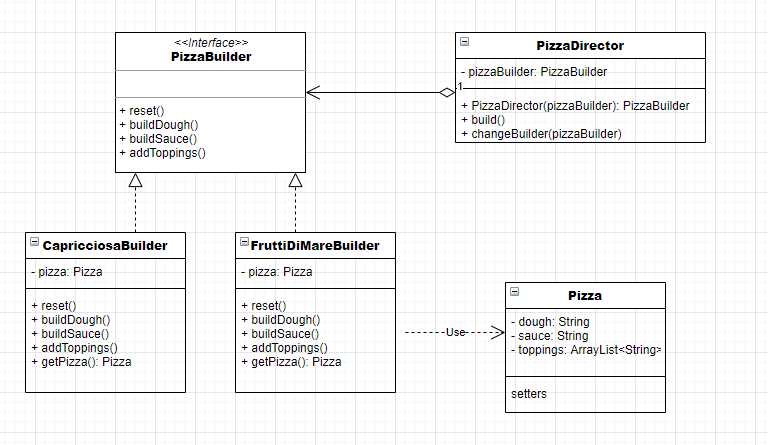
# Design Patterns

## Builder



Director

public class PizzaDirector {  
 private PizzaBuilder pizzaBuilder;  
  
 public PizzaDirector(PizzaBuilder pizzaBuilder) {  
 this.pizzaBuilder = pizzaBuilder;  
 }  
  
 public void build() {  
 pizzaBuilder.reset();  
 pizzaBuilder.buildDough();  
 pizzaBuilder.buildSauce();  
 pizzaBuilder.addToppings();  
 }  
  
 public void changeBuilder(PizzaBuilder pizzaBuilder) {  
 this.pizzaBuilder = pizzaBuilder;  
 }  
}

Builder interface

public interface PizzaBuilder {  
 void reset();  
  
 void buildDough();  
  
 void buildSauce();  
  
 void addToppings();  
}

Capricciosa Builder

public class CapricciosaBuilder implements PizzaBuilder {  
 private Pizza pizza;  
  
 public void reset() {  
 this.pizza = new Pizza();  
 }  
  
 public void buildDough() {  
 this.pizza.setDough("thick rust");  
 }  
  
 public void buildSauce() {  
 this.pizza.setSauce("tomato sauce");  
 }  
  
 public void addToppings() {  
 this.pizza.setToppings(Arrays.*asList*("sirloin", "black olives", "mushrooms", "oregano"));  
 }  
  
 public Pizza getPizza() {  
 return pizza;  
 }  
}

Pizza

public class Pizza {  
 private String dough;  
 private String sauce;  
 private List<String> toppings;  
  
 public void setDough(String dough) {  
 this.dough = dough;  
 }  
  
 public void setSauce(String sauce) {  
 this.sauce = sauce;  
 }  
  
 public void setToppings(List<String> toppings) {  
 this.toppings = toppings;  
 }  
  
 @Override  
 public String toString() {  
 return "Pizza{" +  
 "dough='" + dough + '\'' +  
 ", sauce='" + sauce + '\'' +  
 ", toppings=" + toppings +  
 '}';  
 }  
}

Główna metoda main

public static void main(String[] args) {  
 CapricciosaBuilder capricciosaBuilder =new CapricciosaBuilder();  
 PizzaDirector pizzaDirector = new PizzaDirector(capricciosaBuilder);  
 pizzaDirector.build();  
 System.*out*.println(capricciosaBuilder.getPizza());  
}