Vrinda Subhash

Week 6/7 Design Question:

I think Design Number Two is the most consistent with the SOLID principles. That design respects the SOLID principles because it uses the Liskov Substitution Principle since the StandardDrink class is a subclass of the BubbleTea class, which means wherever a BubbleTea object is, you could substitute it with a StandardDrink object and it should still work. That works for the specification since it was stated that there are preset bubble tea drinks that a user could choose. It also applies the Dependency Inversion Principle, since BubbleTea depends on the Item Interface, not the implementations of the Item (Ingredient and Topping); it depends on the abstraction and not concrete classes. It also uses the Interface Segregation principle since the Item interface is what is implemented by the Topping and Ingredient classes since those are the things that will add to the price of a BubbleTea in addition to the base price for the TeaFlavor, which is a separate class (as the specification explains). The builder takes a TeaFlavour, and then adds the customizations which are the Items (Topping, Ingredient). One way I think that the design could maybe better adhere to the SOLID principles is the relation between the Builder and BubbleTea class, to better follow the Dependency Inversion Principle they should depend on abstractions not concrete, so if the BubbleTea Class changes, it will affect the Builder class. One way to better satisfy the specification could be to let the BubbleTea class allow a list of teaFlavours since the specification says a drink “can contain a variety of tea flavors”, so one BubbleTea should be able to have many TeaFlavours. This would also mean updating the Builder class.