本文档包括5个设计模式：建造者模式 Builder Pattern、组合模式 Composite Pattern、简单工厂模式 Simple Factory Factory、享元模式 Flyweight Pattern、单例模式 Singleton Pattern

1：建造者模式 Builder Pattern

1.1 API

1.1.1: class BulderOrder :

API: public Order createOrder(int[] number)

public void addRice(int number)

public void addPizza(int number)

public void addDrink(int number)

public void addNoodle(int number)

public void addDishes(int number)

public void addHumberger(int number)

描述:建造者Builder通过createOrder()来创建一个新的订单，调用方法addSth()来向订单增加内容。

1.1.2:interface class IBuilderOrder:

API: Order createOrder(int[] number)

描述:IBuilderOrder是BuilderOrder的接口类，为其提供创建订单的接口。

1.1.3:class Order:

API: public int getNumber()

public int getTotal\_price()

public Date getTime()

public void setNumber(int number)

public void setTime()

public void setTotal\_price()

public void display()

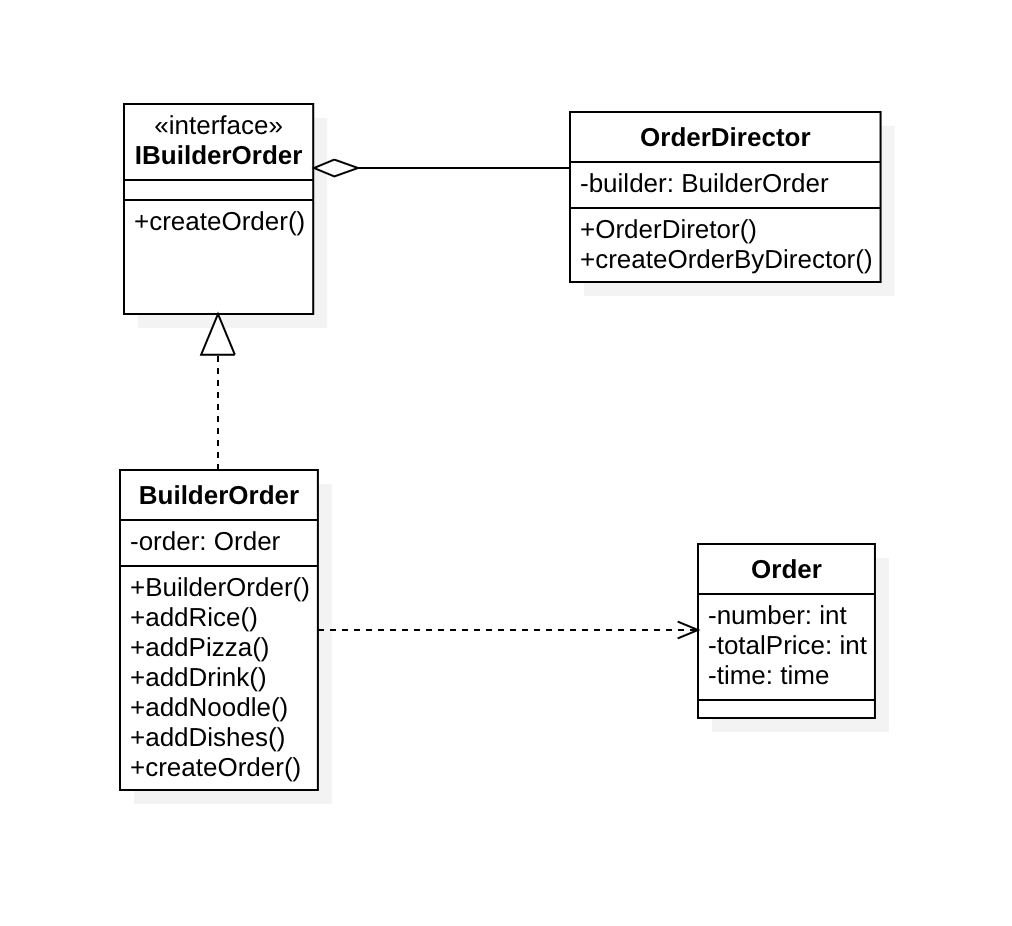
描述:产品类Order提供获得和设置订单号number、总价格total\_price、订单创建时间3个属性的接口，还有display()用来展示订单信息。

1.1.4: class OrderDirector:

API: public Order createOderByDirector(int[] number)

描述:Director指导建造者建造什么样的订单，传递一个数组作为订单的参数，直接返回一个对应的订单。

1.2 类图



2: 组合模式 Composite Pattern

2.1 API

2.1.1:class Menu

API: public String getName()

public void setName(String name)

public void addFood(Food f)

public void updateMenu()

public void display()

描述:菜单类作为组合中合的部分 可以对菜单名进行操作，获取或修改菜单名，调用addFood(Food f)方法向菜单中添加新的食物，并且会自动调用updateMenu()

方法将新食物自动添加到分菜单中，display()将当前菜单格式化输出。

2.1.2 class MenuPart

API: public String getName()

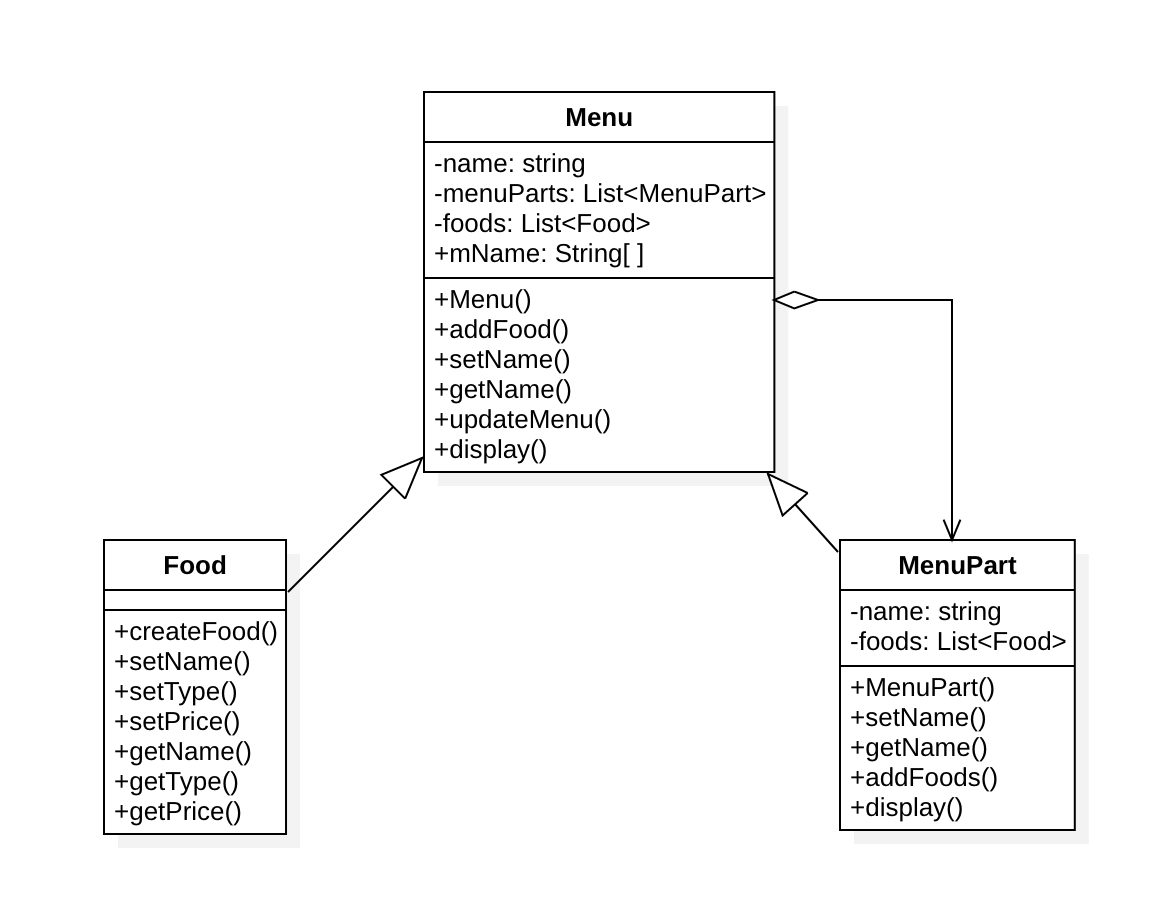
public void setName(String name)

public void addFoods(Food f)

public void display()

描述: class MenuPart是对class Menu的继承，方法和用法基本相同。

2.2 类图



3.

3.1 API

3.1.1 class SimpleFoodFactory

API: public Food createFood(String name,int price,String type)

描述:食物工厂提供了一个创造食物的接口 createFood()，三个参数确定了食物的属性。

3.1.2 class Food

API: public String getName()

public int getPrice()

public String getType()

public void setName(String name)

public void setPrice(int price)

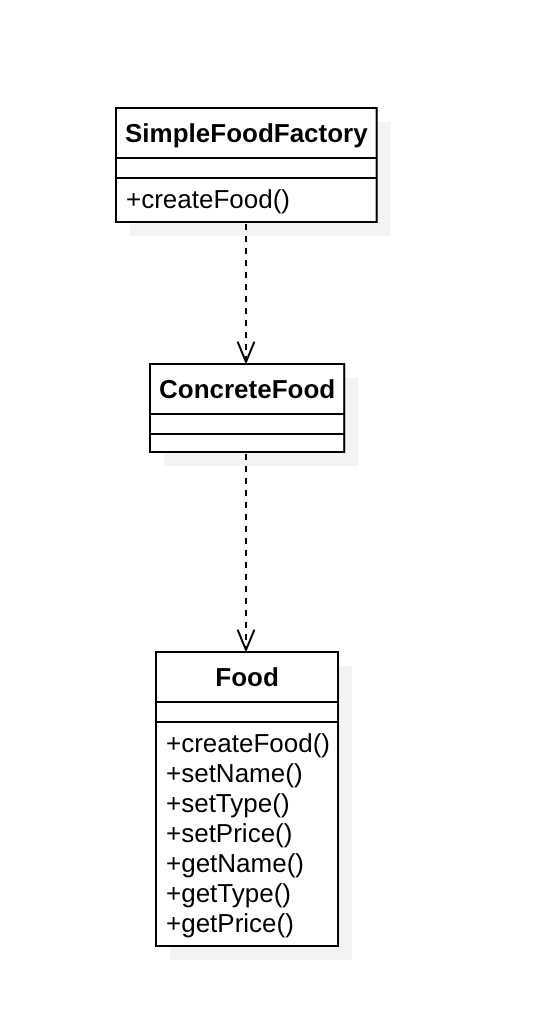
public void setType(String type)

public void display()

public Food createFood(String name,int price,String type)

描述:这是一个抽象类Food，不存储任何有效信息，定义了一个实体类应有的接口。

3.2 类图



4. 享元模式 Flyweight Pattern

4.1 API

4.1.1 class FlyweightOrder

API: public int[] getOrderInf(int number)

描述:这里提供了一个获取享元池中信息（套餐内容）的接口，输入套餐编号number，返回套餐内食物的参数，以用来创建订单。

4.1.2 class ConcreteFlyweightOrder

API: public int[] getOrderInf(int number)

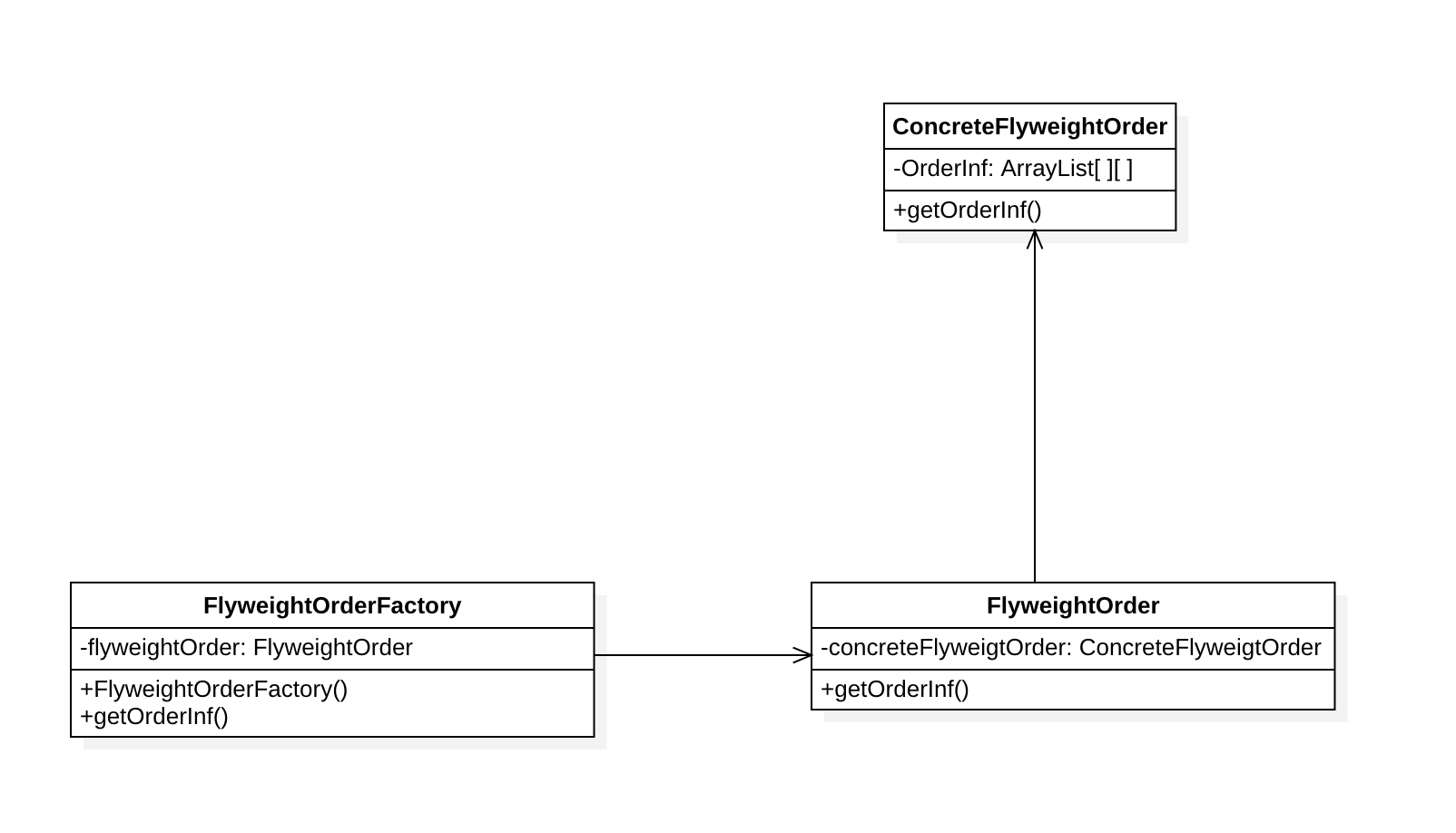
描述:这个方法继承自FlyweightOrder，但ConcreteFlyweightOrder存储了具体的套餐信息，根据判断条件返回对应的参数。

4.1.3 class FlyweightOrderFactory

API: public int[] getOrderInf(int number)

描述:提供外界访问内部套餐数据的接口，在接受到请求后访问ConrcreteFlyweightOrder类来获取信息。

4.2 类图



5.单例模式 Singleton Pattern

5.1 API

5.1.1: class OrderDirector:

API: public Order createOderByDirector(int[] number)

描述:本项目中的建造者模式中的Director可以设计为单例模式。一般只存在一个Director即可完成全部对建造者的指导工作（如收银员，收银系统）。Director指导建造者建造什么样的订单，传递一个数组作为订单的参数，直接返回一个对应的订单。

5.1.2: class SimpleFoodFactory

API: public Food createFood(String name,int price,String type)

描述:本项目中简单工厂模式中的SimpleFoodFactory可以设计为单例模式。一般只存在一个即可满足所有建造食物的需求。食物工厂提供了一个创造食物的接口 createFood()，三个参数确定了食物的属性。

5.1.3:class FlyweightOrderFactory

API: public int[] getOrderInf(int number)

描述:本项目中享元模式中的FlyweightOrderFactory可以设计为单例模式。一般只存在一个套餐享元工厂即可满足所有获取套餐信息的需求。FlyweightOrderFactory提供外界访问内部套餐数据的接口，在接受到请求后访问ConrcreteFlyweightOrder类来获取信息。

5.2 类图

各单例类图详见上述各设计模式中的类图部分。