

2D1385 Hemtenta

2007-05-23

Q1.(i) 10 points

Data Dictionary

Nouns

Carlo Speziale

cheese : can be mozzarella or Reggiano

Chicago pizza ingredient factory

clam: can be fresh or frozen

diner

Dough : can be thin or thick crust

fresh clam

frozen clam

garlic

marinara sauce

Mozarella cheese:

New York pizza ingredient factory

pineapple

pizza

pizza ingredient factory: can be New York pizza ingredient factory or Chicago pizza ingredient factory

pizza cook: actor, invokes the pizza create method

plum tomato sauce

Reggiano cheese

Sauce : can be plum tomato or marinara

Thick crust dough

Thin crust dough

Verbs

create : we can create pizzas, dough, sauce, cheese or clams

reference : is the singleton method getUniqueInstance

Relational Phrases

Carlo speciale uses thick crust dough: one to one, asymmetric

Carlo speciale uses plum tomato sauce: one to two, asymmetric

Carlo speciale uses mozzarella cheese: optional, two if present, asymmetric

Carlo speciale uses frozen clams: optional, but must be present if mozzarella is used in which case 10 to 15, asymmetric

Carlo speciale uses garlic: optional, but present if pineapple is absent, asymmetric

Carlo speciale uses pineapple: optional, but present if garlic is absent, asymmetric

Chicago pizza ingredient factory depends on thick crust dough

Chicago pizza ingredient factory depends on plum tomato sauce

Chicago pizza ingredient factory depends on mozzarella cheese

Chicago pizza ingredient factory depends on frozen clams

Chicago pizza ingredient factory is a pizza ingredient factory

diner eats pizza

Fresh clam is a clam

Frozen clam is a clam

Marinara sauce is a sauce

Mozarella cheese is a cheese

New York pizza ingredient factory depends on thin crust dough

New York pizza ingredient factory depends on marinara sauce

New York pizza ingredient factory depends on Reggiano cheese

New York pizza ingredient factory depends on fresh clams

New York pizza ingredient factory is a pizza ingredient factory

Pizza depends on pizza ingredients factory : through creation methods for dough etc

Pizza depends on dough

Pizza depends on sauce

Pizza depends on cheese

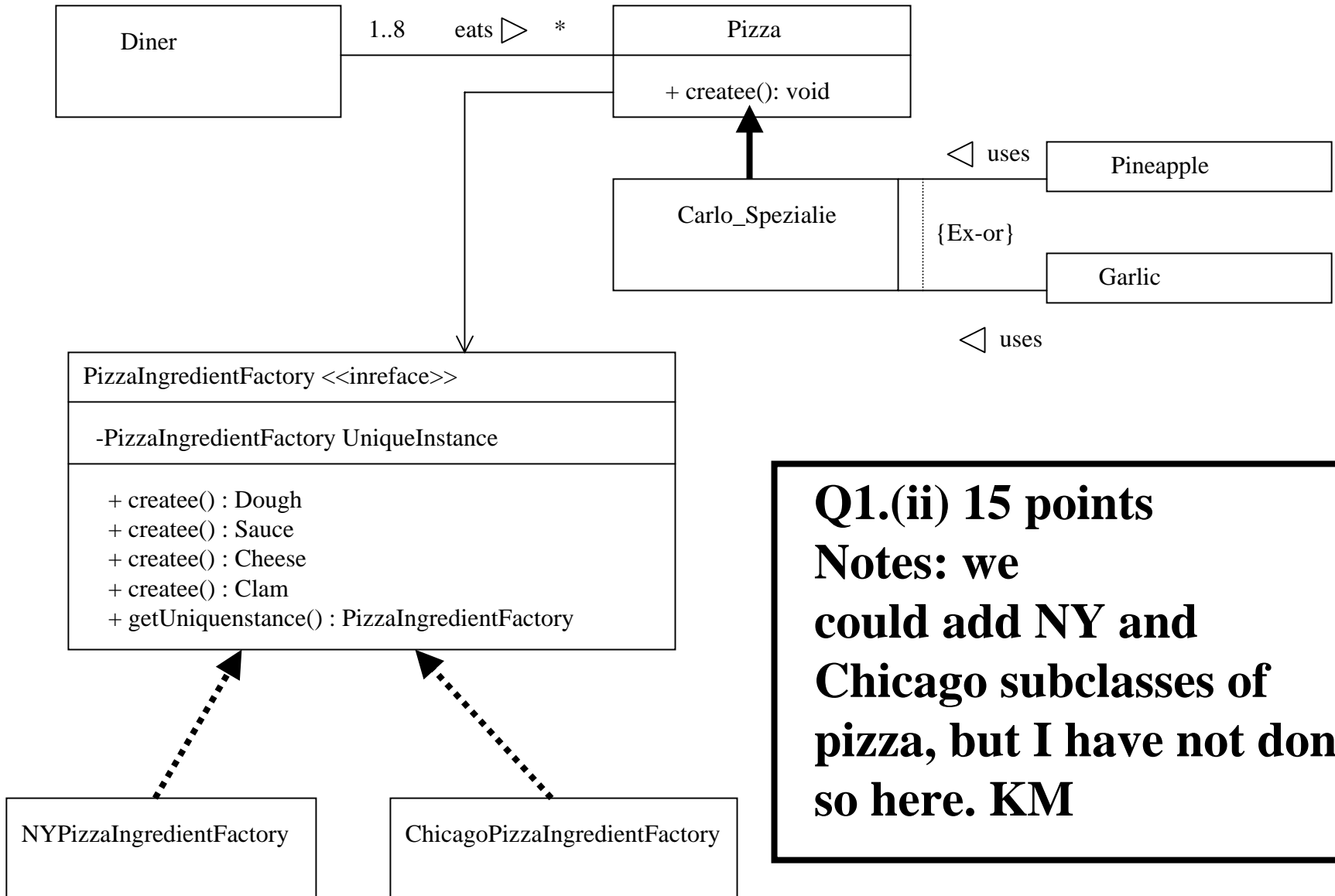
Pizza depends on clam

Plum tomato sauce is a sauce

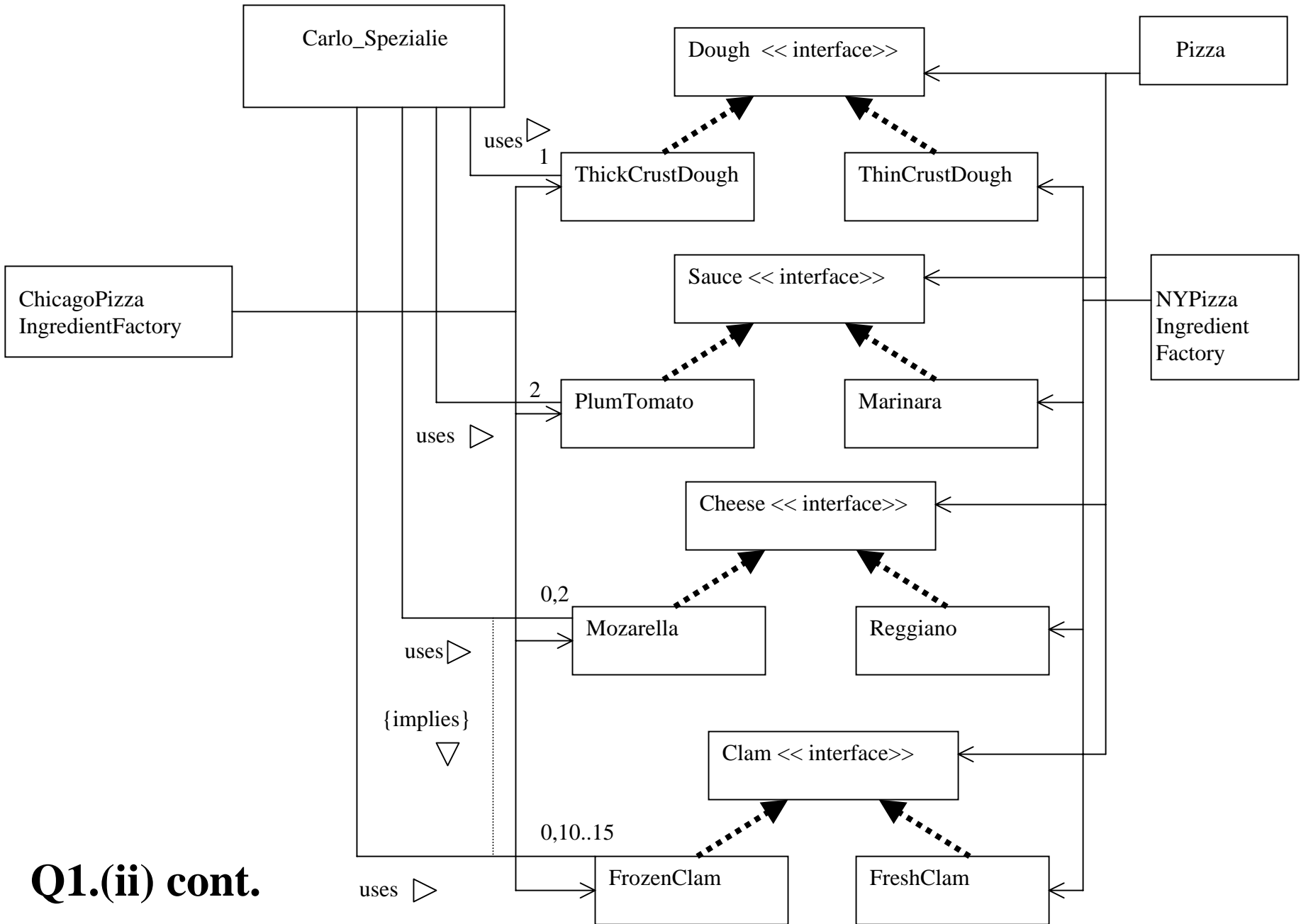
Reggiano cheese is a cheese

Thick crust dough is a dough

Thin crust dough is a dough

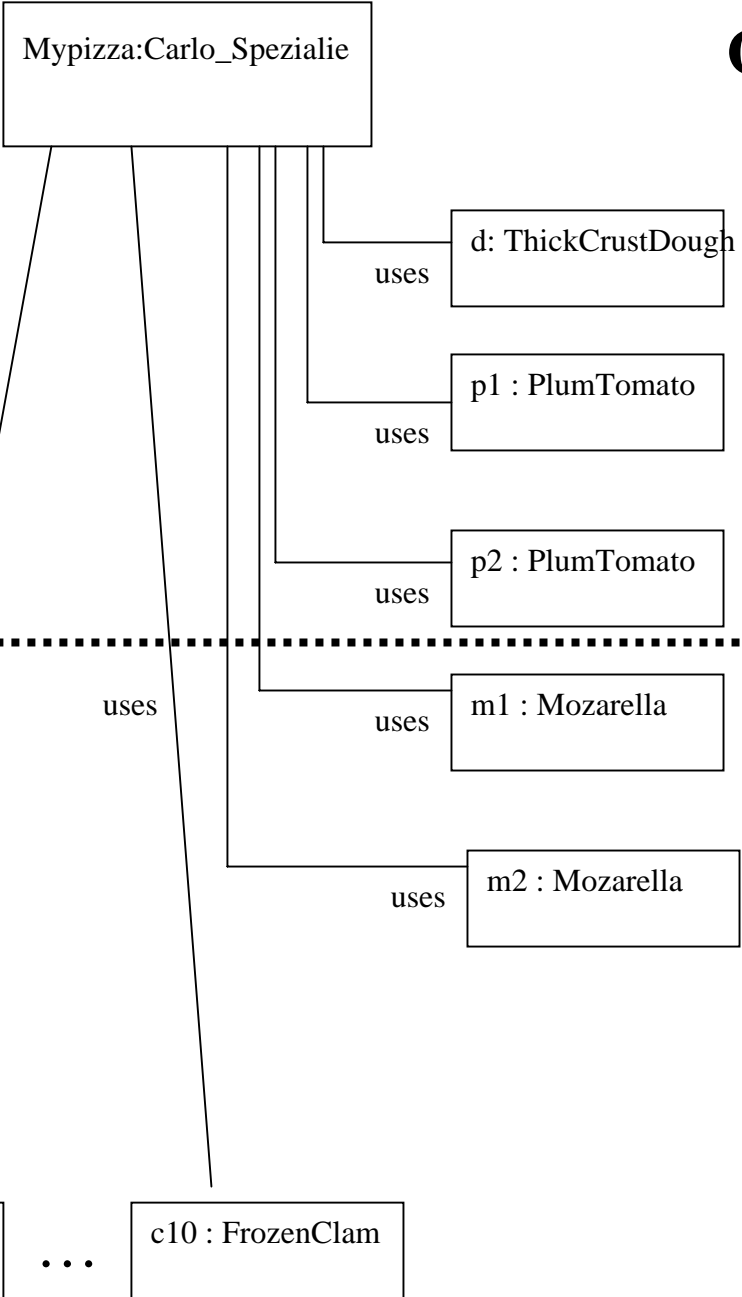


Q1.(ii) 15 points
Notes: we could add NY and Chicago subclasses of pizza, but I have not done so here. KM

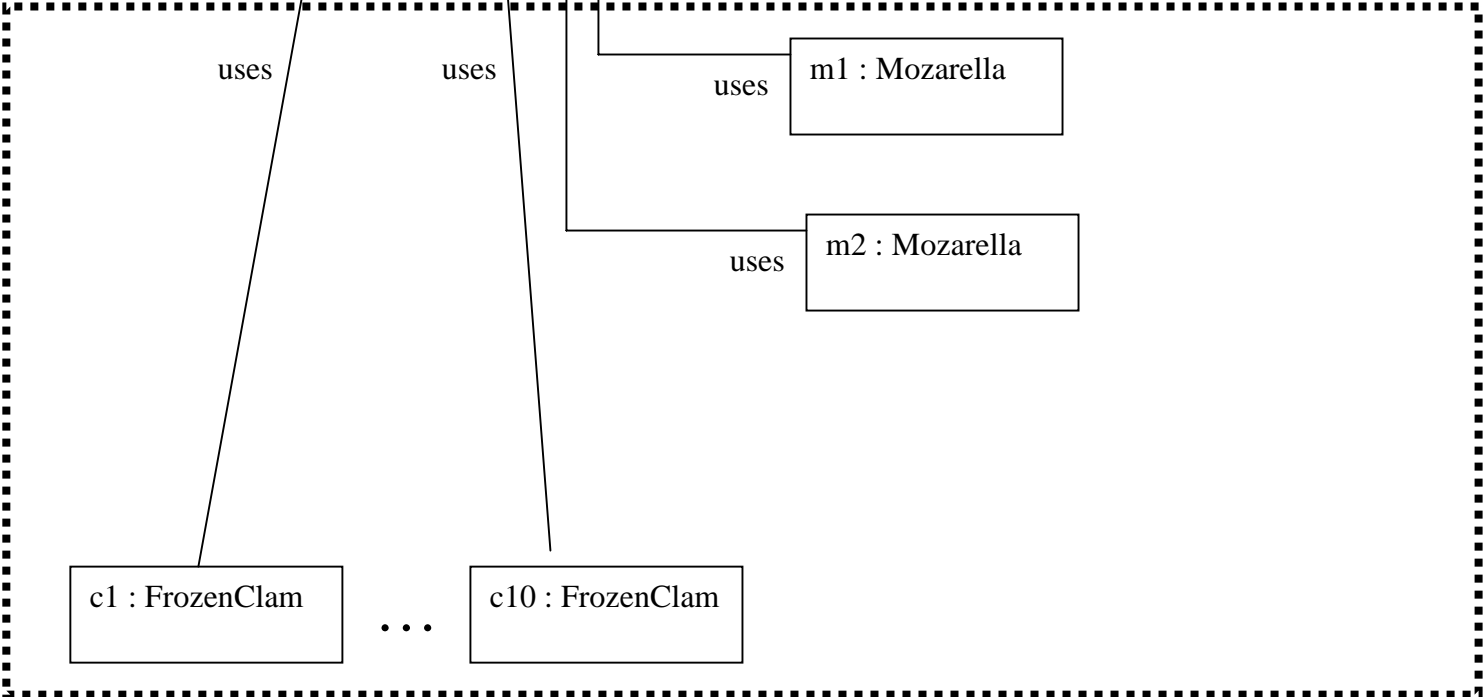


Q1.(ii) cont.

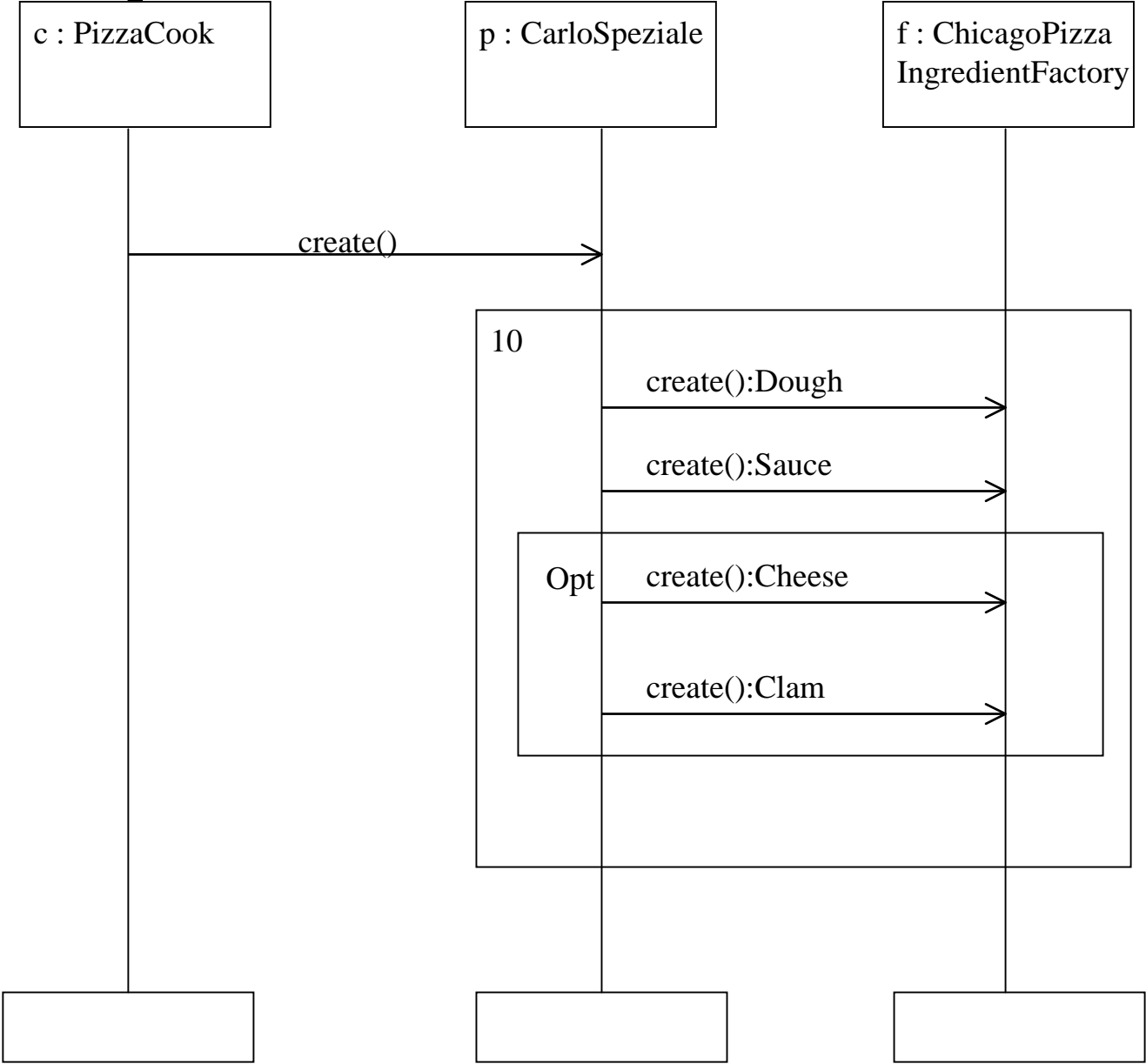
Q1.(iii) 3 points



Optional !



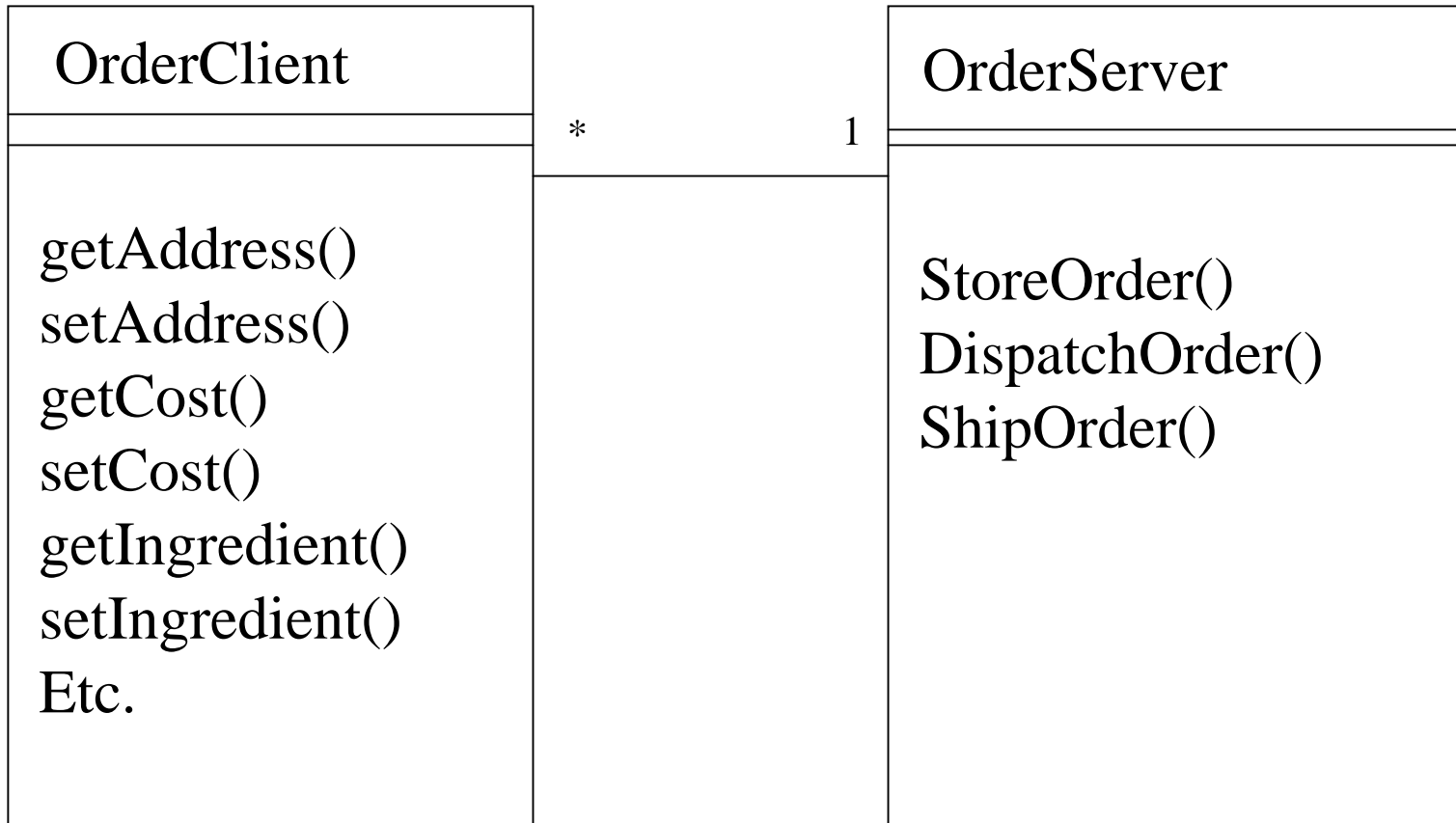
Q1.(iv) 2 points



Q3. 8 points

The Professor uses a client-server architecture as a classic solution to a distributed computing problem. He has used MVC to separate the logical model of pizza orders from the client side display of view and control, which may vary between browsers.

He will use thin web browser clients to allow manipulation of orders by customers, including ingredients, address, etc, and an order server to store and dispatch orders to cooks, as well as ship orders.



Client has graphical view of order (e.g. pizza types) and graphical controller to manipulate ingredients, address etc.

Server has logical model of order including address, cost, ingredients, time etc.

Q3. (i) 5 points

Notice the use of the entity FiveFrozenClam to compact the definition

```
<DOCTYPE pizzaOrder [  
<!ELEMENT pizzaOrder ( Order+ ) >  
<!ELEMENT Order (standard | fullyCustom ) >  
<!ELEMENT standard ( CarloSpeziale ) >  
<ENTITY FiveFrozenClam (frozenClam, frozenClam, frozenClam, frozenClam,  
frozenClam)>  
<!ELEMENT CarloSpeziale (thickCrustDough, plumTomatoSauce,  
    plumTomatoSauce, [ mozzarellaCheese, mozzarellaCheese,  
    &FiveFrozenClam, &FiveFrozenClam, ( [frozenClam, [frozenClam,  
    [frozenClam, [frozenClam ] ] ] ) | [ FiveFrozenClam] ) ) >  
<!ELEMENT fullyCustom ( dough, (sauce)*, (cheese)*, (clam)* ) >  
<!ELEMENT dough ( thickCrustDough | thinCrustDough ) >  
<!ELEMENT sauce ( plumTomatoSauce | marinaraSauce ) >  
<!ELEMENT cheese ( mozzarellaCheese | reggianoCheese ) >  
<!ELEMENT clam ( frozenClam | freshClam ) >
```

```
<!ELEMENT thickCrustDough ( EMPTY ) >  
<!ELEMENT thinCrustDough ( EMPTY ) >  
<!ELEMENT plumTomatoSauce ( EMPTY ) >  
<!ELEMENT marinaraSauce ( EMPTY ) >  
<!ELEMENT mozzarellaCheese ( EMPTY ) >  
<!ELEMENT reggianoCheese ( EMPTY ) >  
<!ELEMENT frozenClam ( EMPTY ) >  
<!ELEMENT freshClam ( EMPTY ) >  
</pizzaOrder> ]
```

(ii) 2 points This speciale pizza has no cheese and hence no clams!

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<pizzaOrder>
```

```
<Order>
```

```
<standard>
```

```
<CarloSpeziale>
```

```
<thickCrustDough></thickCrustDough>
```

```
<plumTomatoSauce></plumTomatoSauce>
```

```
<plumTomatoSauce></plumTomatoSauce>
```

```
</CarloSpeziale>
```

```
</standard>
```

```
</Order>
```

```
</pizzaOrder>
```

(iii) 1 point We can see by the DTD above that it exactly captures the recipe for The Carlo speciale by means of DTD constraints, so yes this is possible.

(iv) 1 point Since the DTD exactly captures this pizza recipe, then we can use an XML validator to check that this particular pizza order is correct.

(v) 3 points

If in general we think of a recipe as containing a finite number of portions of a fixed number of ingredients, then yes we can use DTD constraints to capture a recipe. Also the Boolean constraints between ingredients such as *and*, *or* and *not* can be captured by DTD constraints, comma, or (“|”) and absence. However this only increases succinctness of the DTD, since in the worst case we can list out all possible ingredient combinations.