# TypeScript Lab

## Making our business classes

We're going to be working with products, orders, and customers so it might be safer to create business classes to define the shape of the objects we'll be working with. We'll start by putting them in a shared folder.

1. Create a new folder under *src/app* called the *shared* folder.
2. In it, create five new files called customer.ts, location.ts, order.ts, orderLine.ts, and product.ts.
3. These will be our business classes. In each create one class with the following properties:

**Customer**

| | |
|---|---|
| customerID | number |
| givenName | string |
| familyName | string |
| companyName | string |
| address | string |
| city | string |
| region | string |
| postalCode | string |
| country | string |
| phone | string |
| email | string |
| imageUrl | string |
| password | string |

**Order**

| | |
|---|---|
| orderID | number |
| customerID | number |
| status | number |
| orderDate | Date |
| shipVia | number |
| shipping | number |
| tax | number |
| shipName | string |
| shipAddress | string |
| shipCity | string |
| shipRegion | string |
| shipPostalCode | string |
| shipCountry | string |
| lines | Array<OrderLine> |
| customer | Customer |

**Product**

| | |
|---|---|
| productID | number |
| name | string |
| description | string |
| price | number |
| imageUrl | string |
| featured | boolean |

**OrderLine**

| | |
|---|---|
| quantity | number |
| productID | number |
| locationID | string |
| price | number |
| picked | boolean |
| product | Product |
| location | Location |

**Location**

| | |
|---|---|
| locationID | string |
| description | string |
| productID | number |
| quantity | number |
| product | Product |

4. Note that the order class uses the OrderLine class and the Customer class. Don't forget that you'll need to import these classes into the Order class for this to work. This is true for the Location and OrderLine classes as well.
5. Now open a command window and compile your new classes and all of the site by typing in `ng build`
6. You'll probably see some compile errors. If so, go ahead and correct them.

## Using a class in a component

7. Edit shipping/ship-order.component.ts. Give the class a private property called *order* that is of type Order. (Yes, the one you just created). Don't forget you'll need to import it at the top.*

* We're only going to remind you to add imports statements occasionally from now on so don't forget going forward.

8. In the ngOnInit() method, instantiate that order and set the properties to fake values of your choice. You'll want to create a couple of fake order lines. (Hint: this.order.lines = [], then this.order.lines.push(yourNewLine1))
9. Double-check each line and make sure it has a product. Make sure that product has an imageUrl that looks like this:"/assets/images/productImages/7.jpg".

We do want you to practice with creating properties and seeing the tedium of getting them just right. It's more rigid but more controlled when you use strongly typed objects. You have to balance the tradeoffs of using strongly typed classes vs. JavaScript's dynamic objects.

10. Hopefully you've read far enough ahead to see this. We've provided a code snippet to fill in a fake order so you don't have to type in every single property. Go look in /setup/assets/codeSnippets/anOrderReadyToShip.js. Copy its contents into ship-order.component.ts if you like.
11. Build the project again.
12. Find the dist directory and look in there. You should see a main.js file. This is the file that will be served once we go live.
13. Go ahead and edit main.js. You and your partner look around in there. Do you see your component in there? _____ How about your classes? Are they there? _____ Discuss with some of the students around you why this makes sense.