The command pattern is a pattern which is like a special menu at a magical restaurant. Each item on the menu is like a magical spell which will tell the chef what to cook. You or me as the customer can order by choosing the spells from the menu, and then the chefs will follow the instructions in the spell to make your delicious dishes.

```
// Command interface
interface Command {
    void execute();
}
```

That's the command interface. This interface declares the execute method which the concrete classes will implement.

```
class OrderMeatPizza implements Command {
    private PizzaKitchen pizzaKitchen;

    public OrderMeatPizza(PizzaKitchen pizzaKitchen) {
        this.pizzaKitchen = pizzaKitchen;
    }

    @Override
    public void execute() {
        pizzaKitchen.makeChickenPizza();
    }
}

class OrderVeggiePizza implements Command {
    private PizzaKitchen pizzaKitchen;

    public OrderVeggiePizza(PizzaKitchen pizzaKitchen) {
        this.pizzaKitchen = pizzaKitchen;
    }

    @Override
    public void execute() {
        pizzaKitchen.makeVeggiePizza();
    }
}
```

Here I have created two command concrete classes OrderMeatPizza and OrderVeggiePizza. These classes implement the Command interface. Both classes have a reference to the pizzaKitchen and the execute method invokes specific methods on the pizza kitchen to make either a chicken or veggie pizza.

```
class PizzaKitchen {
    public void makeChickenPizza() {
        System.out.println("Making Chicken Pizza");
    }

    public void makeVeggiePizza() {
        System.out.println("Making Veggie Pizza");
    }
}
```

That's the Receiver class. This class name is PizzaKitchen. In this class I have added the makeChickenPizza and makeVeggiePizza method. This class will represent the actions which will be performed.

```
class Customer {
    private Command command;

    public void setOrder(Command command) {
        this.command = command;
    }

    public void placeOrder() {
        command.execute();
    }
}
```

This is the invoker class. This class name is Costumer. This class acts as an Invoker that triggers the execution of a command. I have also provided the reference to the command and I have also provided a method to execute that command.

There are many benefits of using the command Pattern. The command Pattern decouples the customer(sender) from the Pizza Kitchen(receiver). This separation is beneficial because I can change one part of the system without affecting the other. If I would not have used the command pattern then the customer class would need direct knowledge of the methods in PizzaKitchen. This tight coupling could make it harder to maintain the system because changes in one system could lead to that i need to change something in the other system too. Adding new commands like ordering a different type of pizza can be done without any headache without modifying the existence code. This flexibility makes it easier to add new features to this program. WIthout the command pattern adding new operations or changing existing ones in the PizzaKitchen would you also require modification in the costumer class. This lack of flexibility could lead to code which is harder to maintain.