PizzaNet

Pizza Ordering Application

Kaleb Olson

Moussa Diaby

The Project Concept

CUSTOMER SELECTS WHAT PIZZA THEY WOULD LIKE TO ORDER

CUSTOMER MAY ADD OR REMOVE TOPPINGS, OR CHANGE SIZE

CUSTOMER CAN THEN ADD MORE PIZZAS OR CHECK OUT

CUSTOMER IS PROMPTED FOR PERSONAL AND DELIVERY INFORMATION

Scope

Does

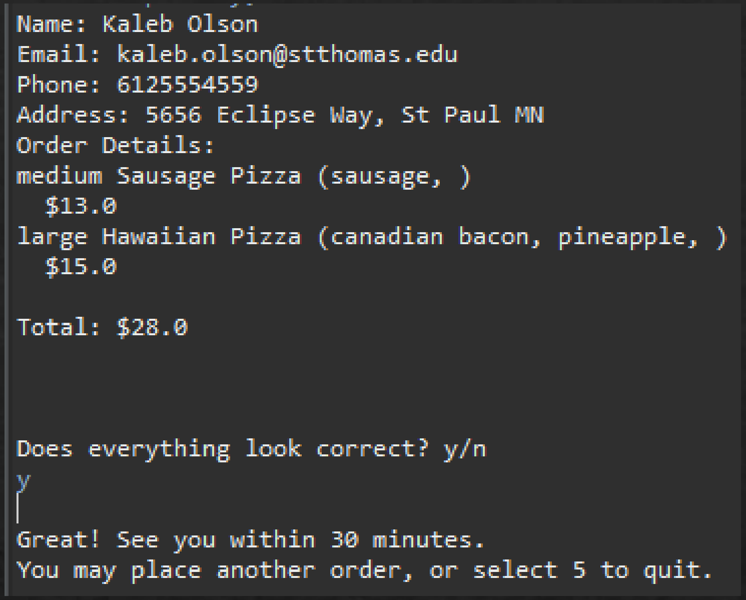
- Allow customer to create order
- Store order information
- Calculate price

Does Not

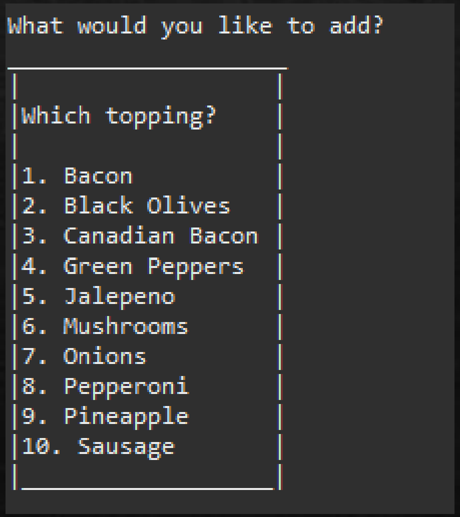
- Implement a map API
- Connect to a network
- Actually make a pizza :(

The Application

- Written in Java using Eclipse
- MVC Design Pattern
- Command Line UI
  - UI uses imported text files for menus



```
Name: Kaleb Olson
Email: kaleb.olson@stthomas.edu
Phone: 6125554559
Address: 5656 Eclipse Way, St Paul MN
Order Details:
medium Sausage Pizza (sausage, )
  $13.0
large Hawaiian Pizza (canadian bacon, pineapple, )
  $15.0

Total: $28.0



Does everything look correct? y/n
y

Great! See you within 30 minutes.
You may place another order, or select 5 to quit.
```

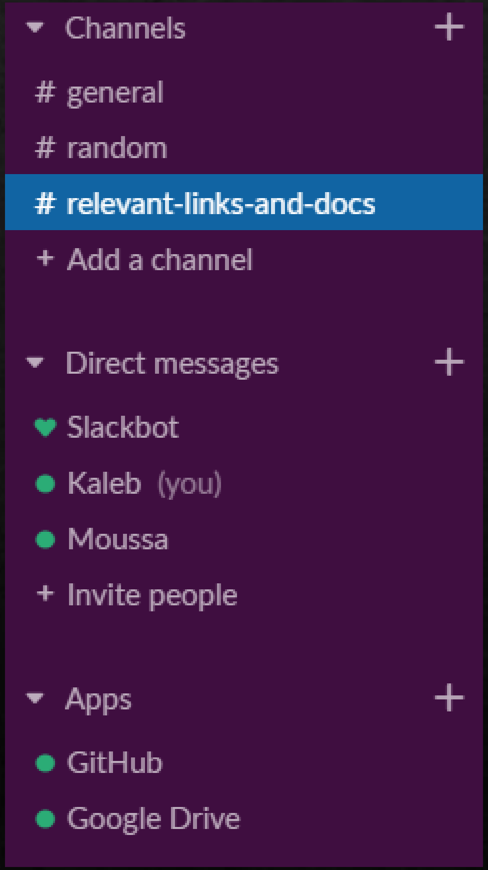


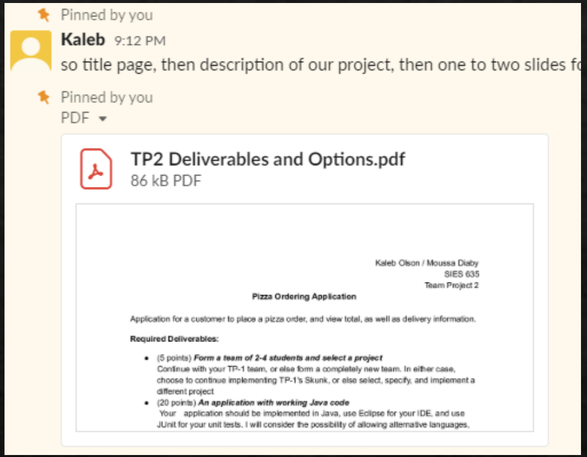
```
What would you like to add?
___________________
|                  |
|Which topping?    |
|                  |
|1. Bacon          |
|2. Black Olives   |
|3. Canadian Bacon |
|4. Green Peppers  |
|5. Jalepeno       |
|6. Mushrooms      |
|7. Onions         |
|8. Pepperoni      |
|9. Pineapple      |
|10. Sausage       |
|__________________|
```

- For this project, Moussa hosted the repository, and Kaleb forked a copy
- Github was a great way to keep track of where our project was at
- Working simultaneously on different tasks is a huge plus
- We did run into several merge conflicts along the way
- When working with a forked repo, not only do you need to pull every time you start working, but you need to ensure your fork is up to date with the primary repo.

Option 1: Slack for Team Communication

- Biggest advantages of Slack
  - Can be used on a cell phone
  - Multiple channels for multiple purposes
  - 3rd Party application integrations


Channels
# general
# random
# relevant-links-and-docs
+ Add a channel
Direct messages
Slackbot
Kaleb (you)
Moussa
+ Invite people
Apps
GitHub
Google Drive


Pinned by you
Kaleb 9:12 PM
so title page, then description of our project, then one to two slides fo
Pinned by you
PDF
TP2 Deliverables and Options.pdf
86 kB PDF

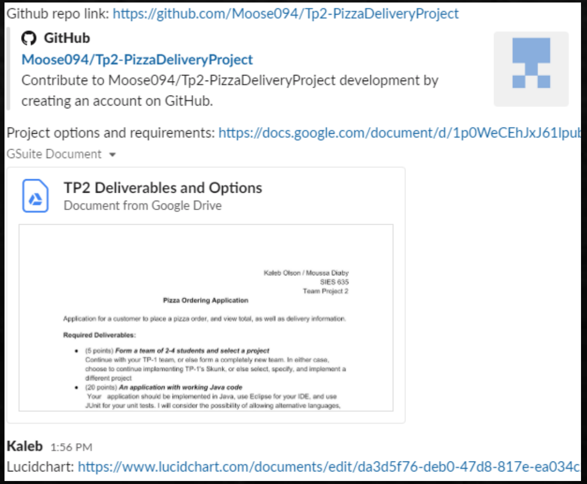
Kaleb Olson / Moussa Diaby
SIES 635
Team Project 2

Pizza Ordering Application

Application for a customer to place a pizza order, and view total, as well as delivery information.

Required Deliverables:

- (5 points) ***Form a team of 2-4 students and select a project***
  Continue with your TP-1 team, or else form a completely new team. In either case, choose to continue implementing TP-1's Skunk, or else select, specify, and implement a different project
- (20 points) ***An application with working Java code***
  Your application should be implemented in Java, use Eclipse for your IDE, and use JUnit for your unit tests. I will consider the possibility of allowing alternative languages.


Github repo link: https://github.com/Moose094/Tp2-PizzaDeliveryProject
GitHub
Moose094/Tp2-PizzaDeliveryProject
Contribute to Moose094/Tp2-PizzaDeliveryProject development by creating an account on GitHub.
Project options and requirements: https://docs.google.com/document/d/1p0WeCEhJxJ61lpub
GSuite Document
TP2 Deliverables and Options
Document from Google Drive

Kaleb Olson / Moussa Diaby
SIES 635
Team Project 2

Pizza Ordering Application

Application for a customer to place a pizza order, and view total, as well as delivery information.

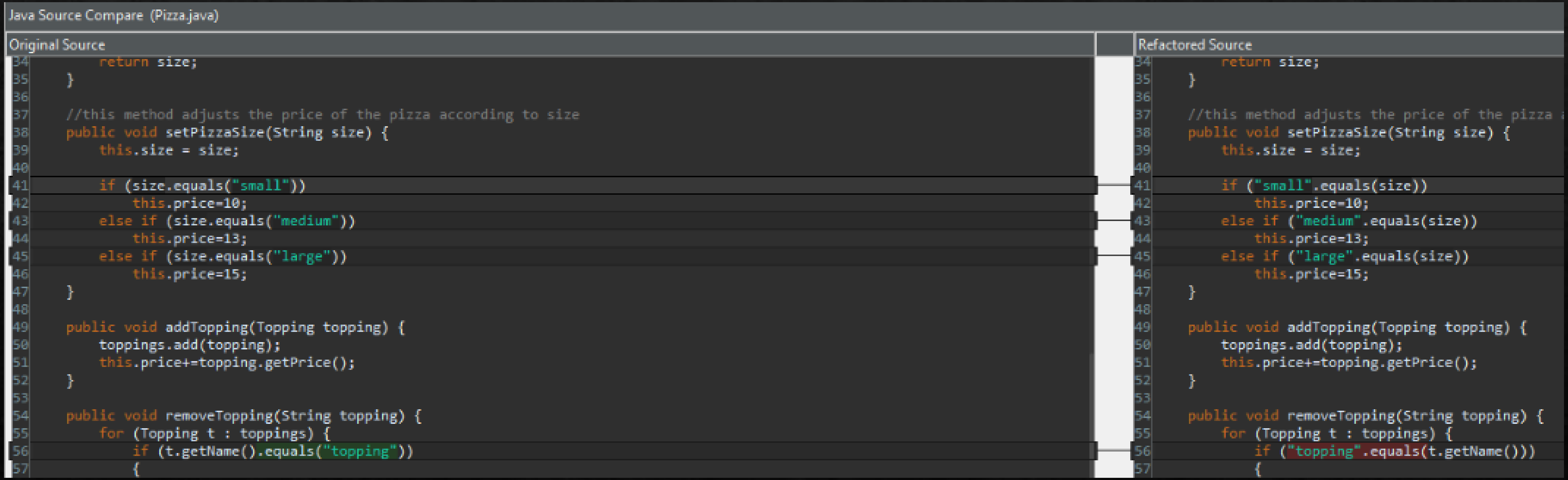
Required Deliverables:

- (5 points) ***Form a team of 2-4 students and select a project***
  Continue with your TP-1 team, or else form a completely new team. In either case, choose to continue implementing TP-1's Skunk, or else select, specify, and implement a different project
- (20 points) ***An application with working Java code***
  Your application should be implemented in Java, use Eclipse for your IDE, and use JUnit for your unit tests. I will consider the possibility of allowing alternative languages,

Kaleb 1:56 PM
Lucidchart: https://www.lucidchart.com/documents/edit/da3d5f76-deb0-47d8-817e-ea034c

Option 2: JSparrow for Refactoring

- Some great refactoring tips
- Automatic implementation
- Free version is very limiting



Java Source Compare (Pizza.java)

Original Source

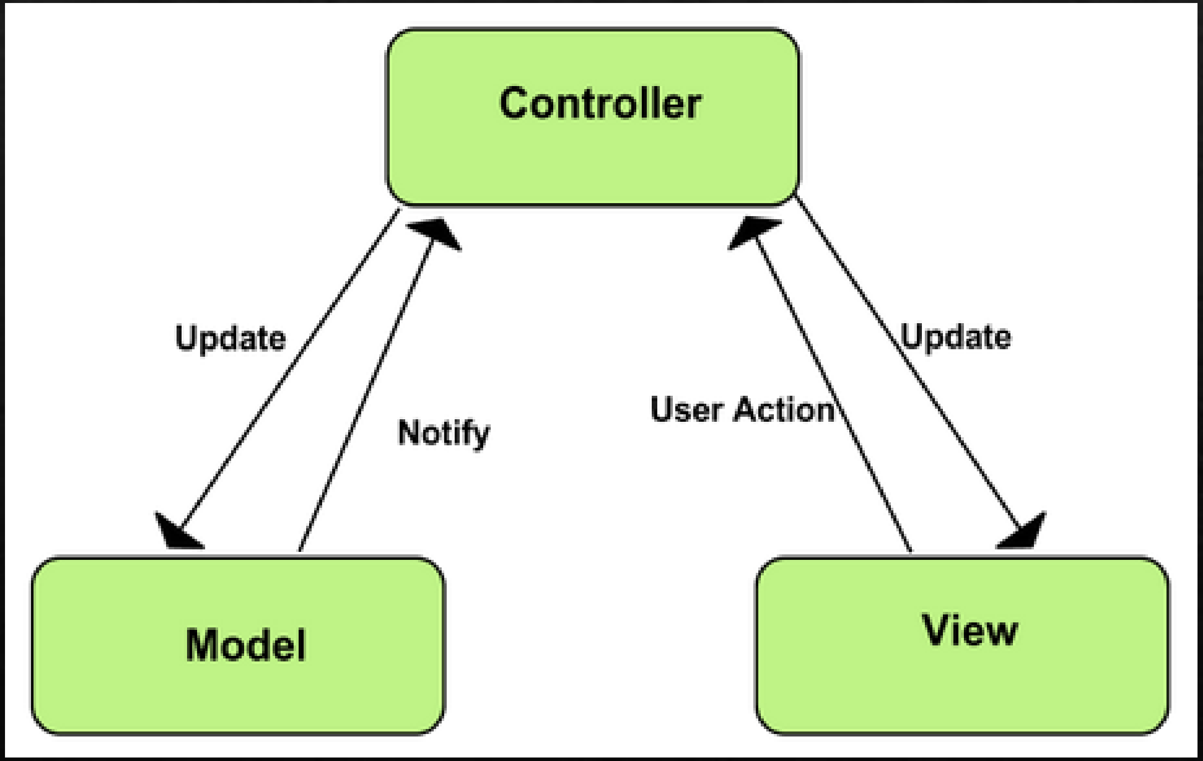
```
        return size;
    }

    //this method adjusts the price of the pizza according to size
    public void setPizzaSize(String size) {
        this.size = size;

        if (size.equals("small"))
            this.price=10;
        else if (size.equals("medium"))
            this.price=13;
        else if (size.equals("large"))
            this.price=15;
    }

    public void addTopping(Topping topping) {
        toppings.add(topping);
        this.price+=topping.getPrice();
    }

    public void removeTopping(String topping) {
        for (Topping t : toppings) {
            if (t.getName().equals("topping"))
            {
```

Refactored Source

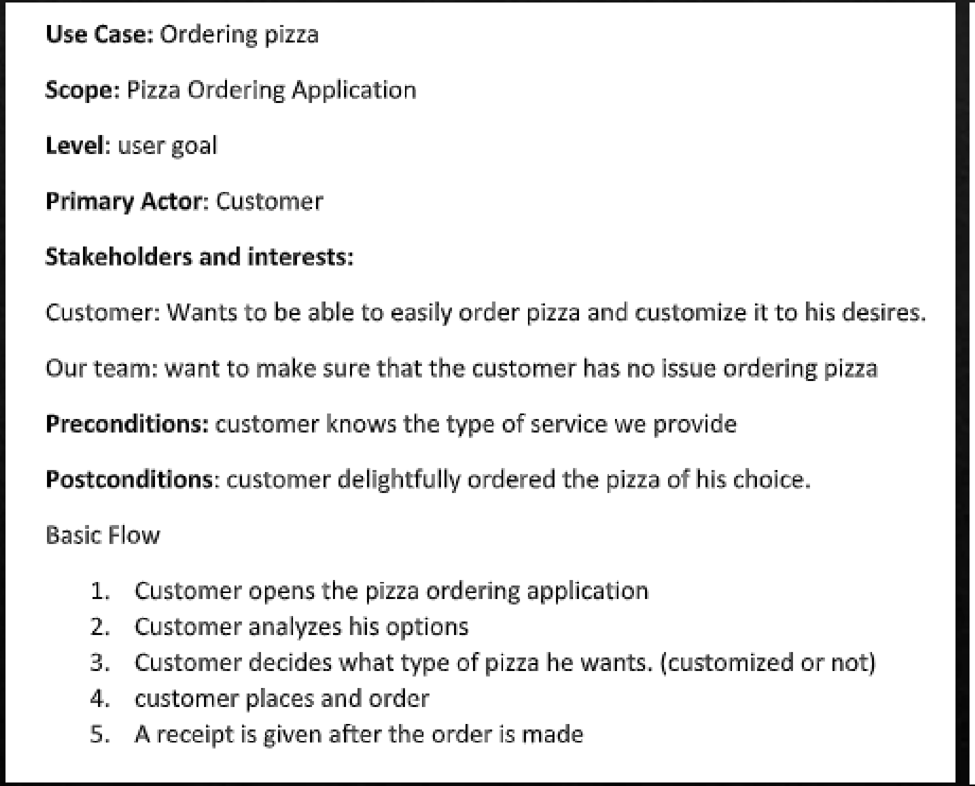
```
        return size;
    }

    //this method adjusts the price of the pizza a
    public void setPizzaSize(String size) {
        this.size = size;

        if ("small".equals(size))
            this.price=10;
        else if ("medium".equals(size))
            this.price=13;
        else if ("large".equals(size))
            this.price=15;
    }

    public void addTopping(Topping topping) {
        toppings.add(topping);
        this.price+=topping.getPrice();
    }

    public void removeTopping(String topping) {
        for (Topping t : toppings) {
            if ("topping".equals(t.getName()))
            {
```

Option 3: Superior Separation of Domain and UI layers

- Application follows Model-View-Controller pattern
  - M: Domain was written initially
  - C: Controller class written next to execute domain layer functions
  - V: UI instantiates and implements controller class (referred to as "app")



Option 4: Use Case



Use Case: Ordering pizza

Scope: Pizza Ordering Application

Level: user goal

Primary Actor: Customer

Stakeholders and interests:

Customer: Wants to be able to easily order pizza and customize it to his desires.

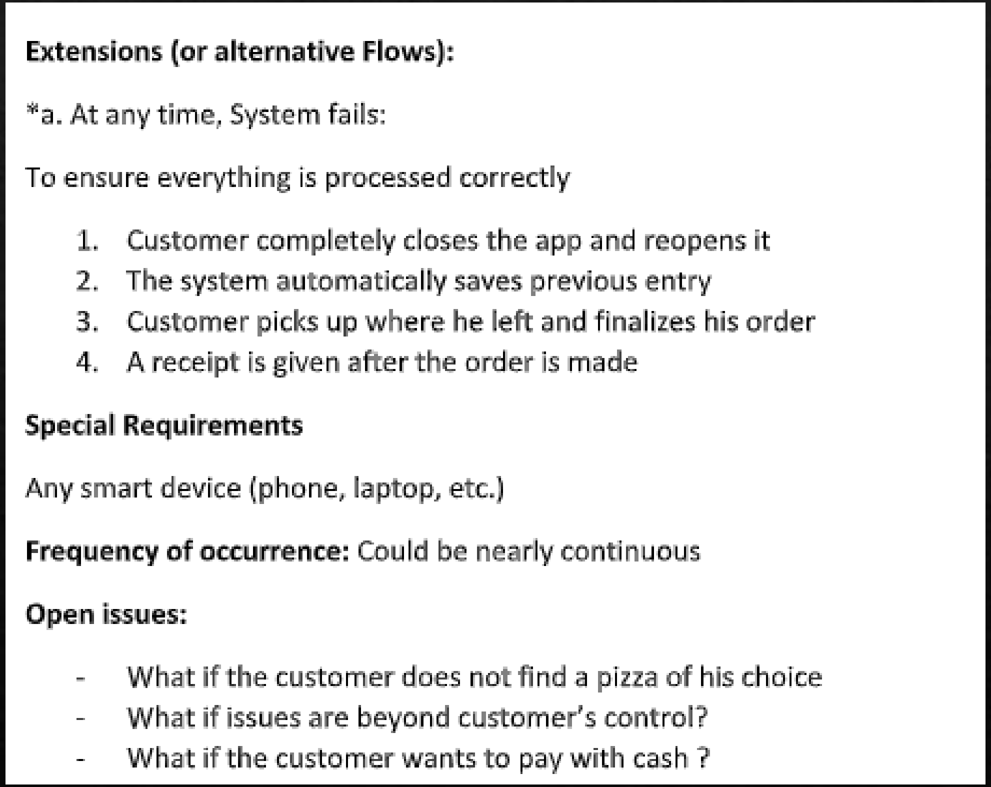
Our team: want to make sure that the customer has no issue ordering pizza

Preconditions: customer knows the type of service we provide

Postconditions: customer delightfully ordered the pizza of his choice.

Basic Flow

1. Customer opens the pizza ordering application
2. Customer analyzes his options
3. Customer decides what type of pizza he wants. (customized or not)
4. customer places and order
5. A receipt is given after the order is made



Extensions (or alternative Flows):

*a. At any time, System fails:

To ensure everything is processed correctly

1. Customer completely closes the app and reopens it
2. The system automatically saves previous entry
3. Customer picks up where he left and finalizes his order
4. A receipt is given after the order is made

Special Requirements

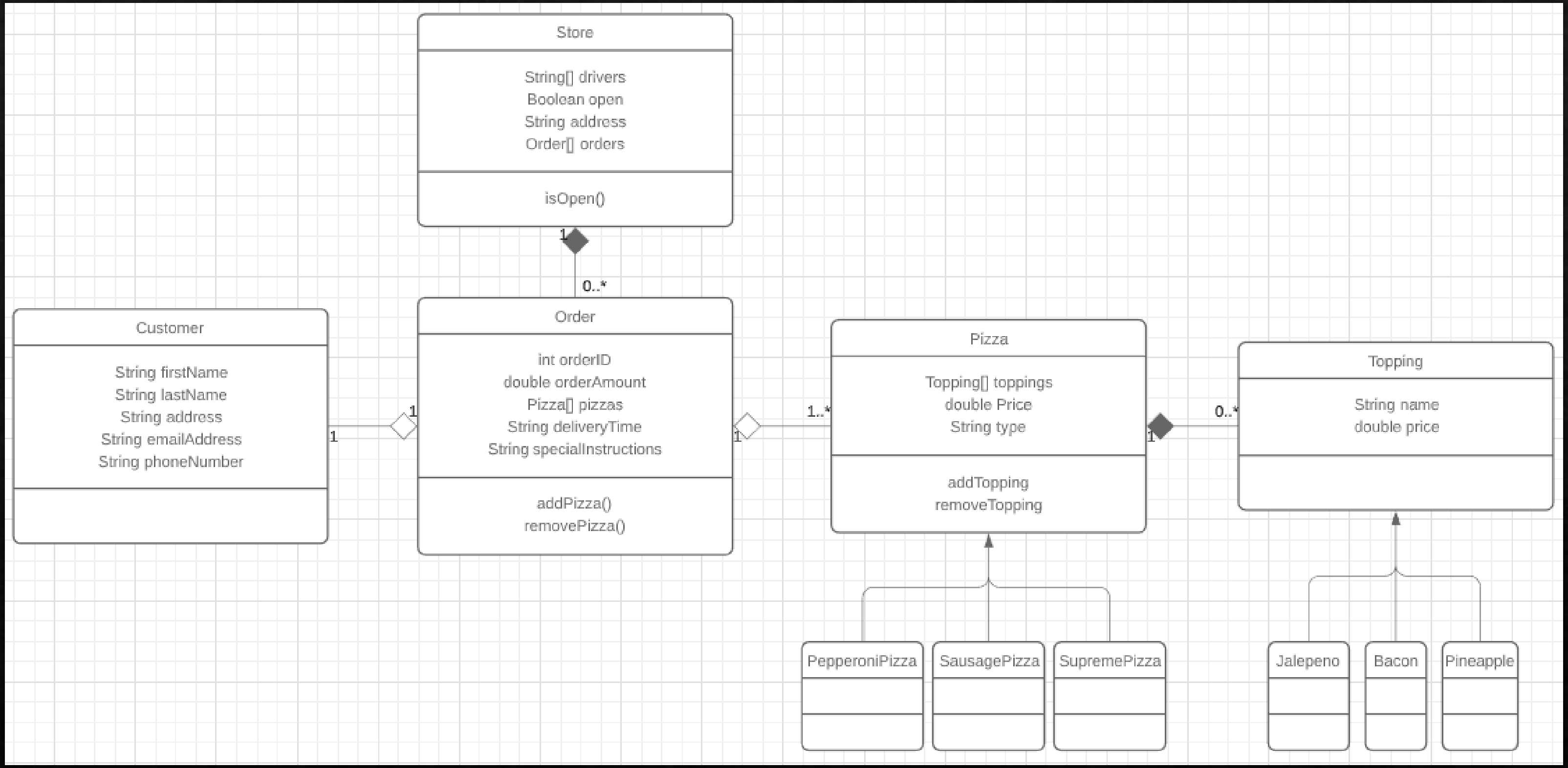
Any smart device (phone, laptop, etc.)

Frequency of occurrence: Could be nearly continuous

Open issues:

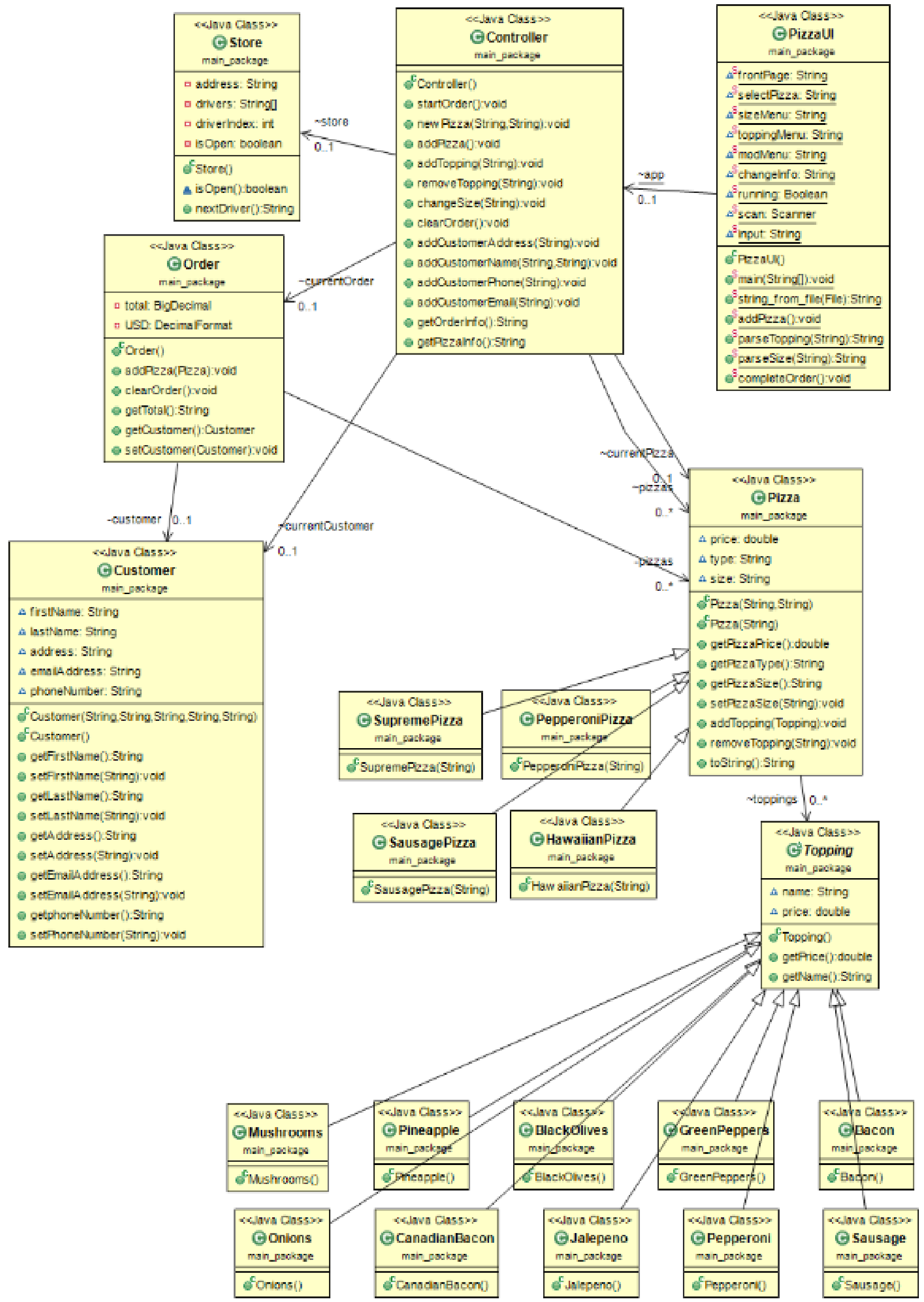
- What if the customer does not find a pizza of his choice
- What if issues are beyond customer's control?
- What if the customer wants to pay with cash ?

Option 5: Domain Class Diagram



Reverse Engineered DoCD

- Here's how it actually turned out!



Code Demonstration!