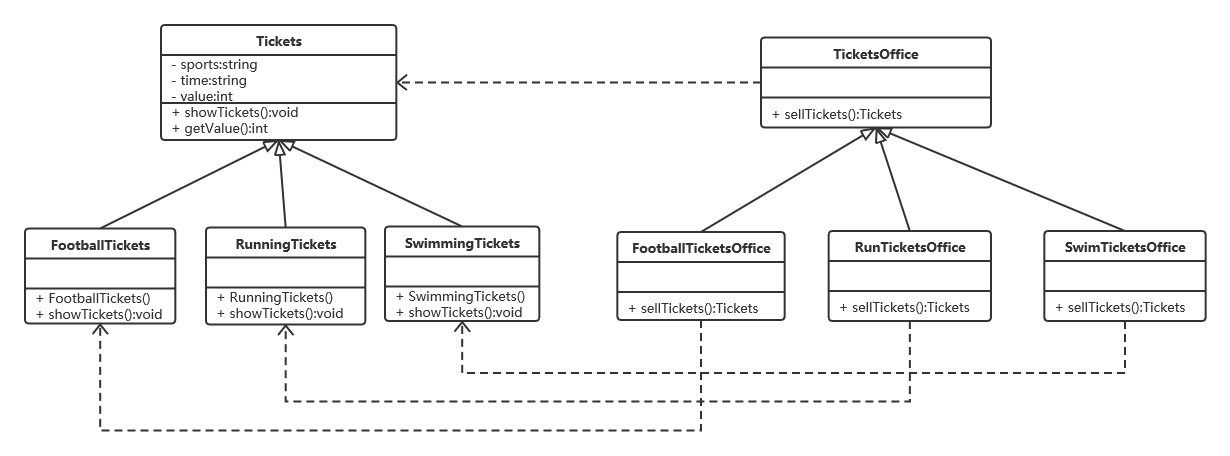
1. 设计模式FactoryMethod
   1. 实现API描述

如类图所示，在抽象类Tickets中，定义了一组私有属性（项目名称，开始时间和门票价值），还定义了获取门票价值的API以及展示门票信息的抽象函数，在继承它的三个实体子类中，除了继承的属性和方法外，每个类都定义了构造函数为私有成员赋值，并重写展示门票信息的API，因为不同种类门票的信息不同。在抽象工厂类TicketsOffice中，定义了抽象函数sellTickets,其返回类型为Tickets。在继承它的三个实体子类中，都对此方法进行重写，每个实体工厂类返回其对应生产的产品子类的实例。这样就实现工厂模式中要求的实例化过程交给子类实现，这种架构也降低了代码的耦合度。测试时只需创建工厂子类的实例然后调用其sellTickets()的API即可获取对应门票实例。

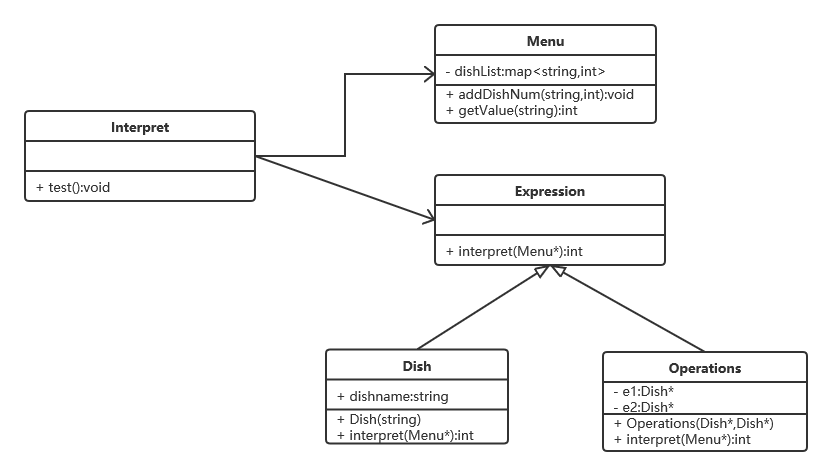
* 1. 类图



1. 设计模式Interpreter
   1. 实现API描述

如类图所示，在Menu中，定义私有成员dishList表示菜单，使用c++的map实现。然后定义了两个接口：addDishNum:向菜单中增加商品；getValue：根据商品名称获取金额。另外，定义一个抽象类Expression,其中定义了一个抽象函数interpret()。而具体的解释任务则由继承它的子类Dish和Operations实现。Dish类相当于解释器中专门处理终结符的类，定义了一个私有成员dishname记录商品名(相当于终结符)，重写了interpret接口将商品名转化成对应金额。Operations类相当于解释器中专门处理非终结符的类，在我们的例子中相当于把加法规则写到这个类中。其中我们重写了interpret接口，调用Dish的interpret接口获取金额并相加，将总金额返回。在类Interpreter中，定义了test()方法来进行测试，主要是将顾客选择购买的商品名传入Operations类的interpret()接口实现商品名称解释为对应金额。

* 1. 类图



1. 设计模式Objectpool
   1. 实现API描述

如类图所示，首先定义了FanCustome类记录服装编号，设计getNum() API来从外部获取服装编号。定义CustomeStatus类来记录服装状态（可借/已借出），定义setUsing(),setFree()接口来修改状态，getStatus()接口来获取状态。在此基础上定义CustomeManager类进行逻辑上的控制，定义私有成员pool作为对象池，默认对象池大小是5。定义三个私有方法，find()：遍历对象池，找到首个可用实例，修改其状态并返回指针；checkfree():统计可借服装数量；free():归还服装后放回对象池，修改其状态。在此基础上对应提供三个公有API进行外部访问。另外定义Client类作为外部使用类，提供applyCustome()接口负责借衣服并登记，returnCustome()接口负责还衣服并登记，show()接口负责显示对象池中每个对象的状态。

* 1. 类图

