StarUML 使用说明 -指导手册

原著： Stephen Wong 翻译：火猴

1. 综述

StarUML 是一种生成类图和其他类型的统一建模语言 (UML) 图表的工具。这是一个用 Java 语言描述的创建类图的简明手册。

StarUML(简称 SU), 是一种创建 UML 类图, 并能够自动生成 Java的“ stub code ”的工具。 SU 也可以做 JAVA 逆向工程，以产生相应的 UML 图表。

在本教程中，我们将使用 SU 设计一个 pizza 饼。执行下列步骤，可以创建如下面所示的 UML 图。SU 可以生成反映类结构的代码， 而不是任何对象的具体行动。因此，在使用 SU 创建图表后，你会为此 stub code添加剩余的功能性代码，填写每种方法本来应该做的事。

2.安装

首先，我们必须先安装将要使用的软件。 StarUML ，是一个开放源码软件， 遵循 GPL 协议许可 （GNU 公共许可证） ，并免费提供下载。

3.启动

安装以后就可以启动该程序。

4.添加新工程

然后，一个名叫： New Project By Approach 的对话框会弹出。 选择“ Empty Project 并且按下 "确定"。这里建议你不要勾选 "设置为默认的做法 " 复选框。

5.选择模块

在右边的“ Model Explorer框中选定 “ Untitled模块。

6.添加模块

通过“ Model主菜单，或右击选定的模型，可以 " Add/Design Model "

7.添加类图

通过“ Model主菜单，或右击选定模型，可以 “ Add Diagram/Class Diagram：”

8.设置 profile

通过“ Model/Profile...菜单去设置工程所需的 profile。这决定了工程所使用的规则和约定。一定要包含 "Java Porfile" 这一项目。

9.保存工程

立即就保存工程，这样在出现问题的时候，您就不会丢失信息。

从“ File 菜单，选择“ Save,”并选择一个地方以保存工程。 你的 StarUML 项目现在应该看起来的是这样的：

10.创造图表

现在，开始真正创造图表，从默认就在屏幕的左边的 “ Toolbox选择“类 ”图标，然后左键单击 diagram 窗口的某处。这样就使用通用名字创造了一个新的类。双击，将类改名为 Circle。

11.添加属性

右击图中的目标，在弹出菜单中选择 “ Add”中的“ Attribute 被标示为绿色)，为其添加一个属性 (或者域)，填入期望的名字 “ _radius。”

具体的数据类型，在属性面板（右下侧的窗口） ，由双打字，在 " 类型"时段。 在窗体右下边的 Properties面板中， 找到“ Type”输入框，输入 double 作为 _radius 属性的类型。

类的内部数据（域 /属性）都是私有的，因为他们是严格由类内部使用的。所以，在 Properties面板中将 _radius 设置为“私有 ”。

12.继续进行设计

重复同样的过程， 添加所谓的名字叫做 Rectangle的类和 double 型的私有成员 _width 和_height。 (下面者段话是使用方面的主意事项，总感觉翻译部太好，九原文搬上来了)You may notice using the "Model Explorer" on the right is faster to add these, but do however note that adding the classes and interfaces themselves in this toolbox (instead of using the toolbox on the left and clicking on the palette to create the object) will not create the objects in the diagram.

13.创造 IShape interface

从 toolbox 中，选择“ Interface，并”点击图表的某处。将其改名为 IShape。创建以后，选中它。

在顶部工具栏，选择 “ Stereotype Display下拉按钮，将值改变为“ None。”这将改变以往的圆形形状，使其变为成长方形。

还是在顶部工具栏， 取消选中 " Suppress Operations "。这将使我们能够看到接口所拥有的方法。

向 IShape接口添加返回值为 double 的 getArea 方法。

可以通过右击 interface的图标，在弹出菜单中点击红色的“ Operation”按钮，然后输入 getArea。

设定返回值类型。在 “ Model Explorer中展开 IShape 节点，右击你刚刚创建的 getArea 方法，并选择 “ Add Parameter。”在 “ Properties框中”，将参数的名子变为空，将 “ DirectionKind变为” “ RETURN”，将 “ Type变为 dboule。

将 IShape和 getArea的 IsAbstract 属性框打上勾，他们在图标上的名字将变为斜体。这是 UML 的标准，表示这是接口或者其他纯虚实体。

14.添加类和接口的关系

可以通过从 toolbox 中选择表示 “ Realization的箭头，并从 Circle 拖拽向 IShape，使 Circle 实现接口 IShape。重复同样的过程， 为 Rectangle 添加实现关系。 这是添加了 Circle 和 Rectangle对于 IShape接口的实现关系。

如果想使连接线表现为直角的方式，右击连接线，并选择 " Format/Line Style/Rectilinear"菜单。你通过这种方式，使箭头重叠在一起，可以使你的图看起来更整洁。

15.添加类基于接口的行为

由于 Circle 和 Rectangle类都实现了 IShape接口, 就必须有同样的行为 (方法)。

在 “ Model Explorer面极中， 复制 getArea 法(按 Ctrl-C 或者右键点击并选择 Copy 菜单），并粘贴到 Circle 和 Rectangle类。

这些实现了的方法在 Circle 和 Rectangle类中都不是抽象的，而是具体的。这是因为他们实际上是执行一些特定行为 （例如， 为一个圆形和长方形分别计算面积） ，所以不要勾选 IsAbstract 框。

16.你的图现在应该是这样的：

17. 添加 Pizza 类

向 Pizza 添加 double 型的私有域 _price 。

添加返回 double 类型的共有操作 getPrice 。

18. 为 Pizza 类添加 IShape 的引用

从 toolbox 中选择 " DirectedAssociation " 箭头，点击 Pizza 类，并向 IShape 拖拽 。

选中箭头，在右边的 “ Properties ” 框上，将name一栏改为 “ has- a ”， “ End1.Aggregation ” 一栏改为 “ AGGREGATE(这个图示说明 Pizza 和 shape 对象是 “ 聚合 “ 的关系)。

将“ End2.Name”一栏改为_shape 。这样就自动为 Pizza 添加一个名字为 _shape, 使用 IShape 接口的私有域, 的所谓 _shape 型 ishape 以 pizza 饼。

将“ End2.Visibility ”改为私有。

为_shape 创建一个“ 获得者 ”方法, 名字叫做 getShape, 返回 IShape 。这就是创建一个行为, 名字是 getShape, 返回 IShape 。

19. 为 pizza 类添加构造函数

为 Pizza 添加构造函数, 右击, 在弹出的“ Add”菜单中选择“ Operation ”。从这里, 增加一个普通的带有 dboule 型 price 参数和 IShape 类型 shape 参数的操作

增加一个输入参数, 就像之前增加了一个返回型的输出参数一样, 你指定的参数的名称, 如价格和形状等, 以及适当的数据类型。

为 Circle 增加一个带有 double 型的 radius 参数的构造函数。

为 Rectangle 增加一个带有 double 型 width 和 height 参数的构造函数。

20. 你的图现在应该是这样的：

21.添加 Test_Pizza 类

为了说明 UML 类图更多的功能, 又增加了一个叫做 “ Test_Pizza”的类, 它用作 测试 目的, 并使用到 Pizza 和 IShape类。

两个类之间的关系有多种形式。 举例来说, 一个类可以实例化另一个类, 而不是将其作为一个成员。又或, 一类的方法可能需要另一个类作为输入参数, 保留一个引用仅仅是为这个方法的执行。

通过从 toolbox 中选择“ Dependency”箭头, 从一个类拖向他所以来的类, 来添加不通类之间的依赖关系。在这个例子中, Test_Pizza依赖于 Pizza, Circle 和 Rectangle类, 因为它实例化了它们。

从 Properties box选择 name属性, 或者双击图表上的 “依赖线 ”, 可以为依赖关系添加标签。特别的是, 当一类实例化另一个类, 我们会把依赖线叫做“ instantiates ”

你可以选中并拖动依赖线的标签, 以达到更美观的效果。

依赖关系不会影响代码生成。

22.你的图现在应该像本文最开始所示。

23.对你的图随意做些修改。

你还可以拖动你的类图，并且使箭头以不通的方式展示 (使箭头显示为直线，选择一个箭头，右击它，弹出菜单中选择 “ Line Style，并选择“ Rectilinear 。你一定要体验这个工具，并去了解它。

24.保存项目

在“ File菜单中，选择 “ Save”SU 的所有资料只有一个单一的项目文件，所以你目前应该只有一个文件生成。

25.导出

将图表导出为其他格式，例如图片等，是非常有用的。您可以通过选择“ File菜单的“ Export Diagram , ”并且选择合适的文件类型来执行改操作。

26.生成 Java stub 代码：

点击主菜单的 “ Tools>Java 菜单，选择 “ Generate Code ”
从对话框中选择你的模块 (这里可能 Model1), 点击“ Next。”
为了使你的模块或者图标的所有类都生成 stub code，选择“ Select All然 ”后按“ Next。”
选择一个有效的输出目录， “ Next。”
In the "Options Setup", be sure to check both "Generate the Documentation by JavaDoc" and "Generate empty JavaDoc". All other checkboxes should be unchecked. Then press "Next".在“ Options Setup，请务必选中“ Generate the Documentation by JavaDoc，“ Generate empty JavaDoc 所”有其他复选框不选中， “ Next。”
现在 StarUML 将从你的图产生代码，点击 “ Finish退出对话框。
现在，您可以编辑生成的代码，以增加应用。

27.添加实现代码

现在就开始定义程序实际做的事情，例如，为你图标中的类描述添加实现代码。

使用 DrJava添加代码，为相关的类， .Java文件添加代码。 代码会和你使用 HW02 是一样的。（注意：为 Test_Pizza些代码，最好由 DrJava自动生成，而非手工在 StarUML 里面创建。我们这里只是为了说明。）
记得那 IShape的 getArea()方法是抽象的，因此没有代码。
请您像代码范例一样添加注释。这种注释是 “ JavaDoc风格的。 关于JavaDoc您将会在随后学到更多。

28.逆向工程

StarUML 还可以从现有的 Java代码创建一个类图， 这被称为“ reverse engineering ，”当你想从现有的代码生成图表，或者你修改了 SU 生成的代码，并且想在图表中反应出来的时候，逆向工程功能就非常有用了。

通过图表或者 DrJava 这样的文本编辑器去反复工作的过程，称作“ round-trip engineering 。”这也是面向对象变成中的一个基本过程。

到主菜单栏中选择 “ Tools/Java/Reverse Engineer„. 可以将现有的代码逆向工程。

选择 Java代码所在的目录，并点击 “ Add”或“ Add All按钮，将它们包括在逆向工程过程中，然后单击 “ Next 。”

选择你想将类加入的模块，这里可能是 “ Model1,”然后“ Next。”

在 Option Setup 里面：

确认“ public, ” package, ”“ protected和“ ” private是选中的（这是默认设置）。

同样，在默认情况下，单选按钮 “ Create the field to the Attribute也是 ”选中的。

除非你想 SU 创建其他东西，例如布局很糟糕的包含所有类的图表，不要选中“ Create Overview Diagram框。”

当你对选项做了检查后，点击 “ Run。”

SU will now import the classes in the selected files into your model. Click "Finish" to exit the dialog when it is complete.苏现在进口班，在被选定的文件到你需要的产品型号，点击 "完成"退出对话框时，就完成了。

SU 会向你的模块添加导入的类， 但不是你的图表。 为了将它添加到您的图，只需要简单地从 Model Explorer 拖动它们即可。