SWE-4501: Design Pattern

Abstract Factory Design Pattern

Md. Nazmul Haque
Lecturer, IUT

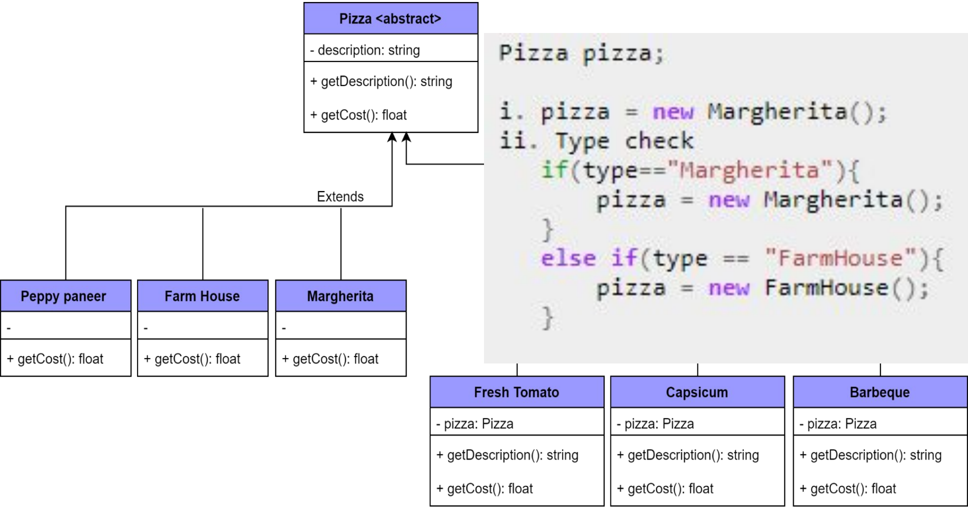
Department of Computer Science and Engineering
Islamic University of Technology

July 26, 2021

Contents

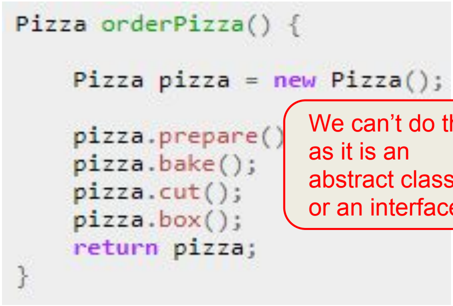
- Motivation
- Solution

Pizza decoration



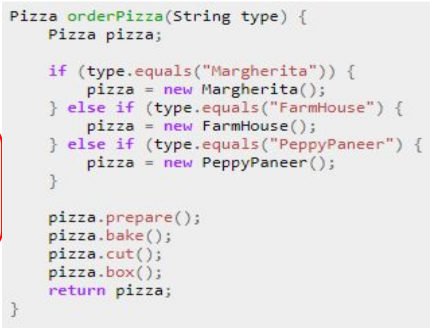
```
Pizza pizza;

i. pizza = new Margherita();
ii. Type check
   if(type=="Margherita"){
        pizza = new Margherita();
   }
   else if(type == "FarmHouse"){
        pizza = new FarmHouse();
   }
```

Pizza decoration

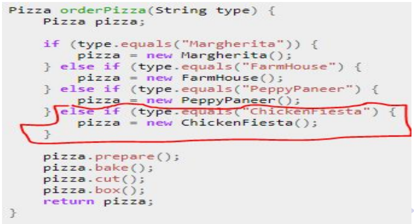


```java
Pizza orderPizza() {

    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.

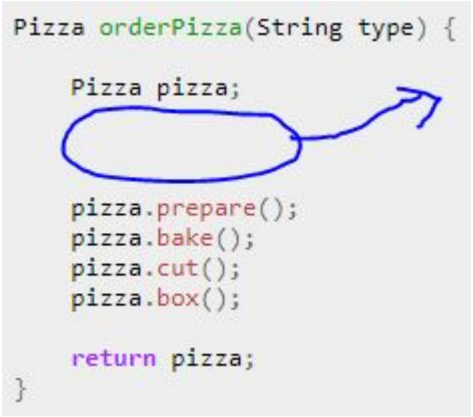


```java
Pizza orderPizza(String type) {
    Pizza pizza;

    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;
}
```

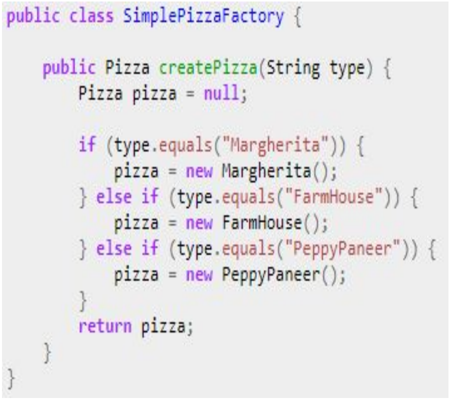


```java
Pizza orderPizza(String type) {
    Pizza pizza;

    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

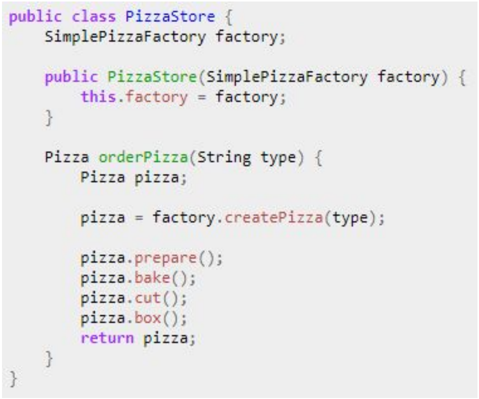


```
Pizza orderPizza(String type) {

    Pizza pizza;



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

    return pizza;
}
```

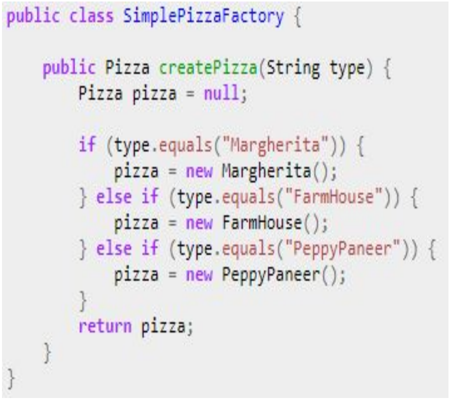


```
public class SimplePizzaFactory {

    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

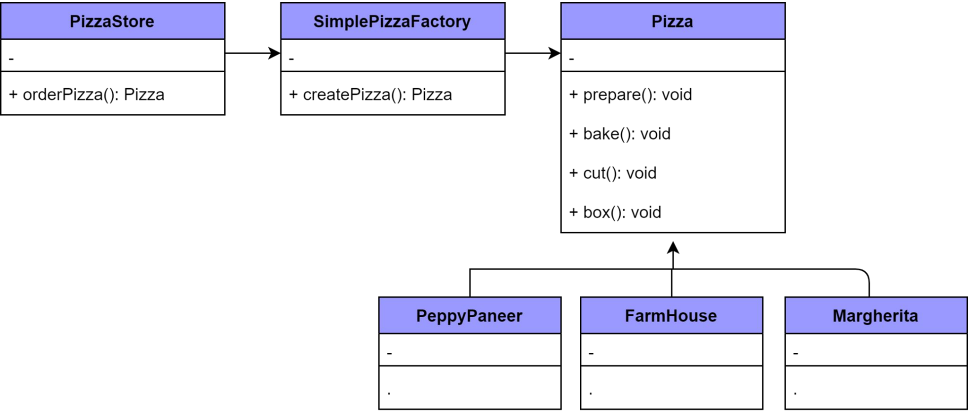


```java
public class PizzaStore {
    SimplePizzaFactory factory;

    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;
    }
}
```

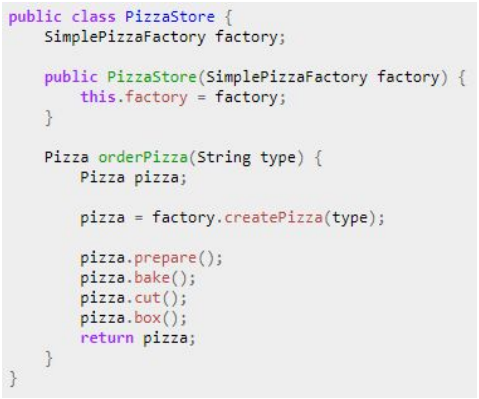


```java
public class SimplePizzaFactory {

    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;
    }
}
```

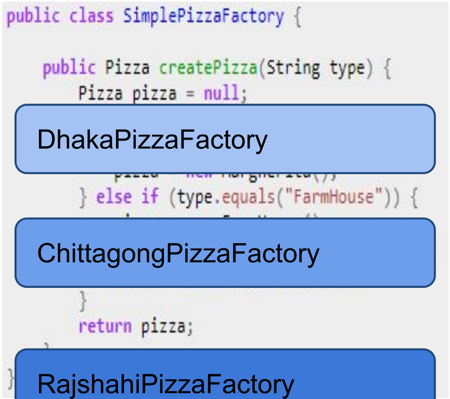
Design a Simple Factory



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;
    }
}
```



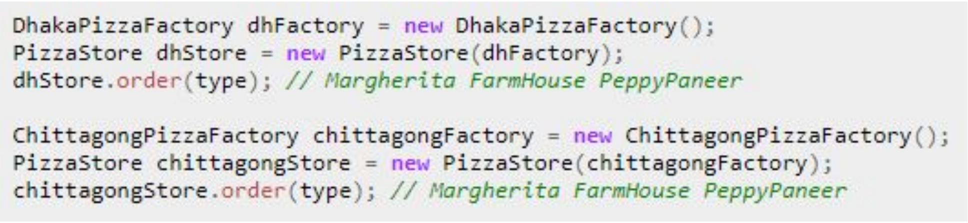
```
public class SimplePizzaFactory {

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

DhakaPizzaFactory

ChittagongPizzaFactory

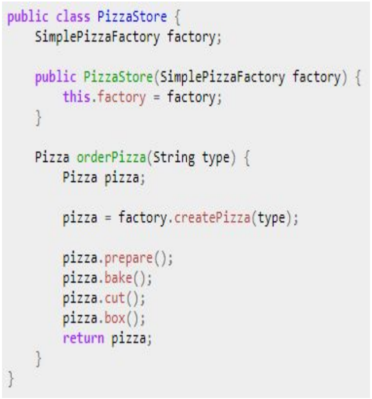
RajshahiPizzaFactory

Pizza decoration

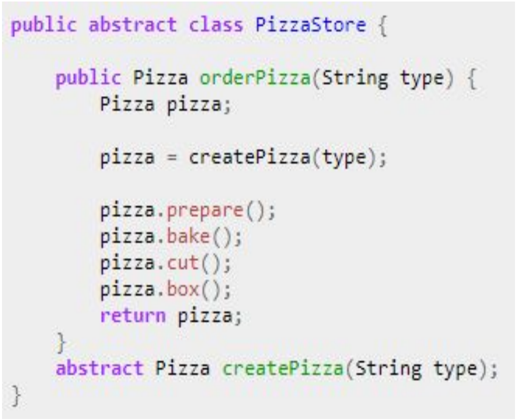


```
DhakaPizzaFactory dhFactory = new DhakaPizzaFactory();
PizzaStore dhStore = new PizzaStore(dhFactory);
dhStore.order(type); // Margherita FarmHouse PeppyPaneer

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

Pizza decoration

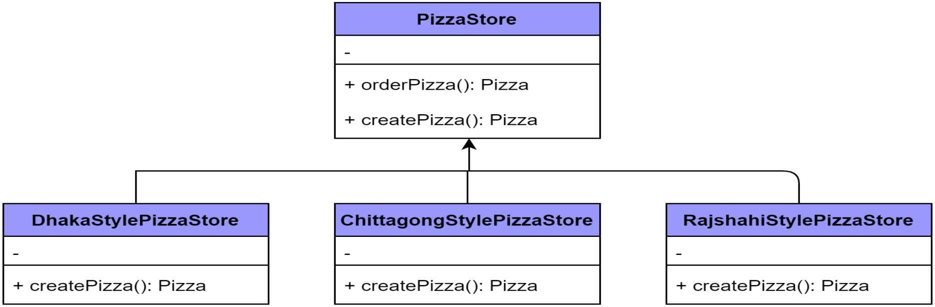


```java
public class PizzaStore {
    SimplePizzaFactory factory;

    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;
    }
}
```



```java
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);
}
```

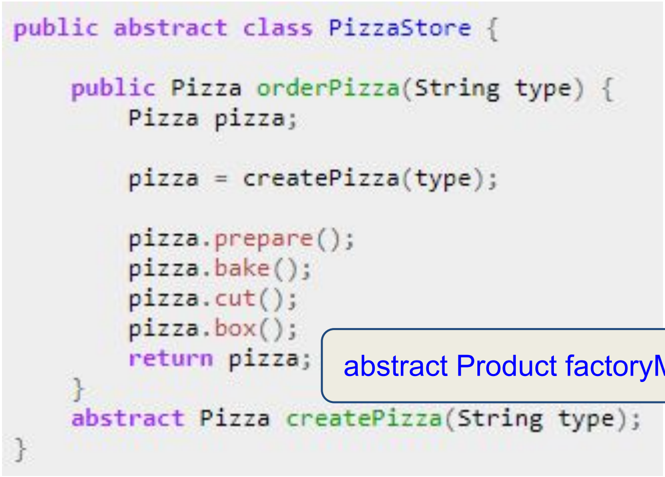
Pizza decoration



```java
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;
}
```

```java
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



```java
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)

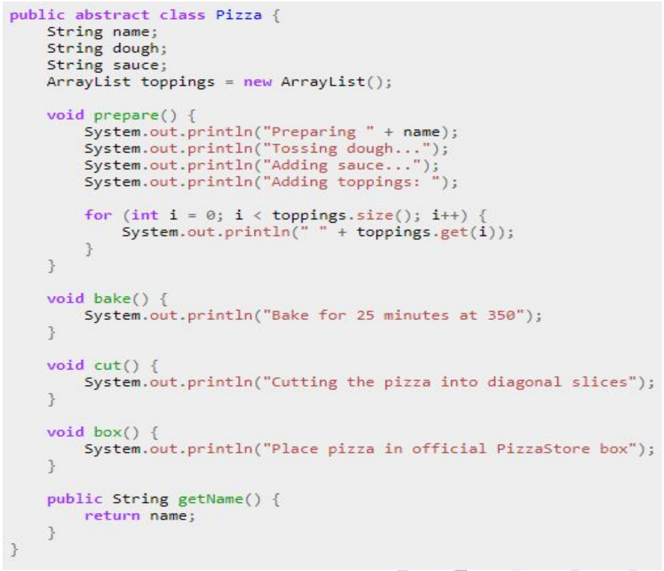
Customer: B (CH)

```
PizzaStore dhStore = new DHPizzaStore();
dhStore.orderPizza("PeppyPaneer");
pizza = createPizza("PeppyPaneer");
```

```
PizzaStore chStore = new CHPizzaStore();
chStore.orderPizza("FarmHouse");
pizza = createPizza("FarmHouse");
```

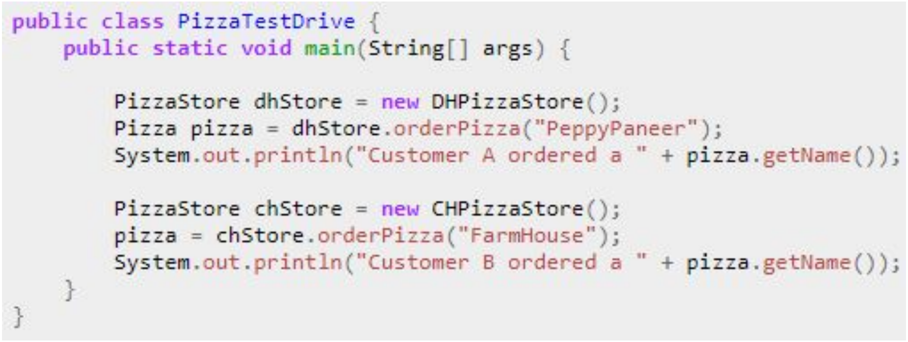
```
pizza.prepare()
pizza.bake()
pizza.cut()
pizza.box()
```

Pizza Class



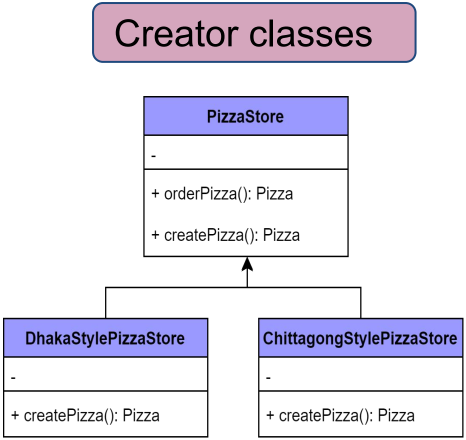
```
public abstract class Pizza {
    String name;
    String dough;
    String sauce;
    ArrayList toppings = new ArrayList();

    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;
    }
}
```

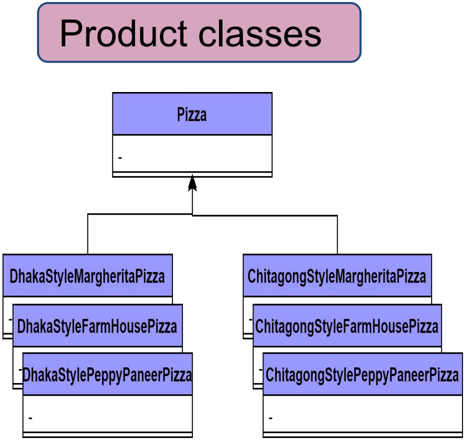
Test Pizza Factory



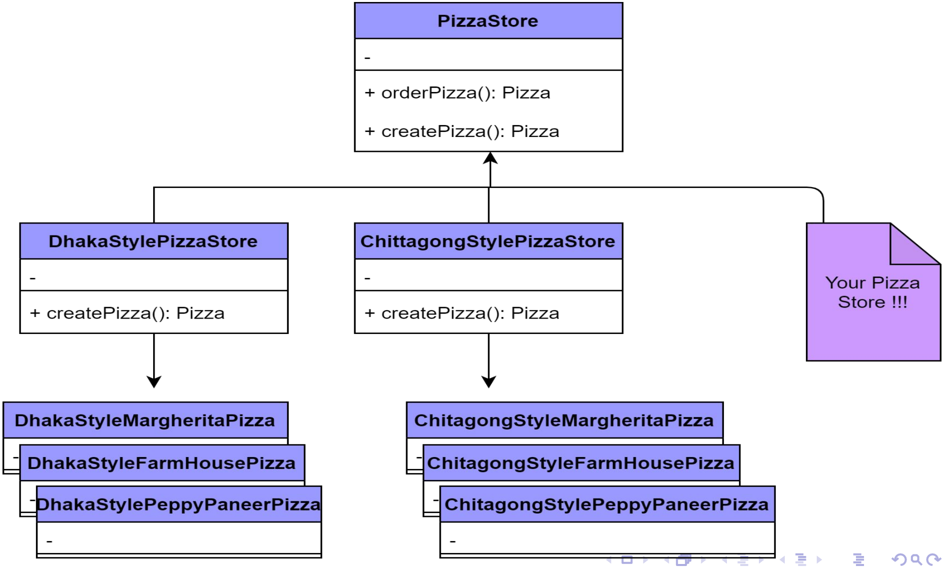
```java
public class PizzaTestDrive {
    public static void main(String[] args) {

        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

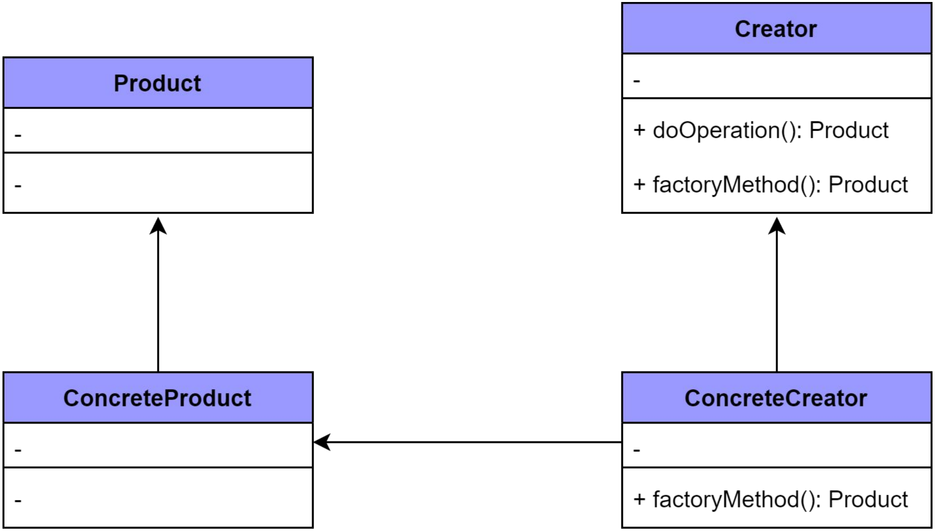



Product classes
Pizza
-
DhakaStyleMargheritaPizza
DhakaStyleFarmHousePizza
DhakaStylePeppyPaneerPizza
-
ChitagongStyleMargheritaPizza
ChitagongStyleFarmHousePizza
ChitagongStylePeppyPaneerPizza
-

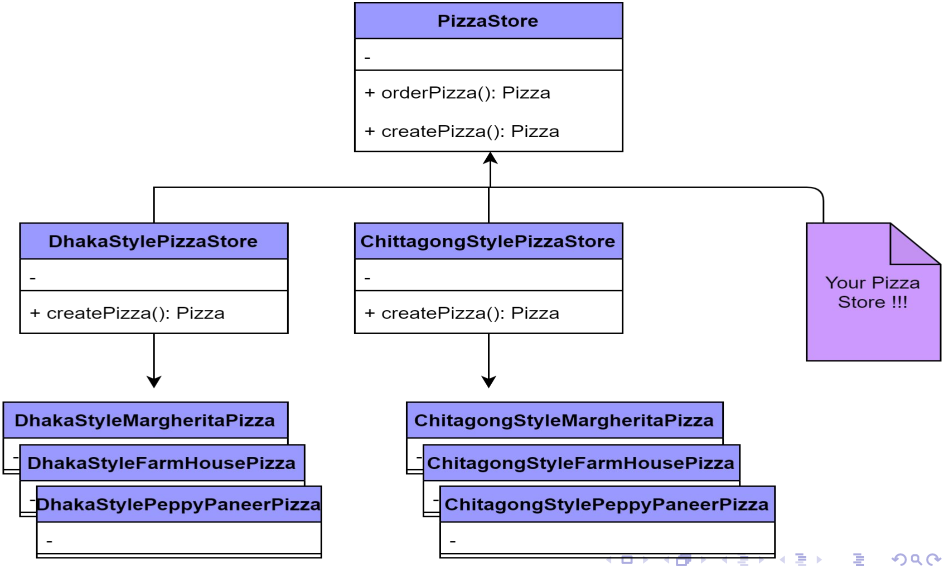
Factory Design Pattern



Factory Design Pattern



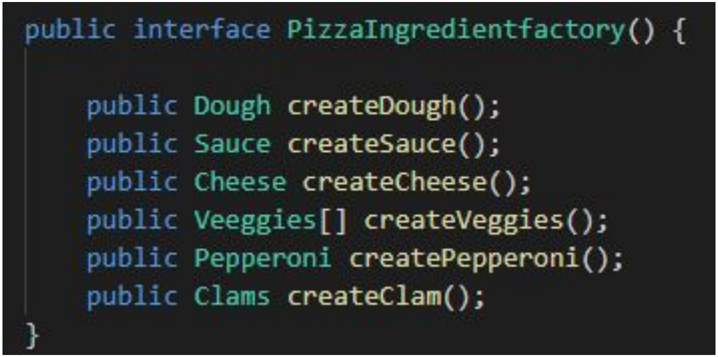
Factory Design Pattern



Families of ingredients

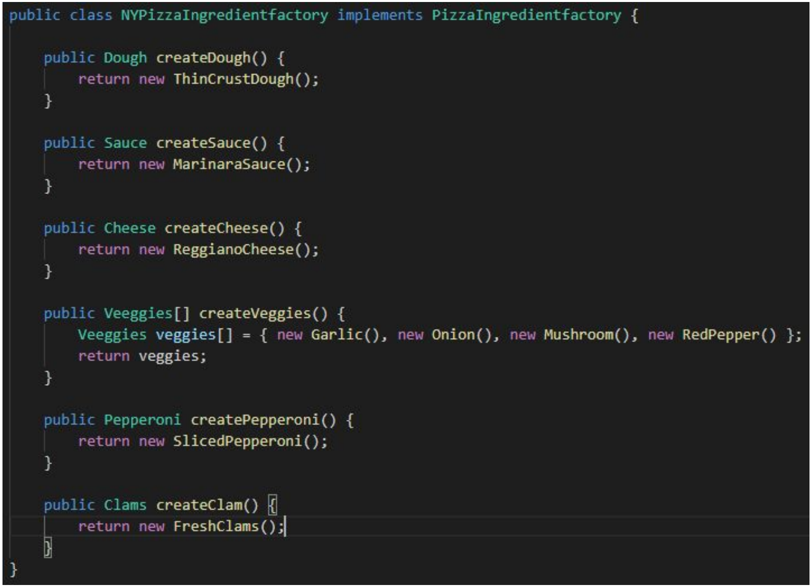
New York	Chicago	California
••FreshClams	••FrozenClams	••FreshClams
••MarinaraSauce	••PlumTomatoSauce	••BrushChettaSauce
••ThinCrustDough	••ThickCrustDough	••VeryThinCrustDough
••ReggianoCheese	••MozzarellaCheese	••GoatCheese

Code snippet of ingredient factory



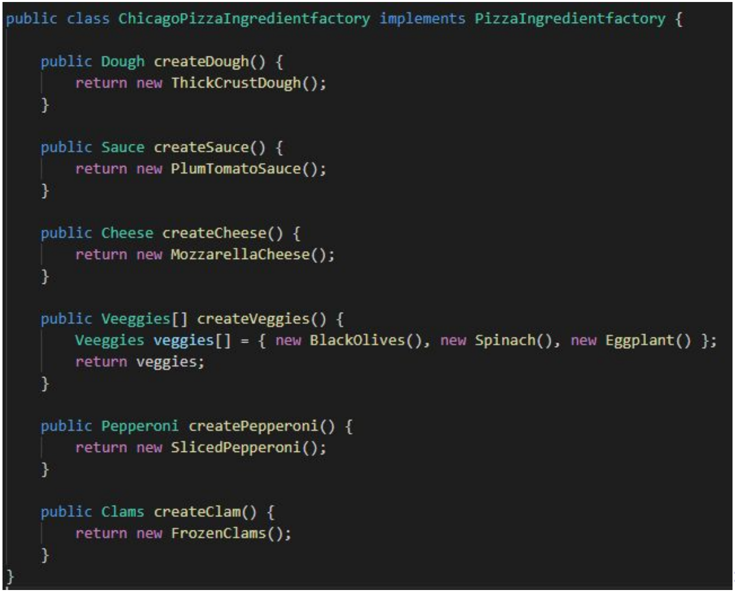
```
public interface PizzaIngredientfactory() {

    public Dough createDough();
    public Sauce createSauce();
    public Cheese createCheese();
    public Veeggies[] createVeggies();
    public Pepperoni createPepperoni();
    public Clams createClam();
}
```

Code snippet of ingredient factory

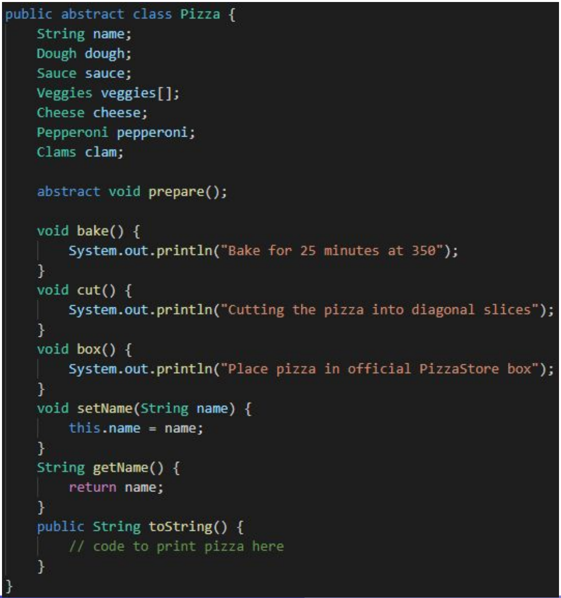


```
public class NYPizzaIngredientfactory implements PizzaIngredientfactory {

    public Dough createDough() {
        return new ThinCrustDough();
    }

    public Sauce createSauce() {
        return new MarinaraSauce();
    }

    public Cheese createCheese() {
        return new ReggianoCheese();
    }

    public Veeggies[] createVeggies() {
        Veeggies veggies[] = { new Garlic(), new Onion(), new Mushroom(), new RedPepper() };
        return veggies;
    }

    public Pepperoni createPepperoni() {
        return new SlicedPepperoni();
    }

    public Clams createClam() {
        return new FreshClams();
    }
}
```

Code snippet of ingredient factory

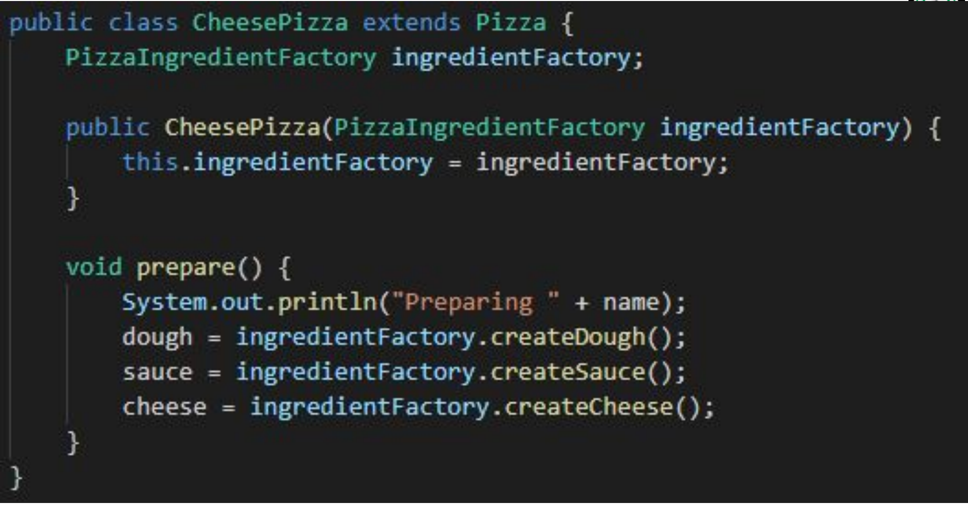

```
public class ChicagoPizzaIngredientfactory implements PizzaIngredientfactory {

    public Dough createDough() {
        return new ThickCrustDough();
    }

    public Sauce createSauce() {
        return new PlumTomatoSauce();
    }

    public Cheese createCheese() {
        return new MozzarellaCheese();
    }

    public Veeggies[] createVeggies() {
        Veeggies veggies[] = { new BlackOlives(), new Spinach(), new Eggplant() };
        return veggies;
    }

    public Pepperoni createPepperoni() {
        return new SlicedPepperoni();
    }

    public Clams createClam() {
        return new FrozenClams();
    }
}
```

Code snippet of ingredient factory



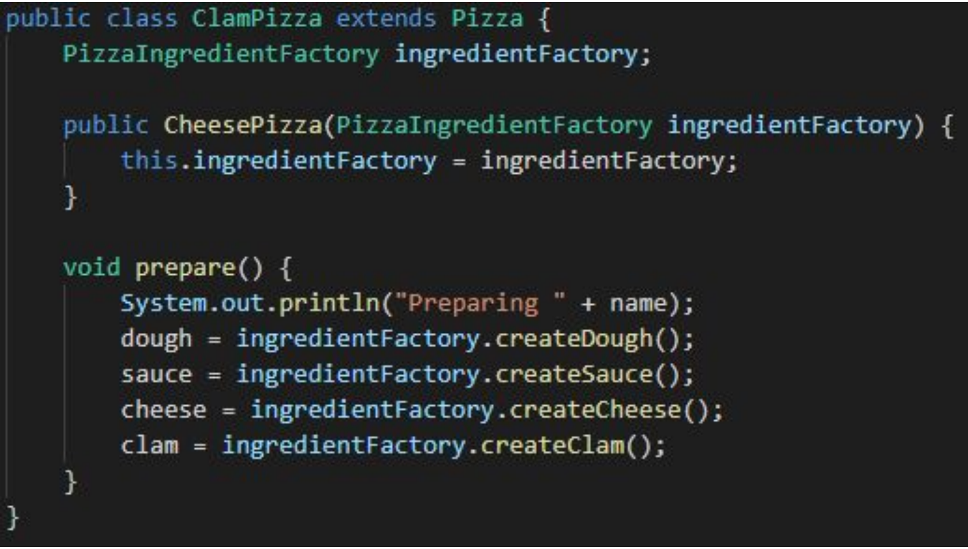
```java
public abstract class Pizza {
    String name;
    Dough dough;
    Sauce sauce;
    Veggies veggies[];
    Cheese cheese;
    Pepperoni pepperoni;
    Clams clam;

    abstract void prepare();

    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");
    }
    void setName(String name) {
        this.name = name;
    }
    String getName() {
        return name;
    }
    public String toString() {
        // code to print pizza here
    }
}
```

Code snippet of ingredient factory



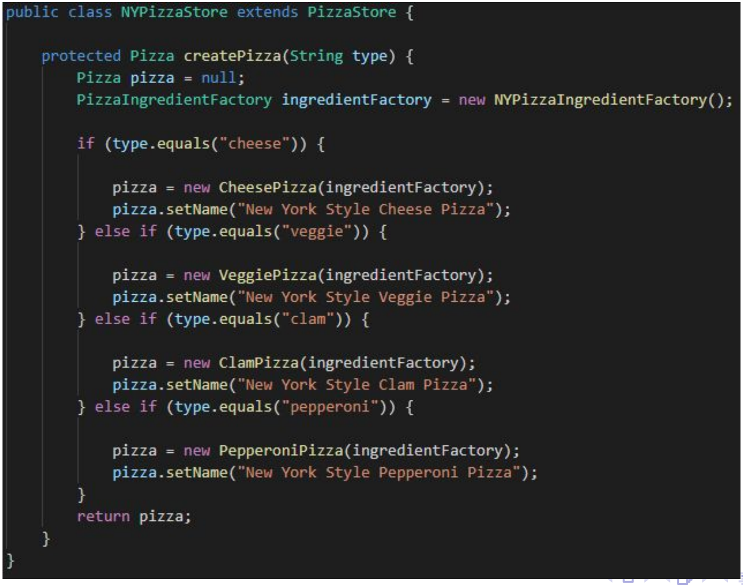
```
public class CheesePizza extends Pizza {
    PizzaIngredientFactory ingredientFactory;

    public CheesePizza(PizzaIngredientFactory ingredientFactory) {
        this.ingredientFactory = ingredientFactory;
    }

    void prepare() {
        System.out.println("Preparing " + name);
        dough = ingredientFactory.createDough();
        sauce = ingredientFactory.createSauce();
        cheese = ingredientFactory.createCheese();
    }
}
```

Code snippet of ingredient factory



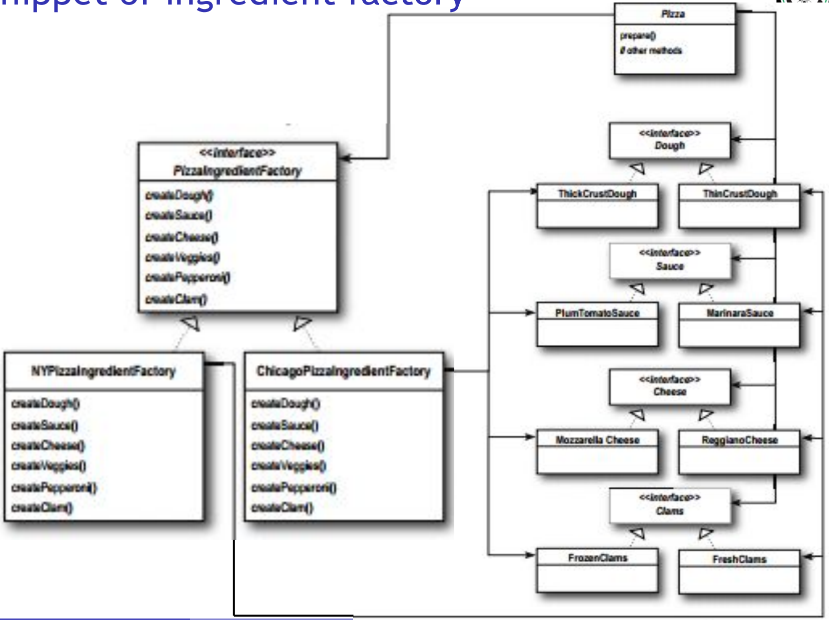
```
public class ClamPizza extends Pizza {
    PizzaIngredientFactory ingredientFactory;

    public CheesePizza(PizzaIngredientFactory ingredientFactory) {
        this.ingredientFactory = ingredientFactory;
    }

    void prepare() {
        System.out.println("Preparing " + name);
        dough = ingredientFactory.createDough();
        sauce = ingredientFactory.createSauce();
        cheese = ingredientFactory.createCheese();
        clam = ingredientFactory.createClam();
    }
}
```

Code snippet of ingredient factory

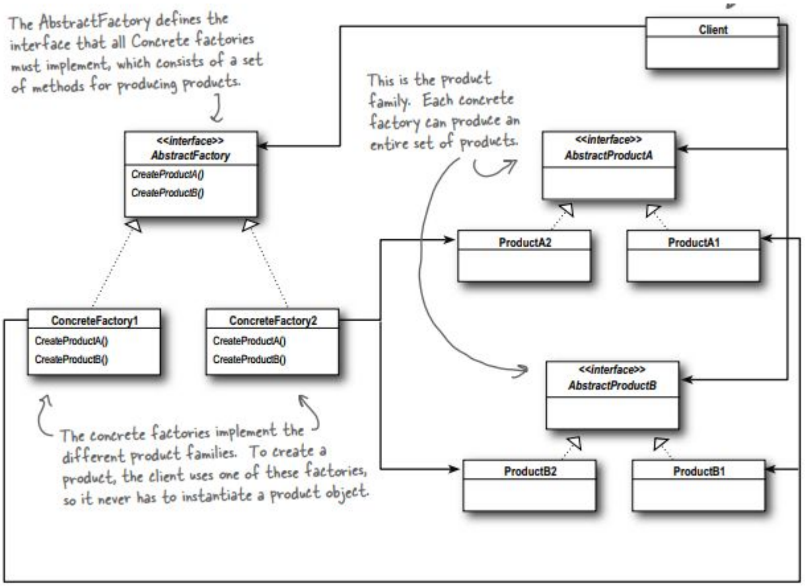


```java
public class NYPizzaStore extends PizzaStore {

    protected Pizza createPizza(String type) {
        Pizza pizza = null;
        PizzaIngredientFactory ingredientFactory = new NYPizzaIngredientFactory();

        if (type.equals("cheese")) {

            pizza = new CheesePizza(ingredientFactory);
            pizza.setName("New York Style Cheese Pizza");
        } else if (type.equals("veggie")) {

            pizza = new VeggiePizza(ingredientFactory);
            pizza.setName("New York Style Veggie Pizza");
        } else if (type.equals("clam")) {

            pizza = new ClamPizza(ingredientFactory);
            pizza.setName("New York Style Clam Pizza");
        } else if (type.equals("pepperoni")) {

            pizza = new PepperoniPizza(ingredientFactory);
            pizza.setName("New York Style Pepperoni Pizza");
        }
        return pizza;
    }
}
```

Code snippet of ingredient factory



Abstract Factory Design Pattern


The AbstractFactory defines the interface that all Concrete factories must implement, which consists of a set of methods for producing products.
<<interface>>
AbstractFactory
CreateProductA()
CreateProductB()
Client
This is the product family. Each concrete factory can produce an entire set of products.
<<interface>>
AbstractProductA
ProductA2
ProductA1
ConcreteFactory1
CreateProductA()
CreateProductB()
ConcreteFactory2
CreateProductA()
CreateProductB()
<<interface>>
AbstractProductB
ProductB2
ProductB1
The concrete factories implement the different product families. To create a product, the client uses one of these factories, so it never has to instantiate a product object.

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