

SWE-4501: Design Pattern



Factory Design Pattern

Md. Nazmul Haque

Lecturer, IUT

Department of Computer Science and Engineering
Islamic University of Technology

July 15, 2021

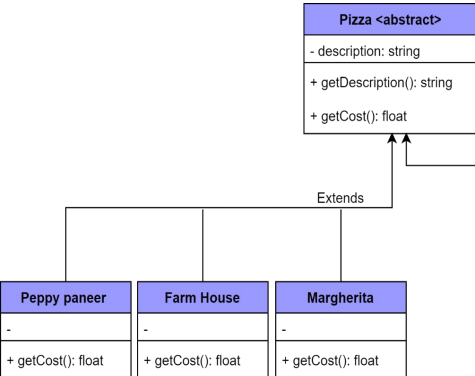
Contents



- Motivation
- Solution



Pizza decoration



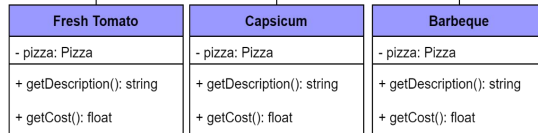
```
Pizza pizza;
```

```
i. pizza = new Margherita();
```

```
ii. Type check
```

```

if(type=="Margherita"){
    pizza = new Margherita();
}
else if(type == "FarmHouse"){
    pizza = new FarmHouse();
}
  
```





Pizza decoration

```
Pizza orderPizza() {

    Pizza pizza = new Pizza();

    pizza.prepare();
    pizza.bake();
    pizza.cut();
    pizza.box();
    return pizza;
}
```

We can't do that
as it is an
abstract class
or an interface.

```
Pizza orderPizza(String type) {
    Pizza pizza;

    if (type.equals("Margherita")) {
        pizza = new Margherita();
    } else if (type.equals("FarmHouse")) {
        pizza = new FarmHouse();
    } else if (type.equals("PeppyPaneer")) {
        pizza = new PeppyPaneer();
    }

    pizza.prepare();
    pizza.bake();
    pizza.cut();
    pizza.box();
    return pizza;
}
```

```
Pizza orderPizza(String type) {
    Pizza pizza;

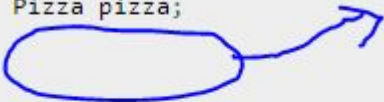
    if (type.equals("Margherita")) {
        pizza = new Margherita();
    } else if (type.equals("FarmHouse")) {
        pizza = new FarmHouse();
    } else if (type.equals("PeppyPaneer")) {
        pizza = new PeppyPaneer();
    } else if (type.equals("ChickenFiesta")) {
        pizza = new ChickenFiesta();
    }

    pizza.prepare();
    pizza.bake();
    pizza.cut();
    pizza.box();
    return pizza;
}
```



Pizza decoration

```
Pizza orderPizza(String type) {  
  
    Pizza pizza;  
  
    pizza.prepare();  
    pizza.bake();  
    pizza.cut();  
    pizza.box();  
  
    return pizza;  
}
```



```
public class SimplePizzaFactory {  
  
    public Pizza createPizza(String type) {  
        Pizza pizza = null;  
  
        if (type.equals("Margherita")) {  
            pizza = new Margherita();  
        } else if (type.equals("FarmHouse")) {  
            pizza = new FarmHouse();  
        } else if (type.equals("PeppyPaneer")) {  
            pizza = new PeppyPaneer();  
        }  
        return pizza;  
    }  
}
```



Pizza decoration

```
public class PizzaStore {
    SimplePizzaFactory factory;

    public PizzaStore(SimplePizzaFactory factory) {
        this.factory = factory;
    }

    Pizza orderPizza(String type) {
        Pizza pizza;

        pizza = factory.createPizza(type);

        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();
        return pizza;
    }
}
```

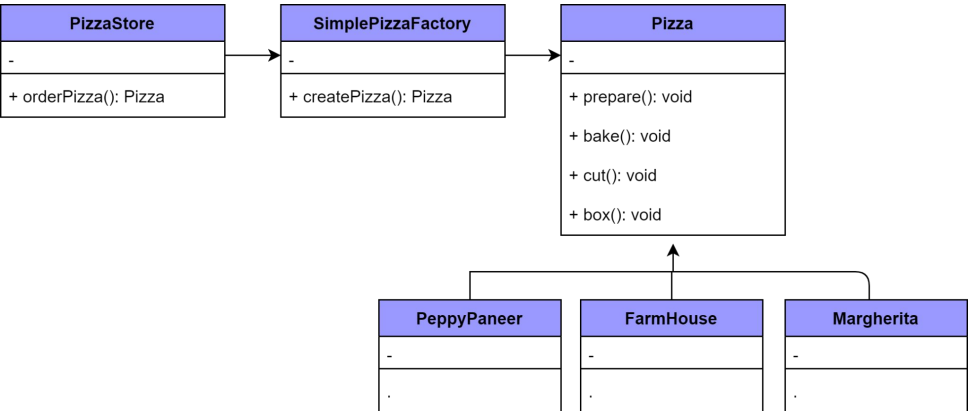
```
public class SimplePizzaFactory {

    public Pizza createPizza(String type) {
        Pizza pizza = null;

        if (type.equals("Margherita")) {
            pizza = new Margherita();
        } else if (type.equals("FarmHouse")) {
            pizza = new FarmHouse();
        } else if (type.equals("PeppyPaneer")) {
            pizza = new PeppyPaneer();
        }
        return pizza;
    }
}
```



Design a Simple Factory





Pizza decoration

```
public class PizzaStore {
    SimplePizzaFactory factory;

    public PizzaStore(SimplePizzaFactory factory) {
        this.factory = factory;
    }

    Pizza orderPizza(String type) {
        Pizza pizza;

        pizza = factory.createPizza(type);

        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();
        return pizza;
    }
}
```

```
public class SimplePizzaFactory {

    public Pizza createPizza(String type) {
        Pizza pizza = null;

        DhakaPizzaFactory

        pizza = new Margherita();
    } else if (type.equals("FarmHouse")) {
        ChittagongPizzaFactory

    }
    return pizza;
}
```

RajshahiPizzaFactory



Pizza decoration

```
DhakaPizzaFactory dhFactory = new DhakaPizzaFactory();  
PizzaStore dhStore = new PizzaStore(dhFactory);  
dhStore.order(type); // Margherita FarmHouse PeppyPaneer  
  
ChittagongPizzaFactory chittagongFactory = new ChittagongPizzaFactory();  
PizzaStore chittagongStore = new PizzaStore(chittagongFactory);  
chittagongStore.order(type); // Margherita FarmHouse PeppyPaneer
```



Pizza decoration

```
public class PizzaStore {
    SimplePizzaFactory factory;

    public PizzaStore(SimplePizzaFactory factory) {
        this.factory = factory;
    }

    Pizza orderPizza(String type) {
        Pizza pizza;

        pizza = factory.createPizza(type);

        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();
        return pizza;
    }
}
```

```
public abstract class PizzaStore {

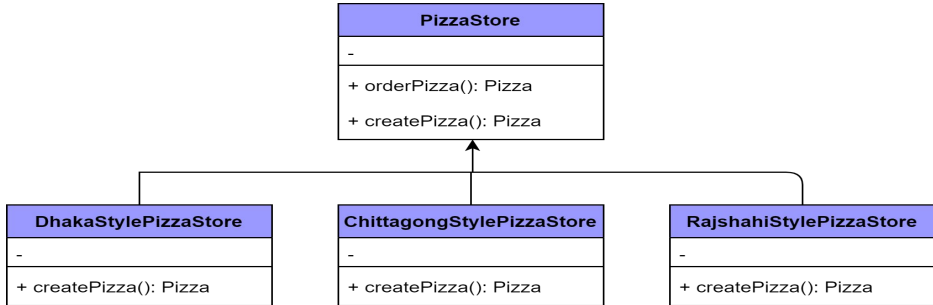
    public Pizza orderPizza(String type) {
        Pizza pizza;

        pizza = createPizza(type);

        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();
        return pizza;
    }
    abstract Pizza createPizza(String type);
}
```



Pizza decoration



```

public Pizza createPizza(type) {
    Pizza pizza;
    if (type.equals("Margherita")) {
        pizza = new DhakaStyleMargherita();
    } else if (type.equals("FarmHouse")) {
        pizza = new DhakaStyleFarmHouse();
    } else if (type.equals("PeppyPaneer")) {
        pizza = new DhakaStylePeppyPaneer();
    }
    return pizza;
}

```

```

public Pizza createPizza(type) {
    Pizza pizza;
    if (type.equals("Margherita")) {
        pizza = new RajshahiStyleMargherita();
    } else if (type.equals("FarmHouse")) {
        pizza = new RajshahiStyleFarmHouse();
    } else if (type.equals("PeppyPaneer")) {
        pizza = new RajshahiStylePeppyPaneer();
    }
    return pizza;
}

```



Pizza Factory

```
public abstract class PizzaStore {  
  
    public Pizza orderPizza(String type) {  
        Pizza pizza;  
  
        pizza = createPizza(type);  
  
        pizza.prepare();  
        pizza.bake();  
        pizza.cut();  
        pizza.box();  
        return pizza;  
    }  
    abstract Pizza createPizza(String type);  
}
```

abstract Product factoryMethod(String type)



Pizza Factory

Customer: A (DH)

```
PizzaStore dhStore = new DHPizzaStore();
```

```
dhStore.orderPizza("PeppyPaneer");
```

```
pizza = createPizza("PeppyPaneer");
```

```
pizza.prepare()  
pizza.bake()  
pizza.cut()  
pizza.box()
```

Customer: B (CH)

```
PizzaStore chStore = new CHPizzaStore();
```

```
chStore.orderPizza("FarmHouse");
```

```
pizza = createPizza("FarmHouse");
```



Pizza Class

```
public abstract class Pizza {
    String name;
    String dough;
    String sauce;
    ArrayList toppings = new ArrayList();

    void prepare() {
        System.out.println("Preparing " + name);
        System.out.println("Tossing dough...");
        System.out.println("Adding sauce...");
        System.out.println("Adding toppings: ");

        for (int i = 0; i < toppings.size(); i++) {
            System.out.println(" " + toppings.get(i));
        }
    }

    void bake() {
        System.out.println("Bake for 25 minutes at 350");
    }

    void cut() {
        System.out.println("Cutting the pizza into diagonal slices");
    }

    void box() {
        System.out.println("Place pizza in official PizzaStore box");
    }

    public String getName() {
        return name;
    }
}
```



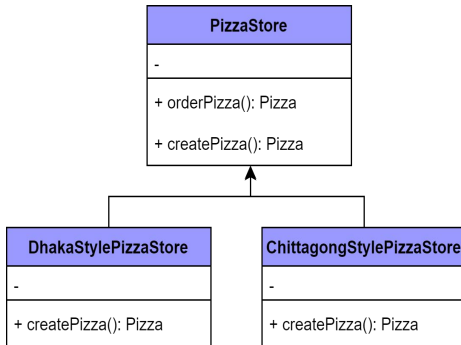
Test Pizza Factory

```
public class PizzaTestDrive {  
    public static void main(String[] args) {  
  
        PizzaStore dhStore = new DHPizzaStore();  
        Pizza pizza = dhStore.orderPizza("PeppyPaneer");  
        System.out.println("Customer A ordered a " + pizza.getName());  
  
        PizzaStore chStore = new CHPizzaStore();  
        pizza = chStore.orderPizza("FarmHouse");  
        System.out.println("Customer B ordered a " + pizza.getName());  
    }  
}
```

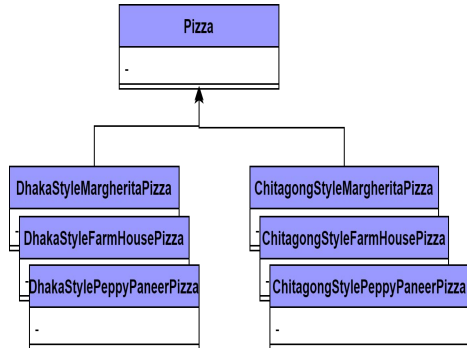


Factory Design Pattern

Creator classes

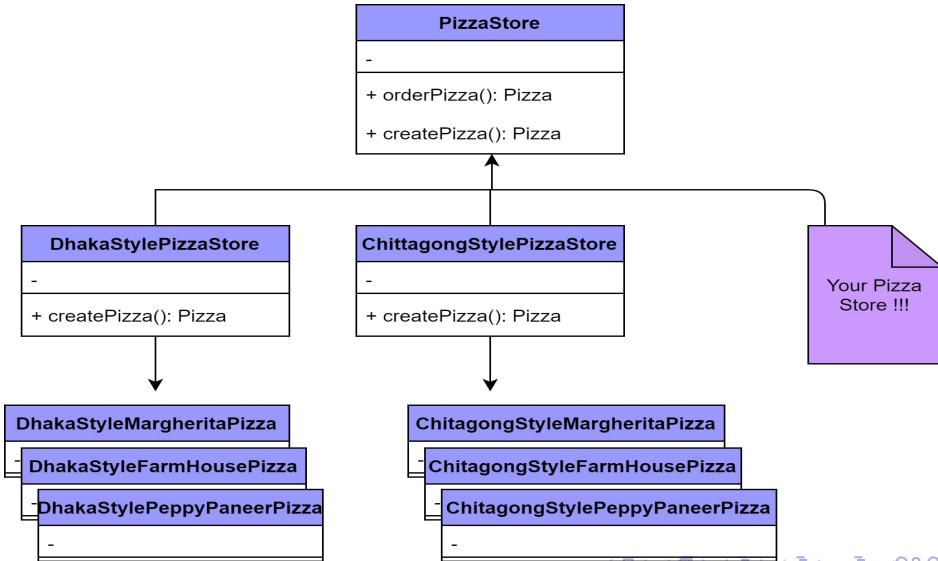


Product classes



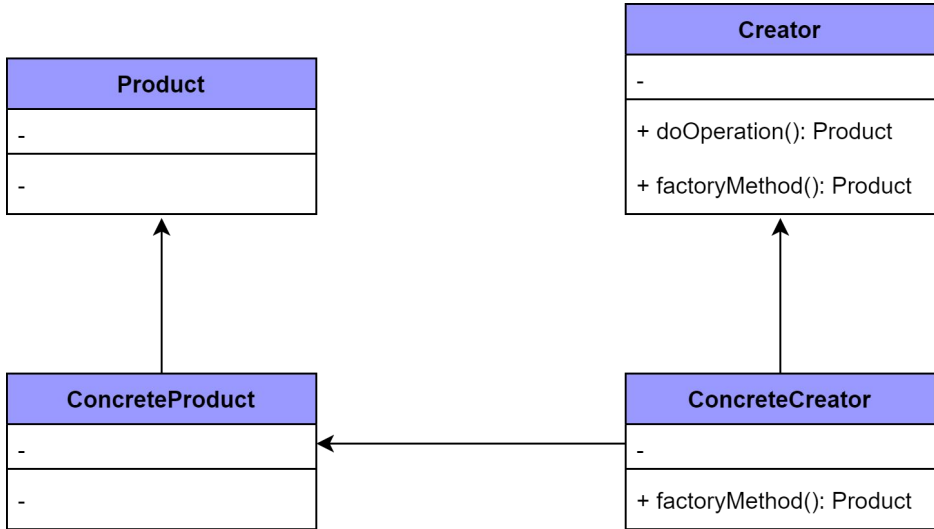


Factory Design Pattern





Factory Design Pattern





**ANY QUESTION ?
THANK YOU !**



Acknowledgements

- [1] Gamma, Erich. Design patterns: elements of reusable object-oriented software. Addison-Wesley Professional, 1 edition, 1994.
- [2] Freeman, Eric, et al. Head first design patterns. " O'Reilly Media, Inc.", 2008.
- [3] TutorialsPoint
- [4] GeeksforGeeks