Programmer Thinking = Object Oriented Programming, Quick Guide

Dr. William L Honig, Loyola University Chicago

Three key things to do for basic OOP; a brief OOP design / implementation process. Often you cannot do these in strict 1,2,3 order, but work back and forth between them all until you "see the footsteps for the programmer to follow". Know how you will implement a few key use cases. Much of this can be done with UML class diagram before writing any code!

A. Pick and Name the Class(es)

A class is a general, generic, common way to represent all possible instances / variables / objects of that class. Class names are singular nouns, InitialCapitalCamelCase.

- Encapsulation. The class writer controls the external view and decides what to allow users of the class to do.
- A class should be a "single responsibility". When in doubt make separate classes that work together.
- "You cannot trip over a class" (only an actual instance of one is real).
- To use you OOP class elsewhere use the ClassName – it works same as other reference types.

B. Select Private Data Attributes

What data, information, details, are needed to represent any and all possible instances of the class. Give each a suitable type (can be basic or reference types including other classes you define). Data attributes are named as variables and usually nouns. All data attributes should be private.

- The set of data in any instance of the class makes the "state" of the object. The class needs to ensure that the state is always correct.
- Only this data exists outside of method calls of the class. If something is important to how the class works or stores information, it must be in one or more of the class data attributes.
- Do not initialize data attributes at the class level (don't use the = when declaring them).
- Use static only for CONSTANTS (at least while a newbie) or for the very special case of a data attribute shared by all objects.

C. Define Public Methods or Operations

What does the class need to do for its users? Each behavior, function, method, operation is a single public method of the class. Methods are usually verbs to show action. It's good to completely write the signature lines for each key method (and brief pseudo code on how it affects the object or what it does) before writing the body code for any method in the class.

- Methods can be accessors (return information or calculate it from the current state), mutators (change the current state), or constructors (set up a new variable of the class). Methods are often overloaded (same method name used more than once).

- Only the signatures of public methods are known to the users of the class. What happens inside the method is up to the class programmer (provides encapsulation).

- Include at least one constructor. Often there are multiple constructors and a "default" constructor (no parameters). Every constructor must set up the full state of the object. If a constructor identifies any error you raise an exception (or simplier to get started S.o.p an error message, or force the program to end).

- Each method should be a few lines of code. You can include helper methods if you want to break things up. Helper methods can be private unless you also wish users of the class to be able to use them.

- You do not need to always have a get and set method for each data attribute; do so only when it makes sense and support the encapsulation you want to maintain.

- Unless good reason, all classes should have toString, equals, compareTo (and eventually hashCode) methods. Learn the correct signature for each.

- All methods must leave the object in a correct state. (More advanced: each method has pre and post conditions; if the pre condition(s) are met at time of call, the method guarantees the post condition(s) will be met when it returns.) All constructors must set up complete initial state for the object.

© William L HONIG 2019