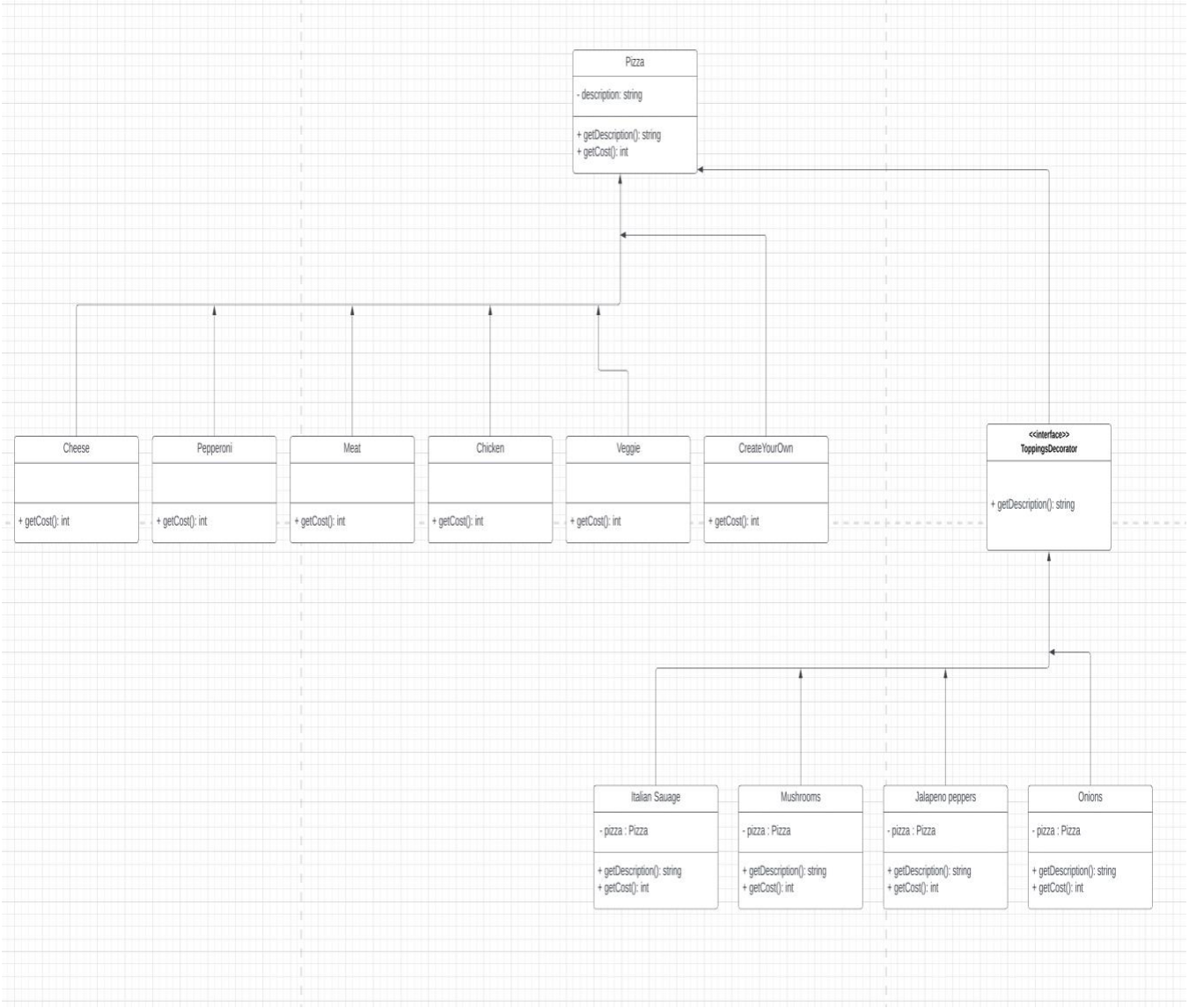
Assignment – 3

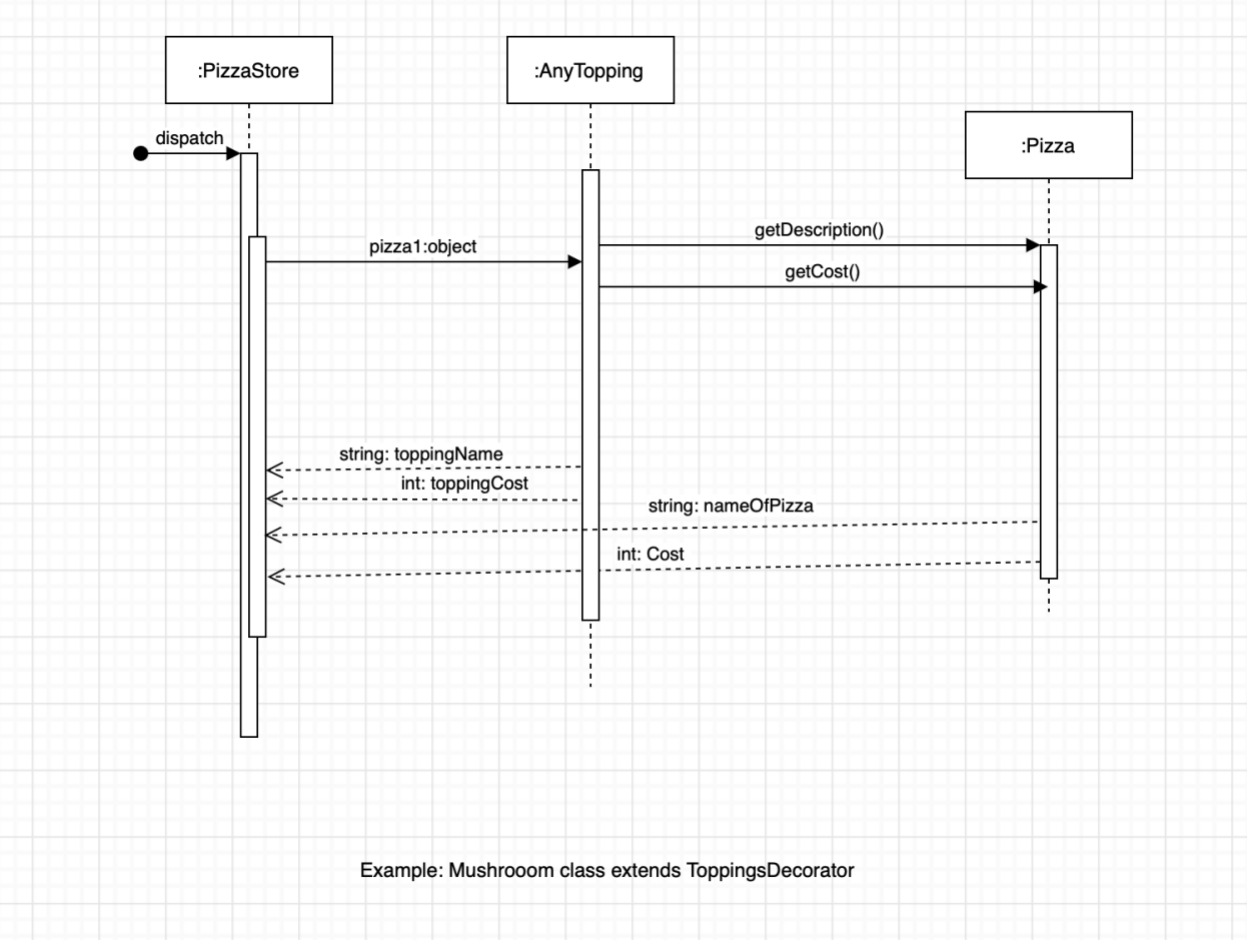
**Find one application of structural patterns to your system and implement in java. Create a class diagram and sequence diagram for it.**

**Solution:** Using decorator design pattern for pizza ordering with toppings.

**Class Diagram:**



**Sequence Diagram:**




//AnyTopping class example is elaborated in Code

**Implementation in Java:**

Pizza Class:

```
abstract class Pizza
{
   // it is an abstract pizza
   String description = "Unkknown Pizza";

   public String getDescription()
   {
      return description;
   }

   public abstract int getCost();
}

//Implementation of decorator as interface.
abstract class ToppingsDecorator extends Pizza
{
   public abstract String getDescription();
}

// Concrete pizza classes
class Cheese extends Pizza
{
   public Cheese() { description = "Cheese Pizza ($14)"; }
   public int getCost() {  return 14; }
}
class Pepperoni extends Pizza
{
   public Pepperoni() {  description = "Pepperoni Pizza ($17)"; }
   public int getCost() { return 17; }
}
class Meat extends Pizza
{
   public Meat()  { description = "Meat Pizza ($20)"; }
   public int getCost() { return 20;  }
}
class Chicken extends Pizza
{
   public Chicken() { description = "Chicken Pizza ($20)";}
   public int getCost() { return 20; }
}
```

```
class Veggie extends Pizza
{
public Veggie() { description = "Veggie Pizza ($20)"; }
public int getCost() {  return 20;  }
}

class CreateYourOwn extends Pizza
{
public CreateYourOwn() { description = "CreateYourOwn ($14)"; }
public int getCost() {  return 14;  }
}

// Concrete toppings classes
class ItalianSausage extends ToppingsDecorator
{
   // we need a reference to obj we are decorating
   Pizza pizza;

   public ItalianSausage(Pizza pizza) { this.pizza = pizza; }
   public String getDescription() {
      return pizza.getDescription() + ", +Italian Sausage ($2)";
   }
   public int getCost() { return 2 + pizza.getCost(); }
}
class JalapenoPeppers extends ToppingsDecorator
{
   Pizza pizza;
   public JalapenoPeppers(Pizza pizza) {  this.pizza = pizza;  }
   public String getDescription() {
      return pizza.getDescription() + ", +JalapenoPeppers ($2)";
   }
   public int getCost() { return 2 + pizza.getCost(); }
}

class Mushrooms extends ToppingsDecorator
{
   Pizza pizza;
   public Mushrooms(Pizza pizza)  {  this.pizza = pizza; }
   public String getDescription() {
      return pizza.getDescription() + ", +Mushrooms ($3)";
   }
   public int getCost()  {  return 3 + pizza.getCost(); }
}
```

```
class Onions extends ToppingsDecorator
{
   Pizza pizza;
   public Onions(Pizza pizza)  {  this.pizza = pizza; }
   public String getDescription() {
      return pizza.getDescription() + ", +Onions ($2)";
   }
   public int getCost()  {  return 2 + pizza.getCost(); }
}
// Other toppings can be implemented similarly

// Main class and method
class PizzaStore
{
   public static void main(String args[])
   {
      // create new Meat pizza
      Pizza pizza = new Meat();
      System.out.println( pizza.getDescription() +
                  "\nTotal Cost : $" + pizza.getCost());

      // create new Pepperoni pizza
      Pizza pizza2 = new Pepperoni();

      // decorate it with ItalianSausage topping
      pizza2 = new ItalianSausage(pizza2);

      //decorate it with Mushrooms topping
      pizza2 = new Mushrooms(pizza2);

      System.out.println( pizza2.getDescription() +
                  "\nTotal Cost : $" + pizza2.getCost());
   }
}
```

## Output:

```
PROBLEMS 4   OUTPUT   TERMINAL   DEBUG CONSOLE                                   Code

[Done] exited with code=0 in 0.366 seconds

[Running] cd "/Users/juythikkalathiya/Desktop/Coding/" && javac PizzaStore.java && java PizzaStore
Meat Pizza ($20)
Total Cost : $20
Pepperoni Pizza ($17), +Italian Sausage ($2), +Mushrooms ($3)
Total Cost : $22

[Done] exited with code=0 in 0.442 seconds

Ln 115, Col 54   Spaces: 4   UTF-8   LF   {} Java
```