Name:_____ Section: _____CM: _____

CSSE 220---Object-Oriented Software Development

Final -- Refactoring, February 2021

Allowed Resources on Refactoring Part. Open book, open notes, and computer. Limited network access. You may use the network only to access your own files, the course Moodle and the course web pages, the textbook's site, Oracle's Java website, and Logan Library's online books. You may only use a search engine (like Google) to search within Oracle's Java website - all others uses or accessing websites other than those mentioned above are not allowed.

Instructions. You must disable Microsoft Lync, IM, email, and other such communication programs before beginning the graphics part of the exam. Any communication with anyone other than the instructor or a TA during the exam may result in a failing grade for the course.

You must actually get these problems working on your computer. Almost all of the credit for the problems will be for code that actually works. If you get every part working, comments are not required. If you do not get a method to work, comments may help me to understand enough so that you can earn (possibly a small amount of) partial credit.

Submit all modified files via Moodle.

Problem Descriptions

Refactoring Problem (Pass/No-Pass)

You are given a working solution (Figure 1, to the right) to a Java program written for a local pizza store to handle orders of its 3 types of pizzas: *cheese, clam, pepperoni*. To see the output run Java Application *PizzaStoreDriver.java.* A copy of the output appears at the end of this document in Listing 1.

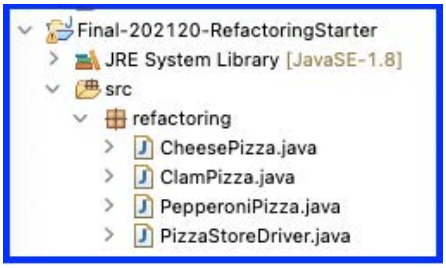

Figure 1

Unfortunately, this design and implementation has an unacceptable amount of code duplication some of which can be determined from its UML diagram in Figure 2 (below).

Your job is to use either (1) *inheritance with a class* or (2) *inheritance with an abstract class* to remove as much code duplication as you from *PizzaStoreDriver.java* and also in the 3 pizza classes (*CheesePizza.java*, *ClamPizza.java*, *PepperoniPizza.java*).

Make any other small changes needed so that the output from your refactored implementation looks identical to what is shown in Listing 1 (below).

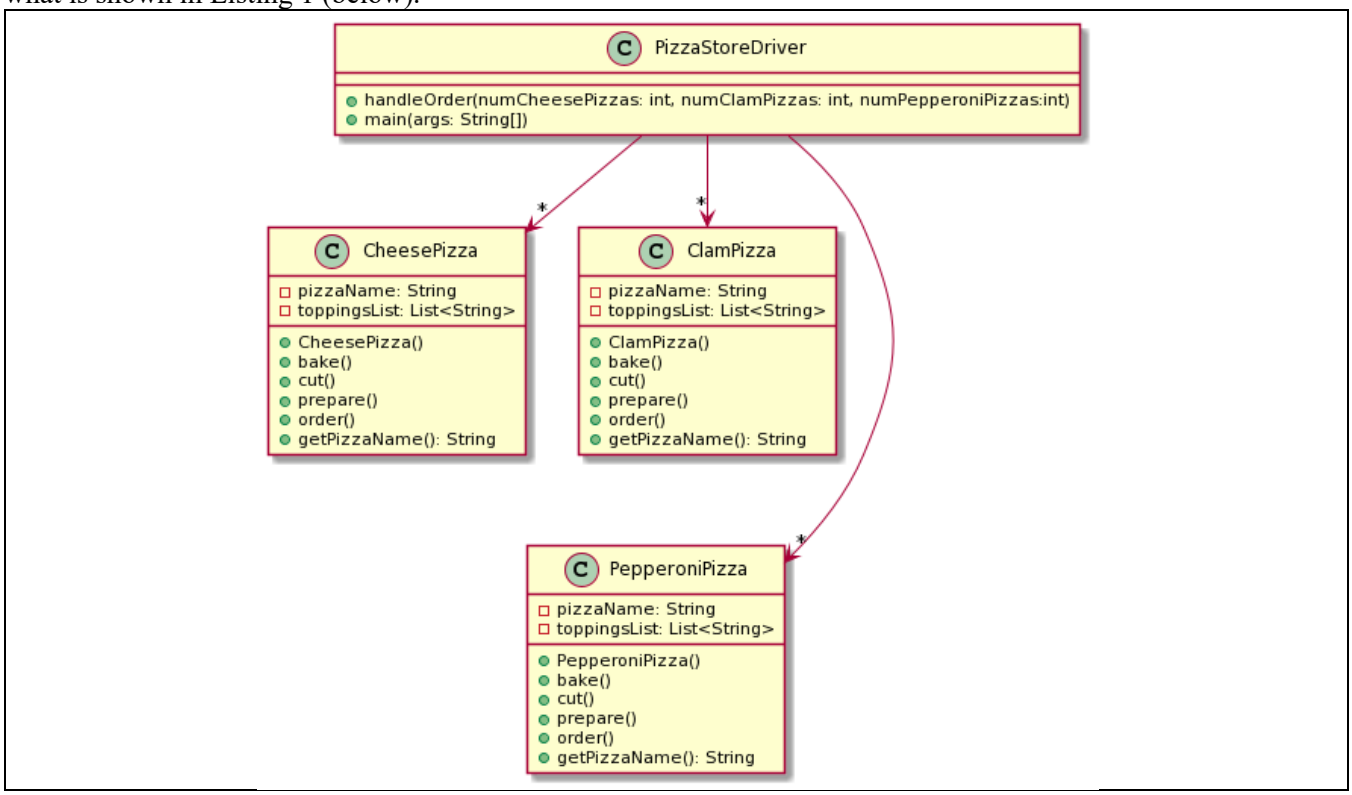

Figure 2 – UML for Starter Code

Part 1

- Refactor the 3 pizza classes (*CheesePizza*, *ClamPizza*, and *PepperoniPizza*)
  Note: do not use a Java *Interface*, that will result in a **No Pass** for this part

Part 2

- Utilize the refactoring done in Part 1 in *PizzaStoreDriver*

Evaluation

- You must pass both Parts 1 and 2

```
=====================================================
==================== Making Pizzas ==================
=====================================================
Preparing pizza: Cheese
Baking pizza: Pepperoni
Cutting pizza: Pepperoni
Boxing pizza: Pepperoni
Preparing pizza: Cheese
Baking pizza: Pepperoni
Cutting pizza: Pepperoni
Boxing pizza: Pepperoni
Preparing pizza: Clam
Baking pizza: Pepperoni
Cutting pizza: Pepperoni
Boxing pizza: Pepperoni
Preparing pizza: Pepperoni
Baking pizza: Pepperoni
Cutting pizza: Pepperoni
Boxing pizza: Pepperoni
Preparing pizza: Pepperoni
Baking pizza: Pepperoni
Cutting pizza: Pepperoni
Boxing pizza: Pepperoni
=====================================================
==================== Pizzas Made ====================
=====================================================
```

Listing 1 – Console Output from *PizzaStoreDriver.java*