Forms and Two-way Binding Lab

An order is ready to ship when all of the items have been picked. So the "Mark as shipped" button should not be clickable until all the order lines are picked. We're going to disable the button until every line is marked as picked.

1. Edit ship-order.component.ts. Create a method called isReadyToShip(order). Note: It should receive an order object.

```
isReadyToShip(order: Order) {
  return order.lines.every((ol:OrderLine) => ol.picked)
}
```

2. Now edit ship-order.component.html. Find the button and do an Angular property binding. Bind the "disabled" property to the return value of the isReadyToShip(order) function. (Hint: use square brackets. Another hint: The "!" character makes a true false and a false true).
3. Run and test. Since none of the lines are picked yet, you'll see that the button is disabled.
4. Now edit ship-order.component.html. Find your "Got it" checkbox. Using banana-in-a-box, 2-way bind it to the order line's "picked" property.
5. Run and test. When you've marked every line as picked, the "Mark as shipped" button should enable. (HInt: if you get errors, remember that you have to import the FormsModule).

Receive Component

Let's move to the ReceiveProduct component. When the user enters a package tracking number, we want it bound back to the business class.

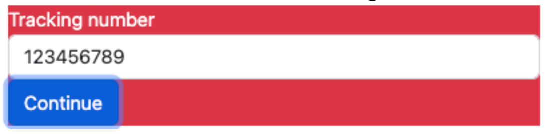
We already have an <input> in the html template for the tracking number. And we have a property in the TypeScript class for trackingNumber. We just need to bind them together.

6. Edit receive-product.component.html. Put a two-way binding on the <input> box to bind it to this.trackingNumber.
7. In the Events Lab you make the button click call a saveTrackingNumber() function. Now, have that function console.log() the tracking number just entered.

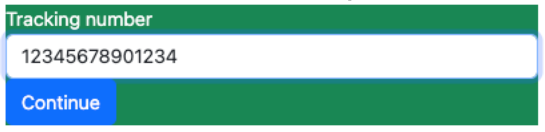
Now notice that there's a pattern attribute on the tracking number input box. If the entry matches the pattern, it is valid. We want the user to get immediate feedback if it is valid or not. You're going to add a Bootstrap class of text-bg-success if the number is valid and text-bg-danger if it is invalid.

8. Add a template reference on the <input> like this #tn="ngModel"
9. Add an [ngClass] directive to the <section> where the user enters a tracking number. If the tracking number is a valid format, make the background of the section turn green by applying the text-bg-success class. If it is invalid, apply the text-bg-danger class. It should look like this:

With an invalid tracking number



With a valid tracking number



10. Run and test with these valid numbers

FedEx	12345678901234
UPS	1Z1234590291980793
USPS	9400 1057 3346 4567 2475 01
USPS	9205 5000 3865 8954 7543 00

11. Bonus!! Add tn.touched to your [ngClass] directives to say to only add the classes after the user has had an opportunity to make a change.

Recording the items received

Let's allow the user to receive items. We want to read in the quantity and product ID and add it to the array of products being received when he/she hits the "Receive Product" button.

12. Add a public array called receivedProducts to the component class. This will be an array to keep track of all the products and quantities that are being received. Add a Product to be received, a quantity and productName string.

```
receivedProducts: Array<Product & { quantity?: number }> = new Array();
productName?: string;
quantity?: number;
currentProduct?: Product;
```

13. Notice that you have an <input> box for the productName being received. Bind that to productName.
14. There's a button that says "A placeholder product" Change it from this:

```
<button class="list-group-item">
  A placeholder product
</button>
```

15. to this:

```
<button (click)="currentProduct={name:productName}" class="list-group-item">
A placeholder product
</button>
```

16. There's another input box for the quantity. Bind that to quantity.
17. When the user hits the receive product button you're calling the receiveProduct() method. Modify the method to add the currentProduct and quantity to the receivedProducts array.
18. In the starter HTML you were given, there's a hardcoded <table> of all the items being received. Convert the <tr> to an *ngFor, iterating receivedProducts and show all the product IDs, names, and quantities in a table. Hint: Here's how two of the table cells might look

```
<td><img [src]="'assets/images/productImages/' + product.id + '.jpg'"
         [alt]="product.name" ></td>
<td>{{ product.id }}</td>
```

19. Run and test. Enter a fake product name into the productName <input>. Then click the "placeholder product" button. Then enter a quantity. And click the "Receive product" button. You should see your new product/quantity added to the table. Cool, right?

20. Bonus!! Note that after you add your product to the array you still have values in the product name and quantity textboxes. Can you make the values clear? Go ahead and do that.