



## Our Object-Oriented Approach (The Core of Backware)

The term object-oriented is used so frequently today that it is difficult to decipher what it really means. When we say that Backware is object-oriented, we mean your data is organized into units called *objects* which each belong to a *class* of objects. Each object has a set of *properties* or ways to describe it. The class that an object belongs to defines the set of properties that all its objects will use. For instance, suppose you have a golden delicious apple and a red delicious apple. Both of them are objects, and they are both of the class “Apple”. The “Apple” class might define the set of properties for all apples to be “Name” and “Color”. The specific values for the first apple would be “Golden Delicious” and “Yellow”, while the property values for the second apple would be “Red Delicious” and “Red”. So, there is one class with multiple objects that have the same set of descriptors but different values.

In object-oriented programming, objects will often contain methods which are ways to modify themselves or other objects. In our system, the objects themselves do not contain methods-only data. However, we provide a system called Processes that allows a user to accomplish those same kinds of tasks with even greater flexibility. For instance, each process can have its own set of permissions, allowing only certain users to do certain things. (For more details, please see the section on processes in the Building Blocks document.) So, the important thing to recognize is that our processes exist outside of objects and manipulate them independently rather than being part of objects and changing them from the inside.

Sometimes, object-oriented systems will provide class inheritance. Inheritance means that a particular class may have one or more parents from which it inherits properties. An example might be that all apples are also fruit and thus inherit all properties thereof. Because of operational dependencies and complexities, Backware does not support class inheritance.