OOP - Second part

Abstraction

Definition

Abstraction helps reduce complexity and programming effort. In PHP, abstraction is done using abstract classes and interfaces. This is one of the most important concepts of OOPs.

Abstract classes are classes in PHP that declare one or more abstract methods.

Abstract class

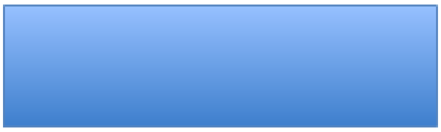
Consider the following class hierarchy:

- The Shape class.
- 3 classes that inherit:
  - o Rectangle,
  - o Circle,
  - o Triangle.

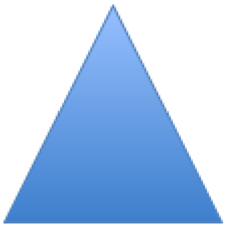
The Shape class is created to save the methods and attributes common to the three Rectangle, Circle, and Triangle classes.

We will now instantiate our objects and see what they look like:

$ rec = new Rectangle ();
// Create a rectangle object



```
$ tri = new Triangle ();
// Create a triangle object
```



```
$ sha = new Shape ();
// create a shape object
```

????

In our example, the Shape class serves us in our inheritance but was not built to be instantiated. Indeed, what can a 'Shape' look like?

Such classes may be marked '*Abstract*'.
An abstract class can not be instantiated.

Abstract

An abstract method is a method that has only the definition of the method (the *signature* of the method) but does not contain an implementation.

Let's go back to the previous example.
The method for calculating the area of a rectangle, circle, or triangle is different. It does not make sense to put a 'calculateSurface' method to theclass, *Shape* but make sure that all inherited classes can access this method.

In this case, we must create our abstract method:

```
public abstract function calculateSurface ();
```

Precisions

1. An abstract class can contain methods with a body (not just abstract methods).
2. A class can even be abstract without containing any abstract method.
3. If a class has one or more abstract methods, it must be 'Labeled' abstract class.

Interface

Definition

An interface looks like a class but has only static constants and abstract methods. PHP uses Interfaces to implement multiple inheritances.
Indeed, in PHP multiple inheritance is not possible. While we can implement several Interfaces.

All the methods of an interface are implicitly public and abstract.

Syntax

To create an interface, here is the syntax:

```
interface IExample {
// Methods
}
```

To implement an interface in a class:

```
class example implements IExample {
}
```

Example

We take the example of our class Animal and subclasses Dog, Pig etc. ..

Imagine an interface:

```
    Domestic interface {
    public function sit();
    }
```

Now we can use the class Dog that will implement the interface Domestic:

```
    class Dog implements Domestic {

        public function sit () {
        echo "The dog implements the interface";
}
    }
```

Interface vs Abstract

class Abstract class

- When ' model '/ template must be set for a group of subclasses, use an abstract class.
- a class can contain concrete methods (with the code).
- abstract classes provide default actions for subclasses.
- used to provide a template / model for potential future inherited classes

Interface

- Use an interface when a role must be defined for other classes as well, regardless of the inheritance of these classes
- You can not instantiate an interface and create an object.
- The interface can not contain concrete methods
- A class can implement multiple interfaces
- An interface can inherit from a class.