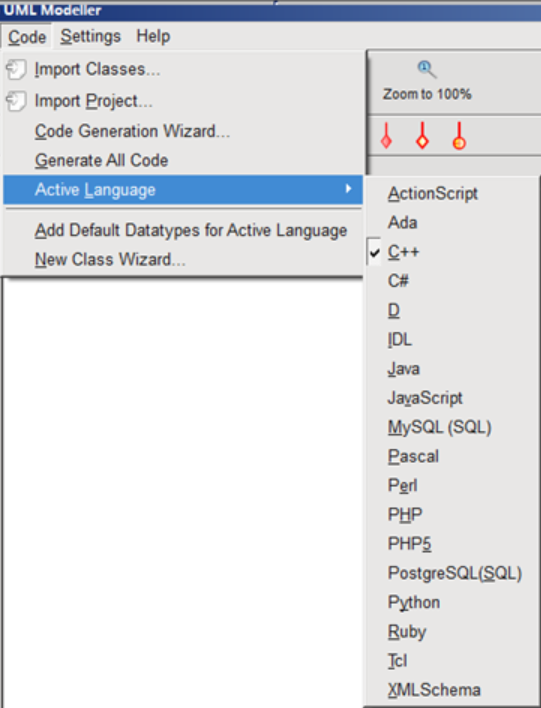
· Ruby

· SQL

· Tcl

· XML Schema

如下图所示



Import Classes/Project：可以通过导入工程或者头文件来引入类图。

Code Ceneration Wizard/Generate All Code：根据类图来生成程序代码

Active Language：选择不同的语言

2>Umbrello设置

点击Settings->Configure Umbrello.

a. 默认语言，General->Default Language中，即启动Umbrello之后默认的语言；

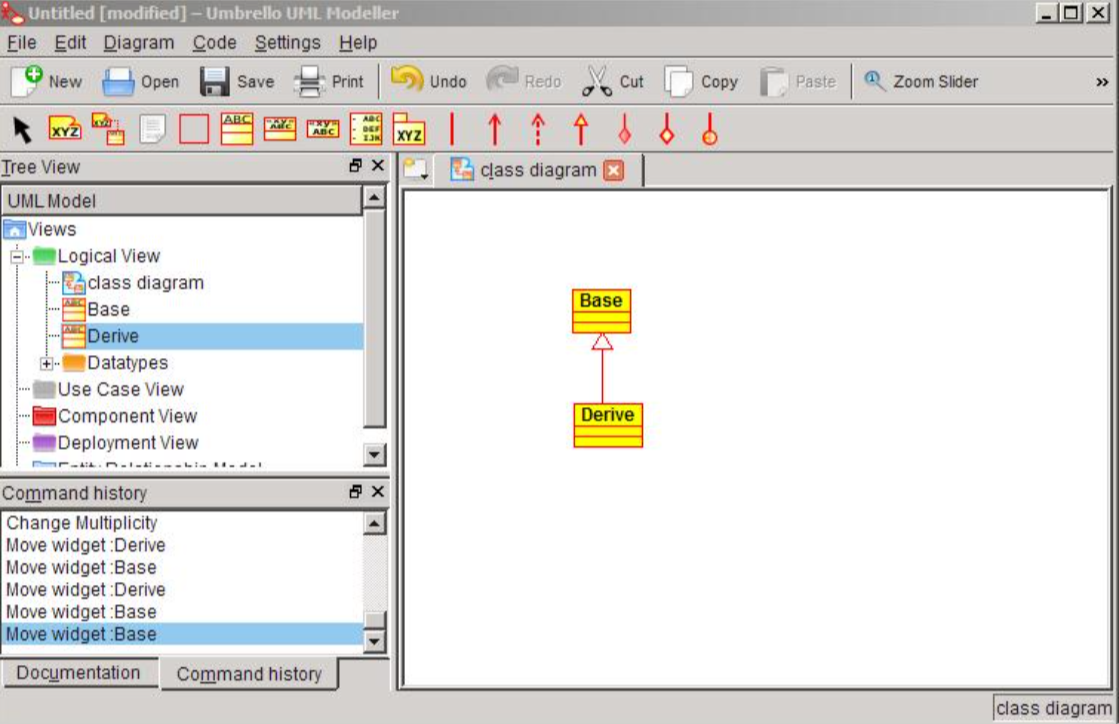
b. 支持的语言，Code Generation->Language中，若默认语言为C++，而你想生成Java，那么在这里选择。同时可以看到Umbrello支持的语言；

c. 生成代码的路径，Code Generation->Folders，这个很重要，如果是一个不存在的路路径，就如法生成代码。

3>Umbrello初体验

a. 点击上图中的“类图”（第3行第6个图标），生成一个名为Base的类，然后再创建一个名为Derive的类。

b. 点击上图中的“实现关系”（第3行第14个图标），然后先点击Derive类，再点击Base类，可以看到如下图的UML类图：



该图表示类Derive集成了Base，Derive实现了Base。

c. 正向工程。点击工具栏Code->Code Generation Wizard，然后选择要生成代码的类。完成之后可以在Settings中设置的代码路径中看到四个文件，Base.h, Base.cpp, Derive.h, Derive.cpp。

如果点击工具栏Code->Generate All Code，则会直接生成所有类的代码。

d. 逆向工程。

删除左边UML Model中的Logical View中的Base和Derive，并且删除绘图区域中的图。

右击上图中的Logical View，选择Import Class，选择刚才生成了Base和Derive类的.h文件。这时又可以看到Logcal View中有了Base和Derive两个类，将这两个类分别拖动到绘图区。Umbrello自动帮我们生成了类图以及类之间的关系。