An introduction to OOP via a Drupal 8 lens

How does OOP work?

--- Object-Oriented Programming (OOP) is a programming model that uses instances of classes to structure information and functionality ---

Interface
- Define what objects will be able to do
- NOT how they do it
- Has no state
- Can extend other interfaces

Class
- Can be instantiated into an object
- **'implements'** an interface
- Can **'extend'** other classes
- Can be **'abstract'** for use as a **blueprint**, stops them being instantiated

Method
- A **function** of a class

Property
- A **variable** of a class
- Sometimes referred to as 'field' or 'attribute'

Object
- An instance of a class

http://searchsoa.techtarget.com/definition/object-oriented-programming

What is a Drupal entity?

--- A Drupal entity is a **blueprint** for data storage (usually content) ---

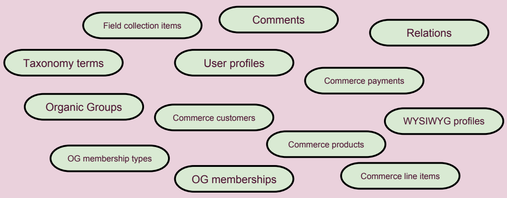
Base

I/O with the DB

Some optional extras

- Theming the content
- Views integration
- Admin UI
- Features integration
- Fieldable bundles
- Destinaton URLs

Remember, nodes are not the only entity types on a Drupal system



https://api.drupal.org/api/drupal/includes%21entity.inc/7

What is a node?

--- A node is an implementation of an entity ---

That said, some important elements are not part of the nodes OOP architecture, or old D7 style procedural files
- The info file `node/node.info.yml`
- Permissions `node/node.permissions.yml`
- Routing (paths) `node/node.routing.yml`

… plus lots more I have not delved into myself yet

https://api.drupal.org/api/drupal/core!modules!node!src!Entity!Node.php/class/Node/8

What creates a node

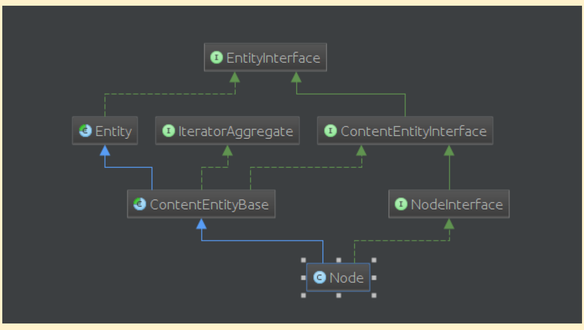
--- This is very complicated due to the inheritance tree, but here is an example ---

- node.module contains the line of code `Drupal\node\Entity\Node` which relates to the class file at `node/src/Entity/Node.php`

Classes
- This class extends one class
  - ContentEntityBase - This **abstract class** is for handling content entities (name gives that away)
- Which, in turn, extends a class itself
  - Entity - This **abstract class** is the base we mentioned earlier

Interfaces
- Node implements a single **interface**
  - NodeInterface - Sets up our nodes blueprints

There are lots of other interfaces extended throughout from this point, too many to go through in detail here. It's mainly done to use more modular interface inheritance.



Note: Diagram above was created by finding the node class in PHPStorm, and right-click to get the diagram. Once done, you can display all methods/fields etc… to get LOTS more info

How do I create my own entity type in D8?

--- The examples module is always the best starting point ---

Just download the examples module, and look at the content_entity_example sub module.

It is creating a 'contact' entity, with paths, permissions, admin UI, and some general form classes.

https://www.drupal.org/project/examples

Reuseable functions across entities?

--- The magic word is 'traits' ---

Adding this next to your class will allow it to use the functionality of that trait. For example fields and many other elements use the below trait for translation

```
use Drupal\Core\StringTranslation\TranslationWrapper;
```

If you want to find and add trait, I think your best bet is:
- Check if your inheritance gets it for free (your IDE may be able to help with this)
- If not, use the below link to find out whan you need to 'use' to get the functionality you require
  eg. search for the name of 'Translation' and description containing' string' on the form

https://api.drupal.org/api/drupal/classes/8