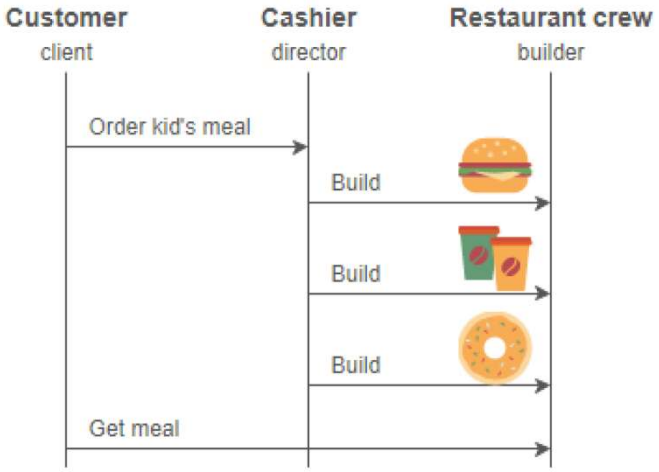
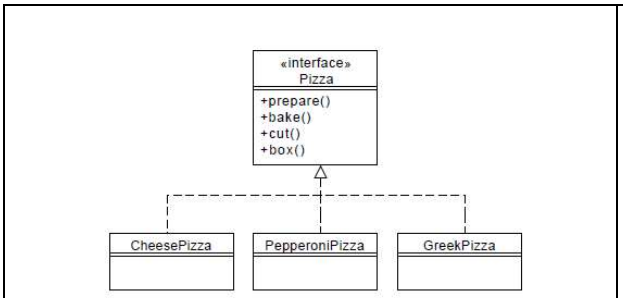
I3305
Graphical Interface and Application
Lab 1 : Creational Patterns

Exercise 1: Builder pattern The Builder pattern separates the construction of a complex object from its representation so that the same construction process can create different representations. This pattern is used by fast food restaurants to construct children's and Adult meals. Children's meals typically consist of a main item, a drink, and a dessert (e.g., a hamburger, fries, Coke, and ice cream). Note that there can be variation in the content of the children's or adult meal, but the construction process is the same. Whether a customer orders a hamburger, cheeseburger, or chicken, the process is the same. The employee at the counter directs the crew to assemble a main item and dessert. These items are then placed in a bag. The drink is placed in a cup and remains outside of the bag. This same process is used at competing restaurants.



Write the necessary code in JAVA to solve this problem.

Exercise 2 : Factory method pattern Let's say you have a pizza shop. To create a pizza, you need to prepare it first, bake it, cut it and then box it. Your pizza shop offers different types of pizza : Cheese Pizza, Greek Pizza, Pepperoni Pizza. Here's the initial version of code:




```
public class PizzaStore {

    Pizza orderPizza(String type) {
        Pizza pizza = null;
        if(type.equals("cheese"))
            pizza = new CheesePizza();
        else if(type.equals("greek"))
            pizza = new GreekPizza();
        else if(type.equals("pepporoni"))
            pizza = new PepperoniPizza();

        // Excellent example of "coding to an interface"
        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();

        return pizza;
    }
}
```

```
public interface Pizza {
    public void prepare();
    public void bake();
    public void cut();
    public void box();
}
```

Build a simple pizza factory in JAVA. Write the Store class.

You have expanded your work and thus you have multiple regional pizza stores. What varies among these regional stores is the style of pizzas they make - Tripoli Pizza has thin crust, Beirut Pizza has thick, and so on. Modify your code to make it responsible for creating the right kind of pizza.

Exercise 3: Singleton pattern Everyone knows that all modern chocolate factories have computer controlled chocolate boilers. The job of the boiler is to take in chocolate and milk, bring them to a boil, and then pass them on to the next phase of making chocolate bars.

Here's the controller class for Choc-O-Holic, Inc.'s industrial strength Chocolate Boiler. Check out the code ; you'll notice they've tried to be very careful to ensure that bad things don't happen, like draining 500 gallons of unboiled mixture, or filling the boiler when it's already full, or boiling an empty boiler !

```
public class ChocolateBoiler {
        private boolean empty;
        private boolean boiled;

        public  ChocolateBoiler() {
                empty = true;
                boiled = false;
        }

        public void fill() {
                if (isEmpty()) {
                        empty = false;
                        boiled = false;
                        // fill the boiler with a milk/chocolate mixture
                }
        }

        public void drain() {
                if (!isEmpty() && isBoiled()) {
                        // drain the boiled milk and chocolate
                        empty = true;
                }
        }

        public void boil() {
                if (!isEmpty() && !isBoiled()) {
                        // bring the contents to a boil
                        boiled = true;
                }
        }

        public boolean isEmpty() {
                return empty;
        }

        public boolean isBoiled() {
                return boiled;
        }
}
```

Choc-O-Holic has done a decent job of ensuring bad things don't happen, don't you think ? Then again, you probably suspect that if two ChocolateBoiler instances get loose, some very bad things can happen.
How might things go wrong if more than one instance of ChocolateBoiler is created in an application ?
Can you help Choc-O-Holic improve their ChocolateBoiler class by turning it into a singleton ?