Exercise 3

- In the app that you modified in Exercise 2, create a new model class in the ordering folder for order header
- Create a new model class in the ordering folder for order details
- Add properties to the order header class for id, order number, description, total, and collection of order detail objects
- Add properties to the order detail class for id, order header id, product number, quantity, total
- In the order header component, create a new instance of order header, set default property values, and bind to order number and description property for display
- In the order detail component, create a new instance of order detail, set default property values, and bind to product number, quantity, and total for display
- In the order detail component, set up an input with two-way binding using ngModel for the quantity
- Confirm that changes to the input value are reflected real-time in the quantity display

Component Communication

Develop Intelligence

Exercise 4

- Start with the app that you modified in Exercise 3
- Add a new @Input() property for orderHeader in order header component
- Add a new @Output() event to order header component for shipped
- Add a new @Input() property for orderDetail in order detail component
- In the order component, create a new instance of order header and order detail, and set default property values
- Pass as @Input()'s from order component to the child components
- Verify that existing display and two-way binding continue to operate as expected
- Add a button to order header component that raises shipped event – emit the order header object as part of event payload (currently there will be no changes since component is view only but positions us for when we can modify)
- Handle event in order component and console.log order header details provided to the event in the payload

Component Change Detection

Develop Intelligence

Exercise 5

- In a new Angular app, create a parent component and a child component
- In the parent, include public properties for an array and an object (properties of your choosing)
- Initialize the properties in the parent component (you choose the values)
- In the child, include input bindings for the array and the object
- In the child view, display each array element as a <li> within a
- In the child view, use a <p> element to display the object properties in JSON format
- In the parent, include an instance of the child and pass the properties as inputs
- Create a button in the parent component that adds a new element to the array
- Create a button in the parent component that modifies one or more properties of the object
- Test using ng serve --open to confirm that changes are reflected when the button is pushed in the parent
- Change the change detection strategy for the child component to “OnPush”
- See what happens when attempting to press the button in the parent (for updates)
- Can you make the necessary fixes to approach to pick up changes (leaving OnPush in place)?