> 本文由 简悦 SimpRead 转码， 原文地址 www.imooc.com

上一节我们学了用例图，本节我们将介绍类图。

《手册》设计规约章节，对类图有下面的规定：

> 【强制】如果系统中模型类超过 5 个，并且存在复杂的依赖关系，使用功能类图来表达并明确类之间的关系。

那么我们来思考几个问题：

- 什么是类图？
- 我们为什么要使用类图？
- 该如何画类图呢？

2. 什么是类图？为什么要使用类图？

类图是软件工程的统一建模语言中的一种静态结构图，该图描述系统的类和其它类和属性之间的关系。

类图描述了面向对象系统中的类的结构，以及类之间的关系，类图中也会体现出约束，也会展示出类的属性。

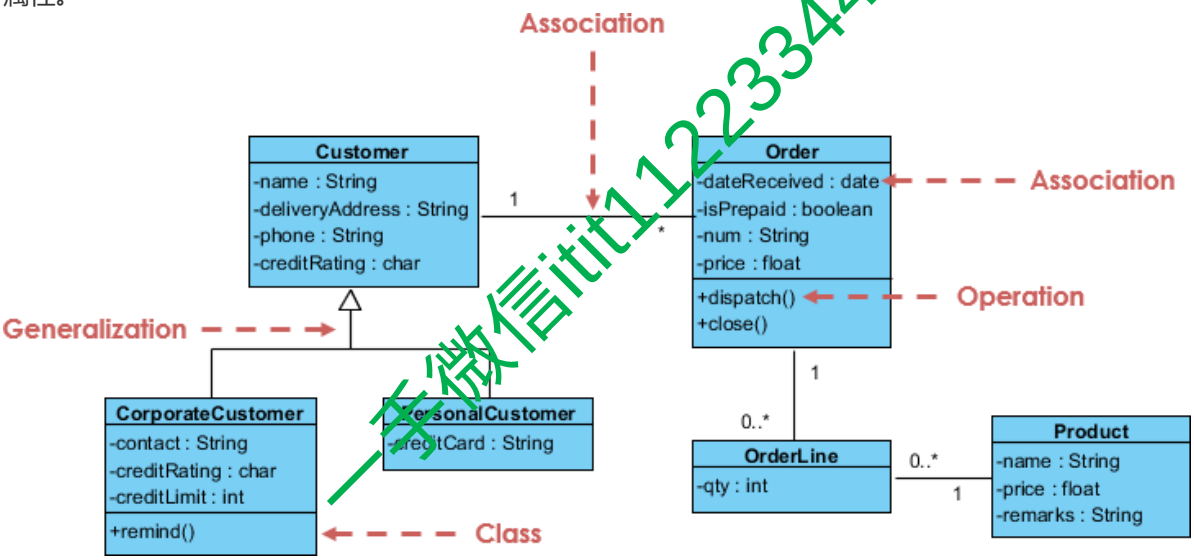


图 1：类图示例（图片来自 Visual Paradigm）

面向对象编程语言就不得不提到类和接口，在大型系统中类的数量众多，正如《手册》所说，当核心类较多且相互之间存在依赖关系时，需要使用类图将这种关系清晰地表达出来。

类图通常用在软件生命周期中的**详细设计阶段**，另外学习一定不要教条，当我们想从类视角来学习和研究源码时，类图也会派上大用场，下文将详细说明这一点。

3.1 了解基本组件

3.1.1 类图符号

如下图所示，类图中，类主要由三部分构成：类名部分、属性部分和操作部分，这三个部分通过横线隔开。

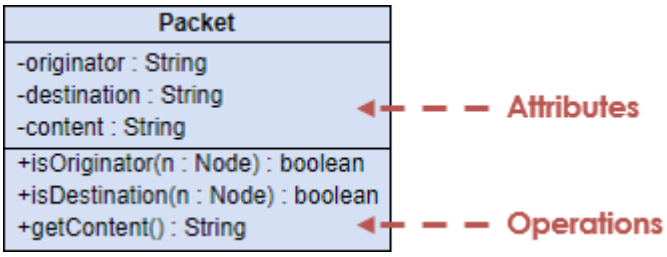


图 2：类图示例（图片来自 Visual Paradigm）

属性的表示形式为：“可见性 属性名称：属性类型”。 上图中的 - content : String，就等价于 Java 在源码中的：

```
private String content;
```

操作的表示形式为：“可见性 函数名：返回值类型”。

3.1.2 关系

就类图而言，关系主要指类之间的联系，主要包括依赖关系、关联关系、聚合关系、组合关系和泛化关系。

依赖关系是五种关系中耦合最小的一种关系，类 A 要完成某种功能引用了类 B，那么 A 就依赖了 B，如类 A 的成员函数的返回值、参数、局部变量或静态方法调用使用了 B。依赖关系的生命周期较短，一般在方法调用期间存在。

图 3：依赖关系

关联主要分为两类：单向关联和双向关联。 关联关系的生命周期更长，随着类的初始化而产生，随着对象的销毁而结束。

其中**单向关联**，用带箭头的实线表示。如下图表示左侧类用到类右侧的类。关联关系表示类比较强的依赖，且这种关系由一方来维护，如学生依赖老师，学生类中有一个老师的成员属性。

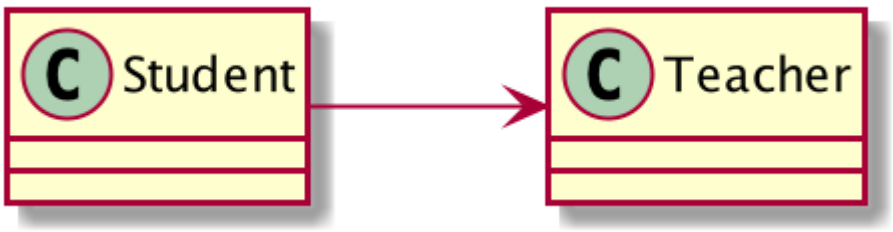


图 4：单向关联

双向关联如下图所示，通过实线连接，表示两个类相互引用。

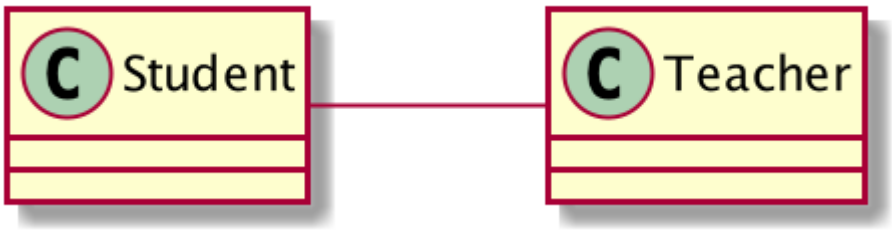


图 5：双向关联

聚合关系使用实心线加空心菱形表示。聚合用来表示集体和个体的关系，是一种 "has-a" 的关系。例如班级和学生之间，公司和员工之间就存在聚合关系。

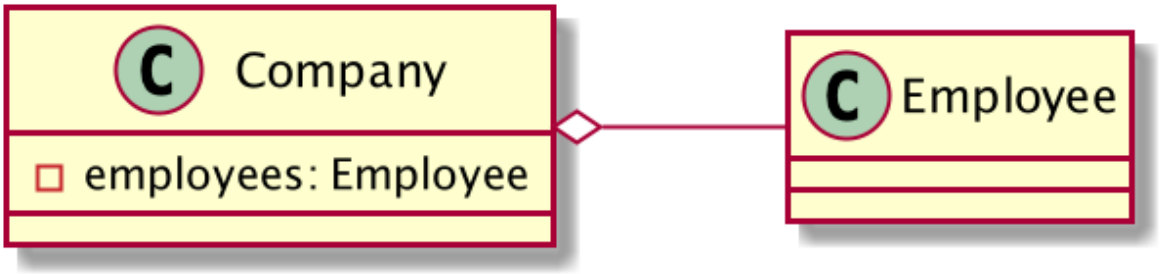


图 6：聚合关系

组合关系是关联关系的一种特例，表示一个整体和其组成部分的关系，是一种 "contains-a" 关系。如人作为一个整体，包括头部、腹部、腿部等组成。

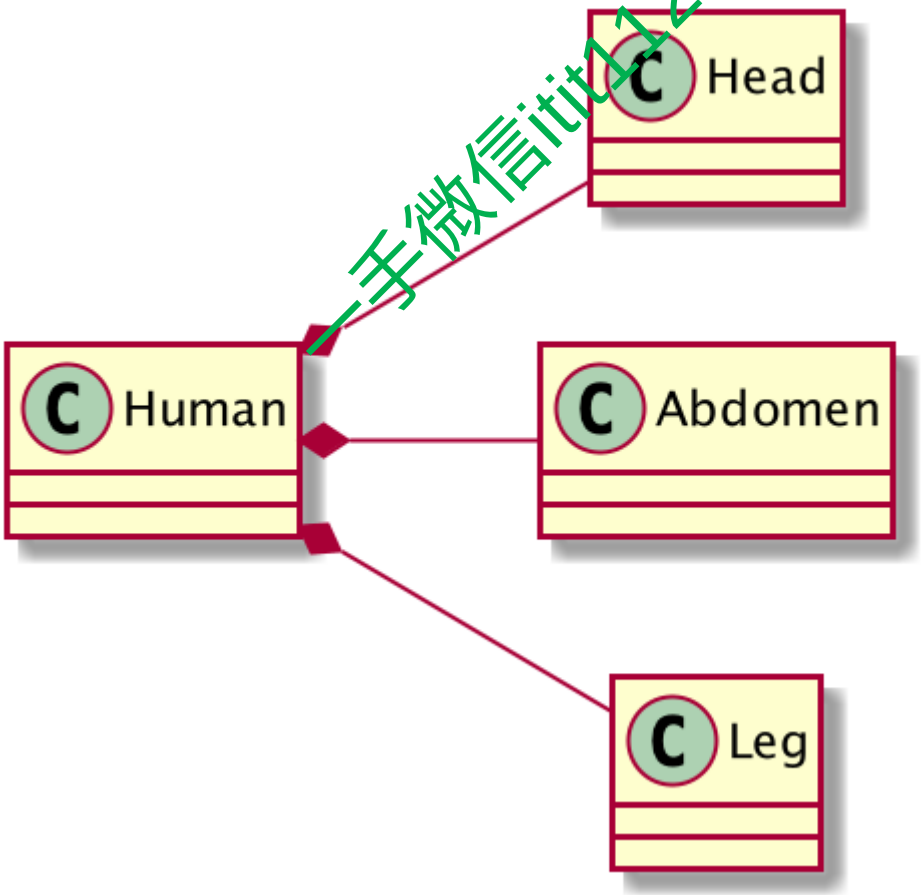


图 7：组合关系

泛化关系 主要指类与类之间的继承关系以及类与接口之间的实现关系。以 Integer 为例，实现了 Comparable 接口，继承了 Number 类，而 Number 类又实现了 Serializable 接口，可用如下类图表示。实现接口通过待箭头的虚线表示，而继承类则通过带箭头的实现表示。

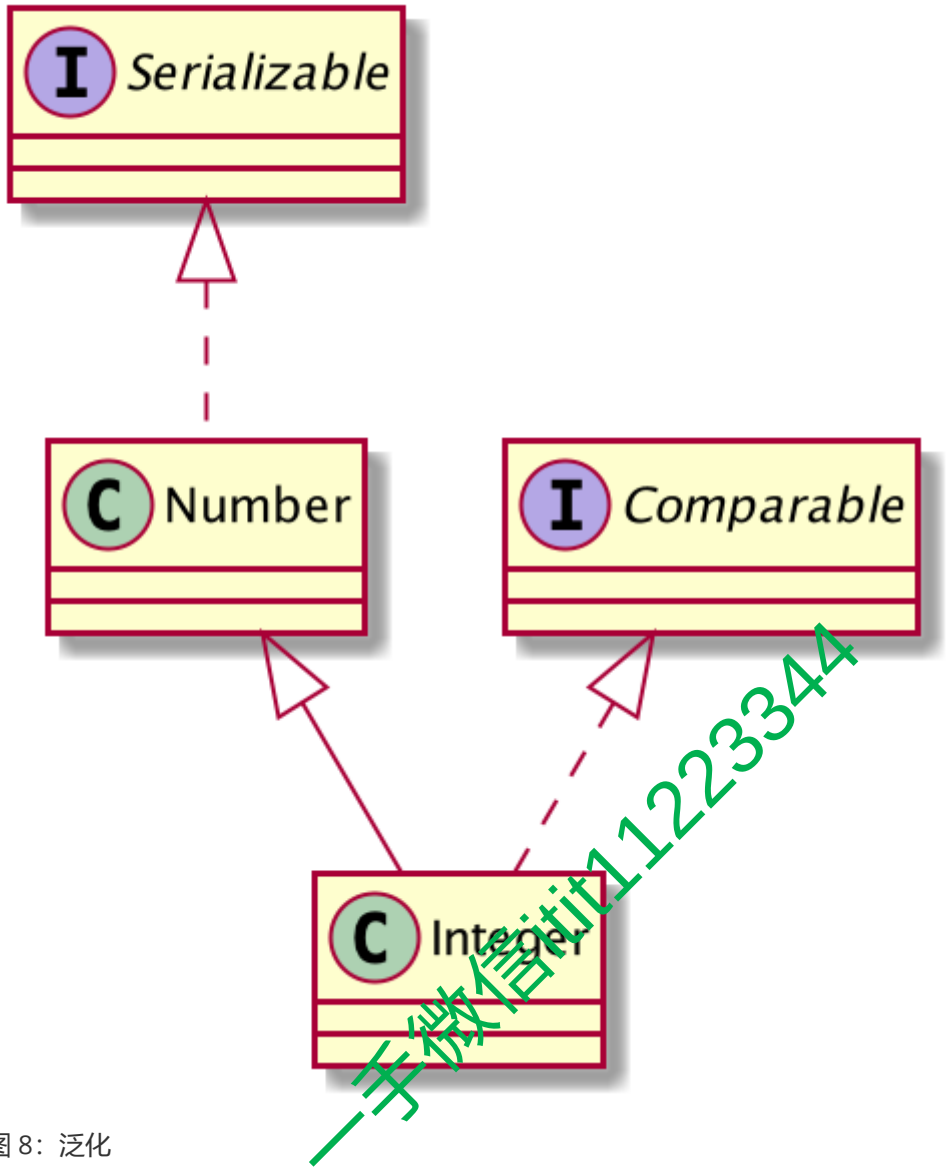
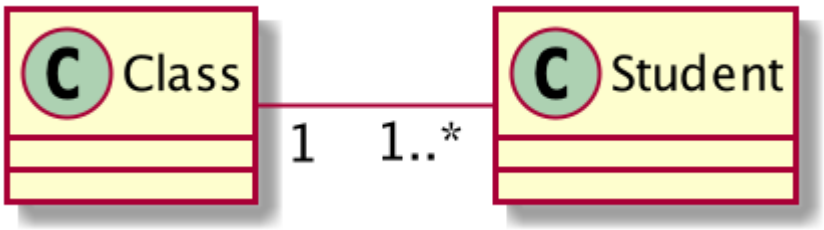


图 8：泛化

关系需要通过标识来表示对应数量之间的关系。其中 `1` 表示只有一个；`0...1` 表示 0 个或者 1 个；`*` 表示多个；`0..*` 表示 0 个或多个；`1..*` 表示 1 个或多个。例如一个班级有一个或多个学生，但是一个学生只能在一个班级上课，则可用下图表示。



图：数量标识

讲完了类之间的联系，我们接下来讲一下可见性。

类图的可见性分为四类：公有 (public) 用、私有 (private)、受保护（protected）、包私有（package private），和 Java 的访问修饰符的含义一致。

其中 public 用 `+` 表示， private 用 `-`, protected 用 `#` 表示， package private 使用 `~` 表示；

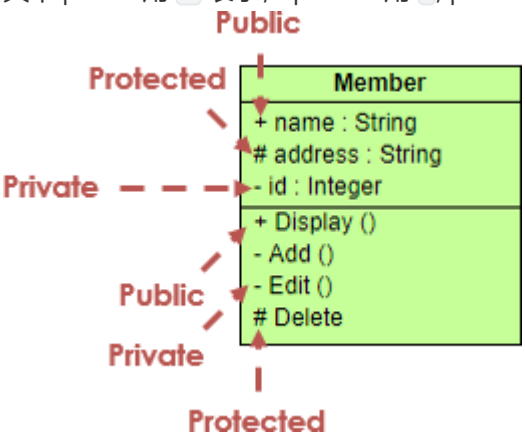


图 10：可见性（图片来自 Visual Paradigm）

注意不同的画图工具的最终呈现形式会有差异，有些会直接显示上述符号，有些会通过不同的图形符号表示（如 PlantUML），而有些则会通过锁的颜色等方式来表示（如 IDEA 自带的类图插件）。

3.2 作图

通过以上的学习我们对类图的基本组件有了基本的认识，下面我们从大家熟悉的几个类来了解类图的画法。

3.2.1 有源码的情况

下面讲解一个技巧，作为常用的开发工具 IDEA 支持根据源码生成类图。

可以选择某个类，右键–> Diagrams --> Show Diagram --> Java Class Diagrams 来自动根据源码绘制类图。

下面是 Number 的类图：

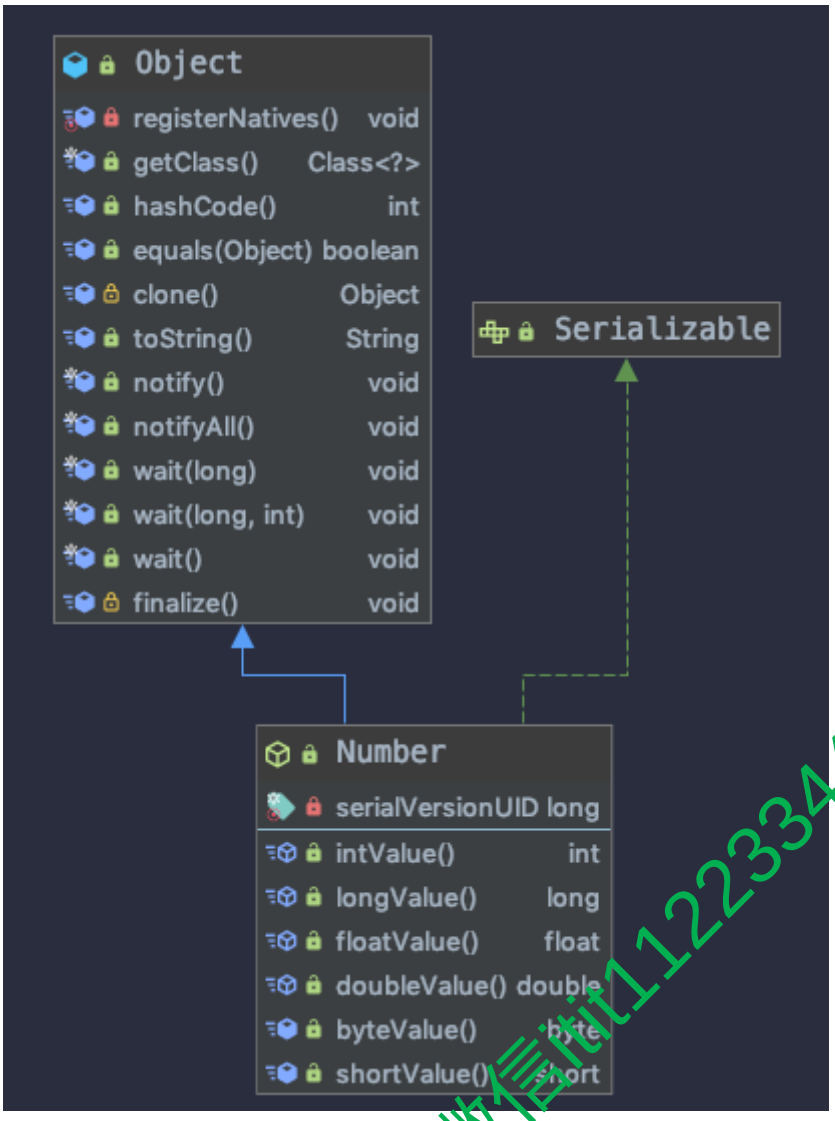


图 11 : Number 类的类图

如果想了解其它类和它的关系，可以直接将其它类拖到该页面上。可以通过页面的菜单开启关闭类图是否包含属性、操作等。

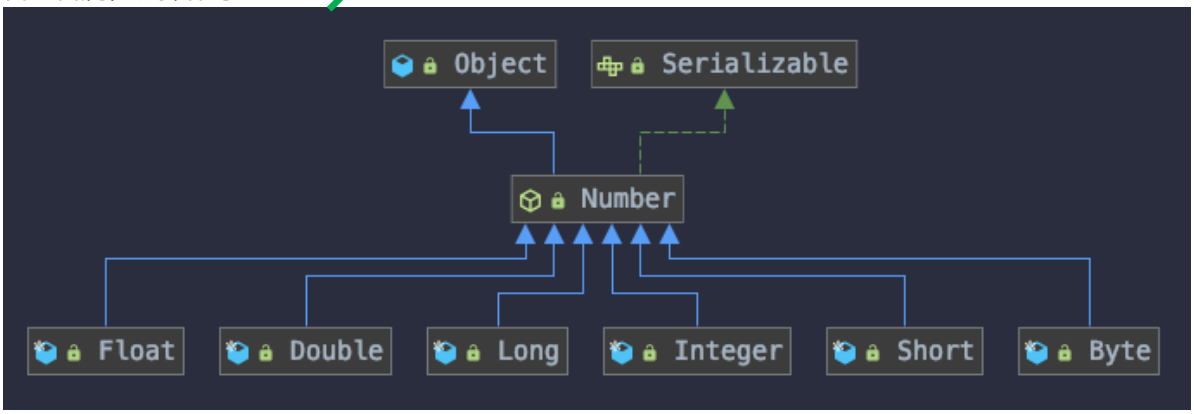


图 12 : 类图示例

通过 IDEA 根据源码生成类图，可以清晰地了解不同类之间的关系，清楚地了解目标类的各种属性和函数，对我们学习源码有极大地帮助。也是我们学习画类图的一个非常权威的参考资料。

如我们通过 IDEA 自带的类图工具绘制 fastjson 核心类之间关系的类图，可以通过全局的视角来观察核心类之间的关系，从先整体后局部的思维来学习源码。

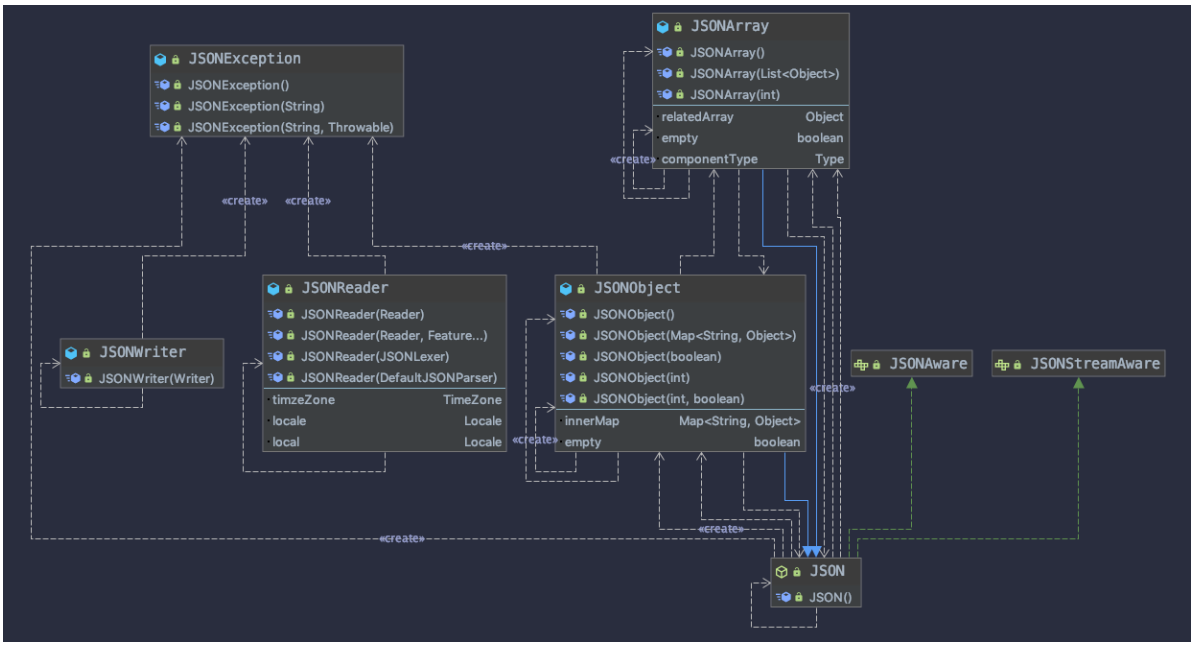


图 13 : fastjson 核心类的类图

3.2.2 没有源码的情况

假如有这样一个场景：

> 一个交易系统中主要包括顾客、订单、订单详情、商品、支付等核心类。其中顾客的属性包括顾客的姓名、收货地址、联系方式和是否活跃。订单的主要属性包括订单的创建日期，订单详情、订单的状态、支付方式。订单的支付方式可以有多种，如信用卡、现金、支票、转账。订单的状态包括创建状态、海运中、投递完成和支付完成这几个状态。订单详情包括商品数量，税信息，订单详情类还要提供计算总金额和重量的功能。订单详情中包含商品，商品有重量和描述信息，商品对象中要提供获取商品重量的接口，还要提供根据重量获取价格的功能。

根据上述描述可以绘制出核心的类、属性和行为，然后再去梳理类之间的关系。

类之间的关系需要通过需求或者常识得出。一个顾客可能未下单，即有零个订单，也可能多次下单从而产生多个订单。一个订单只有一种状态，因此订单和状态之间是一对一的关系。由于支持多种支付方式，因此订单和支付方式之间是一对多的关系。支付方式可以是一个抽象类，每种具体的支付方式是其实现类，因此它们之间是泛化关系。订单必然订单项，最少为一个。而一个商品有可能被一次或多次下单而成为订单项，也可能没有被下单过，因此一个商品可能对应零个或者多个订单订单项。

根据这个场景描述和分析就可以绘制出如下：

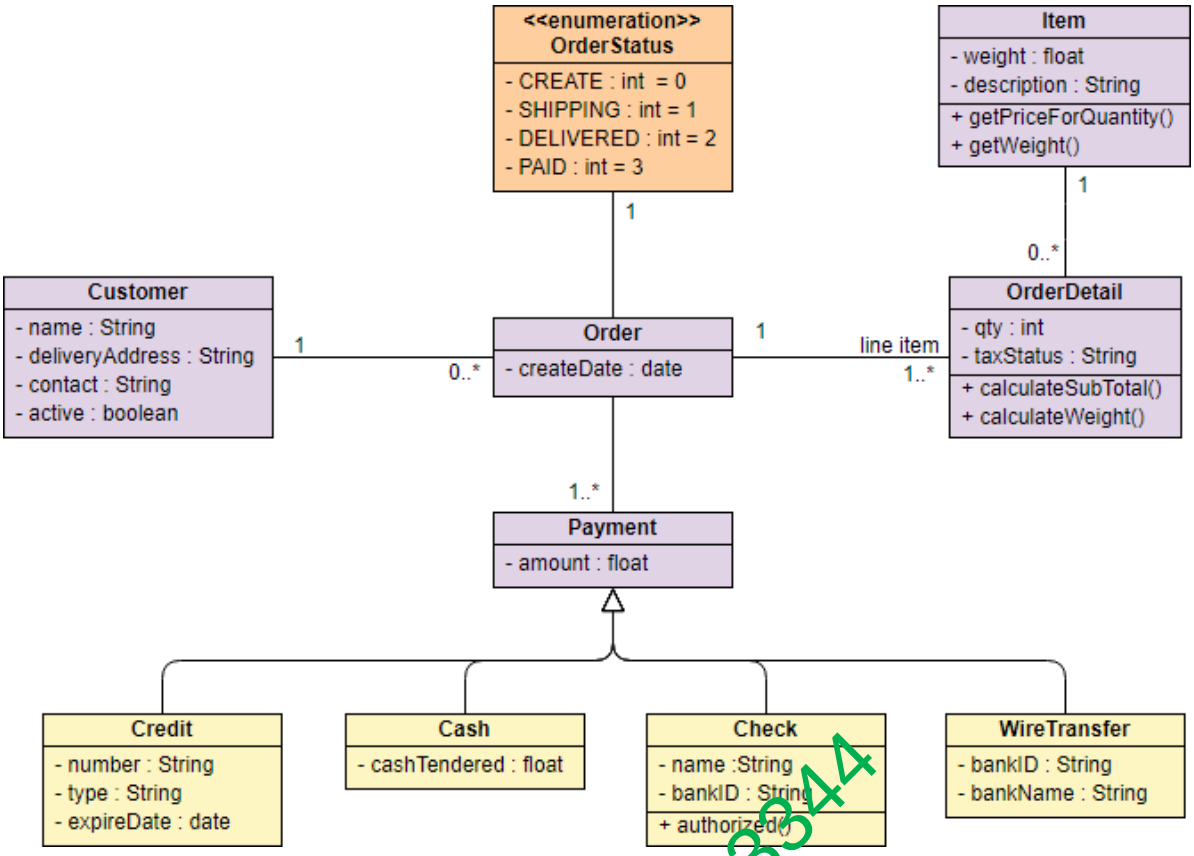


图 1：类图示例（图片来自 Visual Paradigm）

本节主要介绍了类图的基本概念和使用场景，介绍了类图的组件以及组件之间的关系并给出了相关范例。类图不仅是我们设计的工具，更是我们学习的工具，我们可以通过 IDEA 自带的类图工具来了解所要学习的源码核心列之间的关系，反向帮助我们理解源码。

下一节我们将讲述时序图的概念、使用场景和如何画时序图。

使用 IDEA 自带的类图工具作图，通过类图来了解 Java 的集合体系。

- 阿里巴巴与 Java 社区开发者.《Java 开发手册 1.5.0》华山版. 2019
- 维基百科 - 类图
- 类图的语法和功能
- 谭云杰.《大象：Thiking in UML》. 中国水利水电出版社. 2012
- 张传波.《火球：UML 大战需求分析》. 中国水利水电出版社. 2012

}