Task 2 Group 5A

The team implemented 2 design patterns, *Strategy* and *Builder*, which are present in Coupon and Pizza implementations, respectively. Below are in-depth explanations for both cases.

Coupon (Strategy)

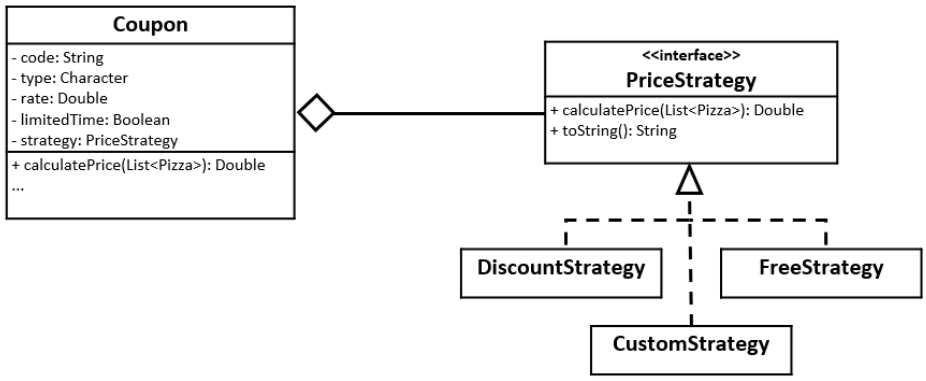
Anni's Pizza provides users with coupons. There are different types of coupons, namely discount, buy-one-get-one-free, and custom coupon (which is to be implemented in the future). When a coupon is applied, the new price of the basket is calculated, but the method to calculate the price differs between the coupon types.

We thought that using if-else blocks to choose the correct method according to the coupon type is not ideal, because there would be too many lines that will be unused. In order to implement the price calculation methods coherently, we decided to use a *Strategy* design pattern. The coupon class having the strategy to calculate the price as an attribute makes it easy to call the correct method.

The *PriceStrategy* interface has the **calculatePrice()** method which returns the correct price of the basket - as a double - after applying the coupon. This interface is implemented by three different classes (*DiscountStrategy*, *FreeStrategy*, and *CustomStrategy*) which contain the correct methods to calculate the price for three different coupon types.

When a coupon is created, the correct *Strategy* is created by looking at the type attribute (and the rate attribute, if the coupon is a discount coupon. The **calculatePrice()** method of the coupon's *Strategy* is called to calculate the total price of the basket when the coupon is applied, or when the basket is edited.

Below is the class diagram displaying how the pattern is structured in your code:



The *PriceStrategy* interface and the *Strategy* classes are implemented in the commons module (/lab-template/commons/src/main/commons/strategies). The *Coupon* class, which is also in the commons module, is responsible for creating these classes.

The **calculatePrice()** method for the strategy is used in the *BasketService* class, which is in /lab-template/basket-microservice/src/main/java/nl/tudelft/sem/template/basket/services. The methods **applyCoupon()** and **calculatePrice()** call the method in PriceStrategy in order to retrieve the correct price when the coupon is applied.

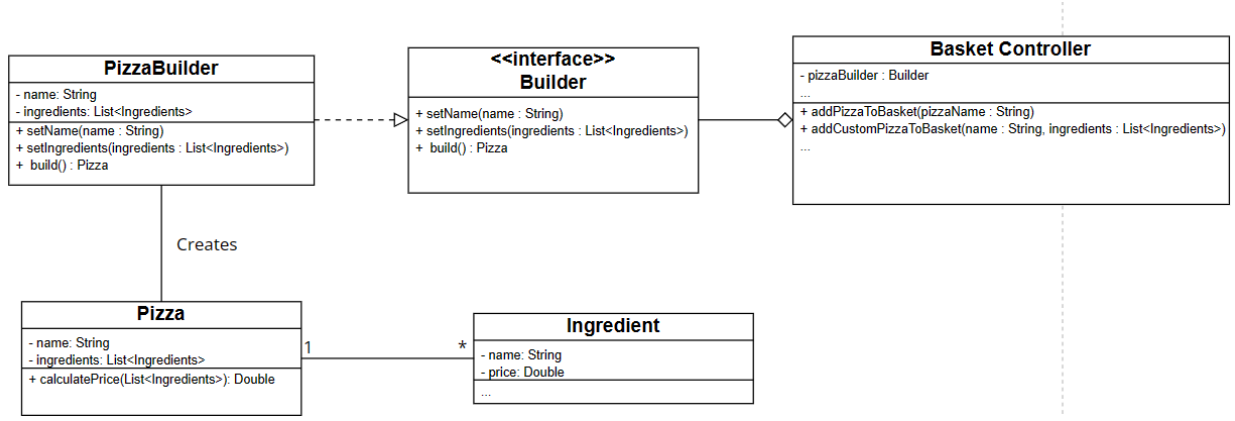
Pizza(Builder)

When a customer orders from Anni's Pizza, they are presented with a pizza menu with a plethora of prebuilt pizzas. In addition, in the scenario description Anni's pizza also clearly described that the customers, if they are not interested in the default pizzas, can also make one themselves using the available ingredients.

Because of this, we need a flexible way of constructing a complex object (the *Pizza* Class), that allows us to create both default Pizzas and custom pizzas that can have any possible combination of toppings. Thus, we chose to implement this functionality using the *Builder* design pattern. This makes our implementation have an easy construction procedure that allows different representations of pizzas.

The *Pizza Builder* class inherits from the Builder interface. This builder implements a setter for every pizza parameter: *name* and *ingredients*. Finally, the **build()** method outputs a pizza with the previously described parameters. This pizza builder is used in our basket controller to create both types of pizzas when a customer wants to add either of the pizza types to their basket.

Below is the class diagram displaying how the pattern is structured in your code:



The *Pizza* and *Ingredients* classes are in the */commons* module, and they can be accessed from the root path with the following path: /lab-template/commons.

The *Builder* is implemented in the basket microservice, found in lab-template/basket-microservice. Inside the builder package (found in /basket-microservice/src/main/java/nl/tudelft/sem/template/basket/builder) we have implemented the *Builder* interface as well as the *PizzaBuilder* class.

In /basket-microservice/src/main/java/nl/tudelft/sem/template/basket/controllers you can find the *BasketController,* which uses the pizza builder to build both custom and default pizzas to be added to the customers' basket.