UDACITY

Where Should One Use Object-Oriented Programming?

One of the main questions novice programmers have while they are learning the basics of coding is: “Where should I use object-orientec programming?” The answer isn’t something simple, like: “always.” Instead, it is important to pay attention to some of the advantages c object-oriented programming and think about whether those advantage: will be of use to you in your current application.

One of the main times to use object-oriented programming is when you need to make use of what is called **polymorphism**. That is any time you might have different types of data as inputs to you program or function and want to be able to deal correctly with each. Like all of these features, it can be done without classes and objects, but much less easily and much less clearly.

An example of this sort of use is within a graphics application. You might have a program which wants to draw some set of shapes on the screen. With polymorphism, each of these shapes might be assumed to have a draw() function which would draw that particular shape, perhaps with arguments for location and orientation of the shape. Your program could simply go down the list of objects on screer and call the draw() function for each of them. Without polymorphism, your overarching program would instead have to define a function whicl checked what kind of shape was being drawn and then executec specific instructions for that shape. An example of how inelegant tha can look can be found here.

This borders on another central reason to use object-orientec programming, called **inheritance**. Inheritance is used to allow for kinds of objects to be grouped together. To use the above example, we coulc have an Octagon class and a Rhombus class that both inherit from the Shape class. This means that each of them could use the code contained in the shape class (such as location, orientation, line thickness, draw() method, etc) and extend that code to fit the more specific needs of an Octagon or Rhombus (the draw() method would be very different for each).

A second advantage of this facet of object-orientec programming is the ability to have your code match your intuitions abou

objects in real life, as in this example. By using inheritance effectivel
we can match our intuitive understanding that, for instance, movies an
tv shows are both more specific versions of something we would call a
'video'.

This is vital whenever you are working on a program whic
might contain many different kinds of data. Graphics programs
aggregator sites, and video game engines are three areas where thi
aspect of object-oriented programming comes to the forefront.

The last big reason to use object-oriented programming is calle
encapsulation. Encapsulation is essentially the idea of grouping cod
snippets and data together and assigning gate-keeping functions whic
are the only way to access that data. This concept has two advantages
The first is security and stability. With proper encapsulation, th
designer of the code can control exactly how data within an object i
able to be modified, which keeps the innards of the code safe fro
unexpected changes.

If a new programmer starts to work with you on a project an
forgets to cast someone's age as an integer before assigning it to th
age variable, that can cause problems down the line. With prope
encapsulation, that check will be done automatically in either th
constructor function or a function to modify the age variable. Becaus
the private variables in the object can only be changed through thes
specific functions, they are less likely to take problematic values.

The second advantage of encapsulation is that this makes i
easy to change how a variable is accessed by all of the parts of
program without having to modify each individual part. This become
useful when using statistical applications, because one can easil
choose to use the data directly or the log of the data or some othe
modification of the data without having to write that code for each dat
point. This also makes it easier to change the way a large project work
without introducing a huge number of bugs - As long as the inputs an
outputs of all functions remain of the same type.

Object-oriented programming is not a silver bullet, and it will no
solve all of your programming woes, but it can go a long way toward
making your code more elegant and easier to read in situations wher
polymorphism, inheritance, or encapsulation are of use.