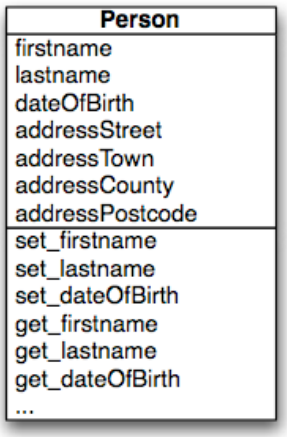
PHP Exercises – Classes and Objects

Hints about PHP syntax are on the back of this document.



One
Make sure that you place all your files in a single folder / directory.

Using an appropriate text editor, create a file called *class.Person.php*

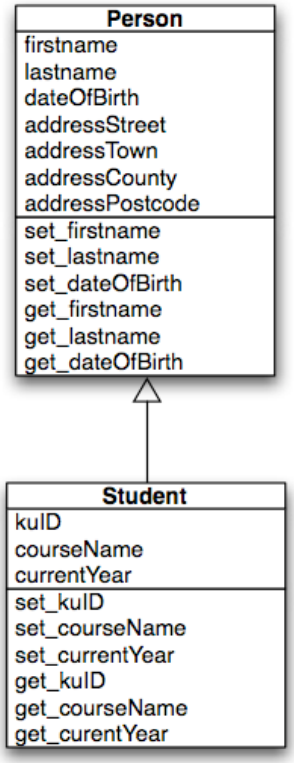
Inside the file, write a class called *Person* with the seven properties shown in the Person class picture (right). All fields are strings, with the date of birth being stored in a string in "YY-MM-DD" format.

Add appropriate methods for the class (a constructor and get/set pairs for each property).

Two
Create a new file called usesPerson.php. Your file should have the following structure:

```
require_once your class.Person.php file
create a Person object called $myPerson
Give the Person object values for name, date of birth and address using the constructor
<html>
<head>
<title>Person example</title>
</head>
<body>
Print the object out using appropriate print / get methods
</body>
</html>
```

Upload the files and test them out.



Three
A student is a descendant of person, but has extra properties

- kuID – student identifier
- courseName – full name of course (i.e. BSc Business Information Technology)
- currentYear -1,2,3 or 4

In a new file called *class.Student.php* create a descendant of *Person* that extends the parent class adding the extra methods along with appropriate get/set properties.

Your student constructor function should reuse the Person constructor where appropriate.

Four
Save the file *usesPerson.php* as a new file *usesStudent.php*.

In this file, change the code so that a student object is declared, appropriate (made up) values are passed into the object through the constructor, then print the object out.

To create an object of a given class, use the `new` keyword:

```
$object = new Class;
```

Once you have an object, you can use the `->` notation to access methods and properties of the object:

```
$object->propertyname
$object->methodname([arg, ... ])
```

The syntax for a class definition:

```
class classname [ extends baseclass ]
{
    [ var $property [ = value ]; ... ]

    [ function functionname (args) {
          // code
      }
      ...
    ]
}
```

To inherit the properties and methods from another class, use the `extends` keyword in the class definition, followed by the name of the base class

```
class $a {
  var …
}

class $b extends $a {
  var …
}
```

If a derived class has a property or method with the same name as one in its parent class, the property or method in the derived class takes precedence over, or *overrides*, the property or method in the parent class.

To access an overridden method, use the `parent::method( )` notation.

You may also provide a list of arguments following the class name when instantiating an object:

```
$newA = new A(values…);
```

These arguments are passed to the class's *constructor* which is a function with the same name as the class in which it is defined. Here's a constructor for the `a` class:

```
class a {
    function a (values) {
        …
    }
}
```