LIS-5364

Object Oriented Programming with PHP

Object-Oriented?

- Not quick
- Not efficient
- Not “pure”
- Not designed for “computers” but for people: Try to map “objects” to the real world

Basic advantages

- Reusable
- Easy to read, understand, and modify
- Clear and real-life oriented
- Mimics the way humans think about things, not as 01011011, but as “dogs bark, cats meow, but they're all furry”
- More work explaining and planning on the front end → less work for addition/maintenance

Abstraction

- Theoretically seamlessly convert real life concepts into computer data objects.

  “Set and forget”

Encapsulation

- Represent and use essential features without necessarily knowing exactly how they work;
- Only allow interactions with “objects,” not the data on the machine. (prevents programmers from tampering with things they shouldn't)

Inheritance

Hierarchically store objects to create "taxonomies."

Classes can have "children" that inherit everything from the parent AND add more

Polymorphism

- The ability to implement and use code that calls for an action or a characteristic, and yet does not require that action/characteristic to be the same every time.

  “I don't care HOW, just get it done”

Classes

- A "general" noun
- A "blueprint"
- An "ideal/theoretical form"
- An "object factory"

(to the computer, an invented data type. Like a string or float or array)

Object (to the human)

- A Proper Noun
- An instance of a class
- One of many possible
- A copy of the "Thing"

(to the computer, a specific piece of data, like a "string" (string) or an array [0,5,19])

Properties (or attributes)

- Adjectives applicable to the class/object
- Characteristics or
- Conditions

(to the computer, variables associated with the class/object)

Methods

- Verbs!
- Anything the object can do
- Anything the object can have done to it

(to the computer, any FUNCTION associated with a class/object)

Now, to make things way more complicated:

When thinking about class - “noun” may be too narrow. Technically, ANYTHING might make a good object. Even something like an action:

What's in a class?

Properties and functions which can be:

Public: accessible by everyone

Protected: Accessible only inside the class and any extending classes

Private: ONLY accessible inside the class

What's in a class?

Properties and functions, mostly:

To take advantage of OOP, let's generally make our properties "protected" (mostly private)

And our functions "public" (so that we can access them as programmers)

Inheritance!

- You may make a new class by extending an old one.

  But note, you can only have ONE parent (in php). Choose wisely.

  e.g. Class Bear extends Animal{

Abstract Classes

Any shared data across a lot of possible items

- Abstract classes cannot be instantiated themselves, therefore they're USELESS unless EXTENDED:

```
abstract class fsuperson {

protected $fsuid =''
}
```

- abstract class Fsuperson {

  protected $fsuid =''

  public function getfsuId()...
  }

Interfaces: Function Prototype

- Merely lists methods that a class MUST implement.
- Why not just use a method? Because an **interface** completes similar actions in different ways

```
Interface canUseClassroomComputer {
public function getComputerAccess();
}
```

```
Class professor extends Fsuperson
implements canUseClassComputer{
//the below MUST be defined
public function getComputerAccess(){
fillOutStupidPointlessForm
}
```