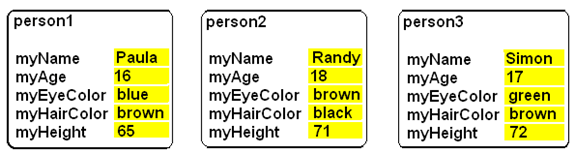
**Making Multiple Objects**

You recognize a class is a blueprint and that objects represent instances of a class. For example, you represent a unique object because you are an instance of a class of objects known as human beings. You share a set of attributes with every other human being; name, age, eye color, hair color, height, mass, etc.; however, the *values* of your instance variables are different from everyone else's.

Yet, as an object of the human class, you execute the same basic behaviors as the next person. Humans have specific methods for seeing, feeling, touching, tasting, and hearing. Everyone uses the same basic method for walking and talking, to pump blood and breathe air, etc. From a programmer's perspective, when a new instance of a human is needed, one is created from the blueprint for the class, complete with its own instance variables but with a common set of behaviors.  
  
Imagine that you need to create three Person objects with attributes for name, age, eye color, hair color, and height. These attributes for each person can be visualized as separate compartments with unique data assigned to each private instance variable.



Notice that the names of each object's instance variables are identical (e.g., myName), but they are each assigned unique private data (e.g. Paula, Randy, and Simon). Furthermore, these compartments are separated, so the private information in each object is inaccessible by the other objects. This is the principle of data hiding in action.

Because each object shares the same instance variable, all three objects can be instantiated by the same constructor. This is one of the keys to object-oriented programming: the ability to construct multiple objects.

**Code 1**

What would a constructor look like for a Person class with attributes for name, age, eye color, hair color, and height? Following the pattern established in previous programs, it includes a declaration of private instance variables, a constructor header with a parameter list, and the assignment of the local parameter variables to the private instance variables within the body of the constructor.

public class Person  
{  
private String myName, myEyeColor, myHairColor;  
private int myAge, myHeight;  
  
public Person(String name, int age, String eyeColor, String hairColor, int height)  
{  
myName = name;  
myAge = age;  
myEyeColor = eyeColor;  
myHairColor = hairColor;  
myHeight = height;  
}  
  
//Person class methods not shown  
}

**Code 2**

You will see this pattern (declaration of private instance variables, a constructor with a parameter list, and the assignment of parameter variables to private instance variables) over and over again in object-oriented programs.

What would the invoking statement to instantiate an object with this constructor look like? It would need to reference the constructor by name (i.e., Person), name the new object, and provide a list of arguments that corresponds to the parameter list. The following statements would instantiate three separate Person objects (person1, person2, and person3), each with a unique set of attributes.

Person person1 = new Person("Paula",16, "blue","brown",65);  
Person person2 = new Person("Randy",18,"brown","black",73);  
Person person3 = new Person("Simon",17,"green","brown",71);

### Code 3

After the objects are created, any methods of the class can be invoked with statements such as the following:

String name1 = person1.getName();  
String hairColor2 = person2.getHairColor();  
int yearBorn3 = person3.calculateYearBorn();

These statements would get the name, the hair color, and the year born (using the age private instance variable) for the person1, person2, and person3 objects, respectively. Consequently, once a class is created with specific instance variables and methods, any number of independent objects can be constructed.