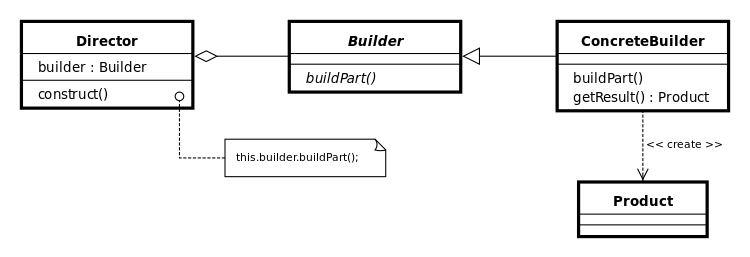
Separates object construction from its representation

Builder pattern builds a complex object using simple objects and using a step by step approach. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

A Builder class builds the final object step by step. This builder is independent of other objects.



Product : pizza

Builder : pizza builder

Director : waiter

/\* product \*/

class Pizza{

private String dough = "";

private String sauce = "";

private String topping = "";

public void setDough(String dough) {

this.dough = dough;

}

public void setSauce(String sauce) {

this.sauce = sauce;

}

public void setTopping(String topping) {

this.topping = topping;

}

@Override

public String toString() {

return this.dough+":"+this.sauce+":"+this.topping;

}

}

/\* abstract builder \*/

abstract class PizzaBuilder{

protected Pizza pizza;

public Pizza getPizza() {

return pizza;

}

public void createNewPizzaProduct() {

pizza = new Pizza();

}

public abstract void buildDough();

public abstract void buildSauce();

public abstract void buildTopping();

}

/\* concrete builder \*/

class HawaiianPizzaBuilder extends PizzaBuilder{

@Override

public void buildDough() {

pizza.setDough("cross");

}

@Override

public void buildSauce() {

pizza.setSauce("mild");

}

@Override

public void buildTopping() {

pizza.setTopping("ham+pineapple");

}

}

/\* concrete builder \*/

class SpicyPizzaBuilder extends PizzaBuilder{

@Override

public void buildDough() {

pizza.setDough("pan baked");

}

@Override

public void buildSauce() {

pizza.setSauce("hot");

}

@Override

public void buildTopping() {

pizza.setTopping("pepperoni+salami");

}

}

/\* director \*/

class Waiter{

private PizzaBuilder pizzaBuilder;

public void setPizzaBuilder(PizzaBuilder pb) {

pizzaBuilder = pb;

}

public Pizza getPizza() {

return pizzaBuilder.getPizza();

}

public void constructPizza() {

pizzaBuilder.createNewPizzaProduct();

pizzaBuilder.buildDough();

pizzaBuilder.buildSauce();

pizzaBuilder.buildTopping();

}

}

public class PizzaDemoBuilder {

public static void main(String[] args) {

Waiter waiter = new Waiter();

PizzaBuilder hawaiianPizzabuilder = new HawaiianPizzaBuilder();

PizzaBuilder spicyPizzaBuilder = new SpicyPizzaBuilder();

waiter.setPizzaBuilder(hawaiianPizzabuilder);

waiter.constructPizza();

Pizza pizza = waiter.getPizza();

System.out.println(pizza.toString());//object constructed

}

}