T-302-HONN: Lab 4
Design Patterns

Þórður Friðriksson
thordurf@ru.is

Háskólinn í Reykjavík — Reykjavík University

Introduction

Attention Please read over all the following points before you start the project

1. **Groups**
   - This assignment can be done in groups of 1-2 people
   - Only one member of each group needs to hand in the assignment
   - Remember to add each group members name and email to the front page of the handed-in assignment
2. **Rules about cheating**
   - The solution needs to be you own work, if it is discovered that cheating has occured then all participating parties will receive a 0 for this assignment if this is their first offence on the other hand if this is a repeating occurance then more severe measures can be expected. See the school's policy and rules on assignments ru.is/namid/reglur/reglur-um-verkefnavinnu
3. **Assignment hand-in**
   - the assignments need to be handed-in on Canvas both for the pdf file with all the written answers as well as a .zip file for the code
   - pdf file must be named: `{student1@ru.is}-{student2@ru.is}-lab4.pdf`
   - zip file must be named: `{student1@ru.is}-{student2@ru.is}-lab4.zip`
   - **If the instructions for the handin are not follwed you can expect a grade deduction**
4. **Late submissions**
   - See late submission policy

Notice.

- Skeleton code made with Python 3.9

1 The Factory Pattern (50 points)

1.1 (Total points: 30) Pizza Store python implementation

Implement the last version of the `PizzaStore` application in the Factory chapter in the Head First Design Patterns book, that is to say with the `IngredientFactory` and with the `Template Method`, the skeleton for this implementation can be found in `factory_skeleton`

Notice.

- Hand-in your code in the zip file under a folder named factory

On on the next page you can see the expected output that you will get for the correct implementation when the main file, which you can find in the skeleton code, is run.

Command Line

```
$ python main.py
--- New York Style Pizzas ---


preparing: New York Style Cheese Pizza
Bake for 25 minutes at 350
Cutting the pizza into diagonal slices
Place pizza in official PizzaStore box

            Name: New York Style Cheese Pizza,
            Dough: ThinCrustDough,
            Sauce: MarinaraSauce,
            Veggies: [],
            Cheese: ReggianoCheese,
            Pepperoni: NoneType,
            Clams: NoneType

---

preparing: New York Style Pepperoni Pizza
Bake for 25 minutes at 350
Cutting the pizza into diagonal slices
Place pizza in official PizzaStore box

            Name: New York Style Pepperoni Pizza,
            Dough: ThinCrustDough,
            Sauce: MarinaraSauce,
            Veggies: [],
            Cheese: ReggianoCheese,
            Pepperoni: SlicedPepperoni,
            Clams: NoneType

---

preparing: New York Style Clam Pizza
Bake for 25 minutes at 350
Cutting the pizza into diagonal slices
Place pizza in official PizzaStore box

            Name: New York Style Clam Pizza,
            Dough: ThinCrustDough,
            Sauce: MarinaraSauce,
            Veggies: [],
            Cheese: ReggianoCheese,
            Pepperoni: NoneType,
            Clams: FreshClams

---
```

Command Line

```
preparing: New York Style Veggie Pizza
Bake for 25 minutes at 350
Cutting the pizza into diagonal slices
Place pizza in official PizzaStore box

            Name: New York Style Veggie Pizza,
            Dough: ThinCrustDough,
            Sauce: MarinaraSauce,
            Veggies: ['Garlic', 'Onion', 'Mushroom', 'RedPepper'],
            Cheese: ReggianoCheese,
            Pepperoni: NoneType,
            Clams: NoneType

---



--- Chicago Style Pizzas ---


preparing: Chicago Style Cheese Pizza
Bake for 25 minutes at 350
Cutting the pizza into diagonal slices
Place pizza in official PizzaStore box

            Name: Chicago Style Cheese Pizza,
            Dough: ThickCrustDough,
            Sauce: PlumTomatoSauce,
            Veggies: [],
            Cheese: Mozzarella,
            Pepperoni: NoneType,
            Clams: NoneType

---

preparing: Chicago Style Pepperoni Pizza
Bake for 25 minutes at 350
Cutting the pizza into diagonal slices
Place pizza in official PizzaStore box

            Name: Chicago Style Pepperoni Pizza,
            Dough: ThickCrustDough,
            Sauce: PlumTomatoSauce,
            Veggies: [],
            Cheese: Mozzarella,
            Pepperoni: SlicedPepperoni,
            Clams: NoneType

---
```

Command Line

```
preparing: Chicago Style Clam Pizza
Bake for 25 minutes at 350
Cutting the pizza into diagonal slices
Place pizza in official PizzaStore box

            Name: Chicago Style Clam Pizza,
            Dough: ThickCrustDough,
            Sauce: PlumTomatoSauce,
            Veggies: [],
            Cheese: Mozzarella,
            Pepperoni: NoneType,
            Clams: FrozenClams

---

preparing: Chicago Style Veggie Pizza
Bake for 25 minutes at 350
Cutting the pizza into diagonal slices
Place pizza in official PizzaStore box

            Name: Chicago Style Veggie Pizza,
            Dough: ThickCrustDough,
            Sauce: PlumTomatoSauce,
            Veggies: ['EggPlant', 'BlackOlives', 'Spinach'],
            Cheese: Mozzarella,
            Pepperoni: NoneType,
            Clams: NoneType

---
```

1.2 (Total points 10) Problems with the PizzaStore

There are a couple of things wrong with the PizzaStore implementation that could be done better that were done for simplicities sake and to more easily show the Factory Pattern

What it wron g and how could you fix it ?

1.3 (Total points 10) Template Method

In this example / solution you can also find the `Template Method` pattern, explain where you can find that pattern and how it is the `Template Method` pattern.

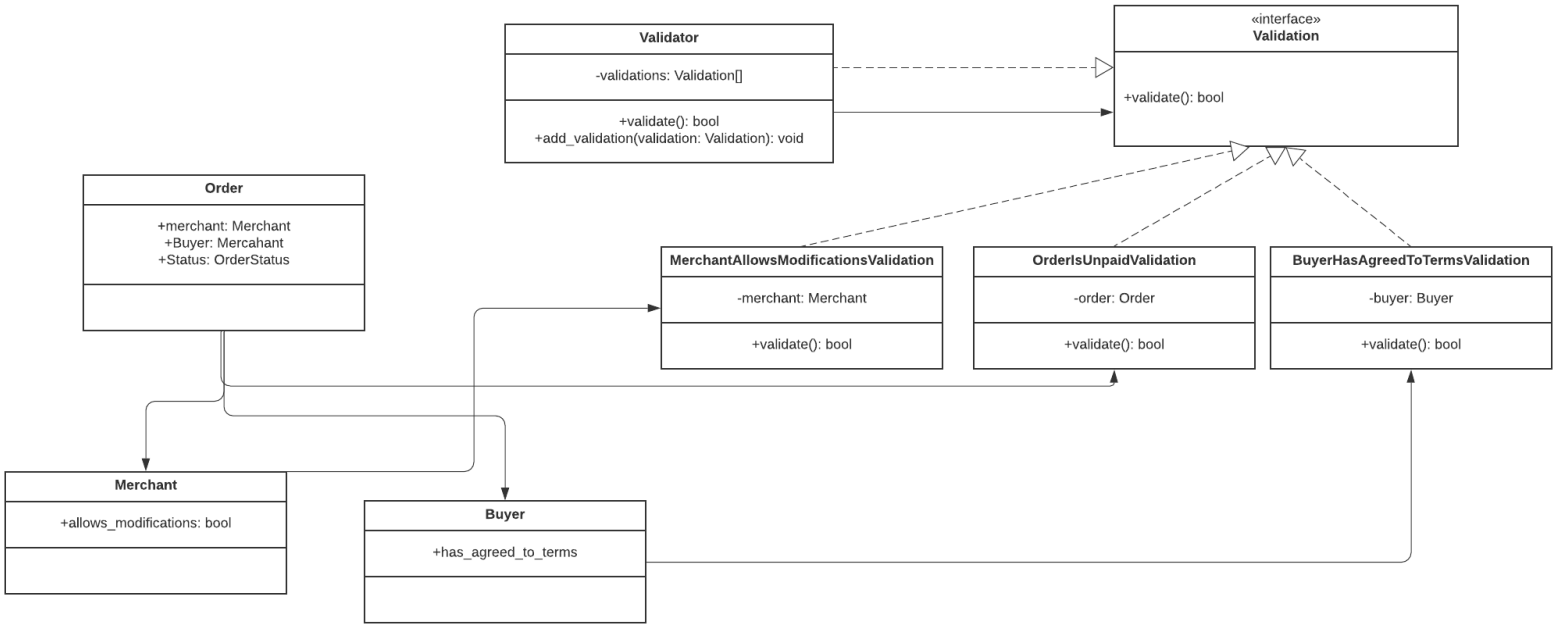
2 The Command Pattern (50 points)

2.1 (Total points: 10) Validation

On the diagram below you can see somewhat of a framework for validations, explain how the diagram below is in fact the command pattern

Info:

- Keep in mind that Design Patterns in the real world often don't take the exact same image as textbook design patterns, instead they often come forth as Design Patterns in disguise. That is to say that they may not look exactly like the class diagrams of Design Patterns in textbooks but they represent the same idea.
- Also keep in mind that this example, like many other practice examples, are overkill and if this were a real application we would probably not want to implement it like this, instead we would probably just have the validation logic inside a function inside some OrderModificationValidator class, but this validation framework could become useful when you have a big applications with many validators with complicated validation logic that is duplicated in many places (but even then we would probably just use some pre-existing framework).
- Also it is worth mentioning that the `Order`, `Merchant` and `Buyer` are all very shallow data classes for the sake of simplicity.



2.2 (Total points: 25 points) Validation Implementation

Implement the diagram above, all you need to get started can be found in the `command_skeleton`. The expected output when the main function is run with the correct implementation is:

Command Line

```
$ python main.py
False
False
False
True
```

2.3 (Total points: 7.5) Simple Factory

There is a place in the skeleton that would be good to replace with a Simple Factory, where is that place and what object creation should we be encapsulating? Be clear in exactly what code we would encapsulate.

2.4 (Total points: 7.5) A Second Pattern

In this example there is another pattern that isn't the *Command* pattern, name that pattern and explain.

Notice. I am not looking for the Strategy Pattern