For whoever didn’t get a chance to do this

1. Add a getTotal() method in Order that calculates the total cost of the order, looping through the Item array and getting their prices and adding them together.
2. Add this total into the Order class’s print method
3. Add a field/instance (variable) to the Item class to indicate if the item is taxable (boolean)
4. Add a method to Order that calculates the tax on the order

HINT: Add a static variable for tax rate, based on which item is taxable calculate the tax. (for-loop the items, check their Taxable property, if yes, multiply by tax rate and add to tax total, return total tax)

1. Change our getTotal() method in Order to include the tax. Maybe create a getSubTotal() to do what the getTotal() method did previously.
2. Fix the print methods in Item and Order to reflect this new taxable field/variable

-----------This we will do in class, only do if you want to-------------

1. In class, we wrote code that created “items” and added them to an order. In a more realistic program, there would likely be a fixed set of items actually available to order (you can’t order milk from a shoe store). We will do this by creating an “ItemDatabase” class which will maintain the list of available items (no we aren’t really up to databases yet, this class will be a mock database and instead will store the items in an array). When orders are created we will look up the items (by name or id) in the ItemDatabase and add them to the order. (In future we will learn ways to prevent any other items from being created other than those we create in the ItemDatabase – and ways to ensure that only a single ItemDatabase is created)
   1. First add an int field “ID” to the item class. It should be private and should only have a public get – no one should ever change the ID of an item after creation so we will not provide a public set property (you can add a private one if you like)
   2. Assign each item that is created an ID. You can either pass the ID into the constructor (and make sure to pass a different one each time) or a better approach might be to have a static int in the Items class keeping track of next available ID number and assign that ID automatically in the constructor.
   3. Write a new class ItemDatabase. This class should have an array of items (maybe 100 long). In its constructor, create a bunch of items and add them to the that item array.
   4. Write a method for this ItemDatabase class that allows you to get an item based on its ID
   5. Replace the code in the “OrderForm” class that creates the orders to create and use an instance of an ItemDatabase to find items rather than creating random items as it currently does. Have it get items using the method you created above to get items by id