# Lab: OOP Advanced

This document defines the **exercise assignments** for the "[PHP Web Development Basics" Course @ Software University](https://softuni.bg/trainings/1746/php-web-developmentbasics-september-2017) .

## Part I. Basic Use of Interfaces

## Circle and Area interface

## One Class and One Interface

Build a simple class Circle and an interface like this:

|  |
| --- |
| inteface  Area |
| **getSurface()** |

|  |
| --- |
| **Circle** |
| **radius** |

You should have one class called **Circle** with one property **radius.** Define an interface called **Area** which calculates the area of the circle on the basis of its radius. It should implement one method **getSurface()** which returns the area of the circle. For the calculation look at: http://mathworld.wolfram.com/Circle.html

**Create one instance of Circle and use the implemented method to calculate a circle with radius 10 mm.**

|  |  |
| --- | --- |
| **Input** | **Output** |
| $myCircle with radius 10 mm. | Circle, radius = 10 mm, area = … mm |

## Circle and rectangle and Area Interface

## Two Classes and One Interface

Now we extend the previous task with one more class: Rectangle. The rectangle doesn’t have a radius but **width** and **height**. So its area is calculated in a different way. But the **Area interface** should be applied to it also. Implement the method **getArea()** in the Rectangle class. What is different in the method now?

Create an instance of a Rectangle $myRec and calculate the area of a specific rectangle as given in the Input /Output example

|  |  |
| --- | --- |
| **Input** | **Output** |
| $mRec with width = 15 mm and height = 35 mm | Rectangle, width = 15mm, height = 35 mm, area = … mm |

Your hierarchy of classes and interfaces should look like this:

|  |
| --- |
| inteface  Area |
| **getSurface()** |

|  |
| --- |
| **Circle** |
| **radius** |

|  |
| --- |
| Rectangle |
| **width**  **height** |

## Part II. Extended Use of Interfaces

## 2.1. Circle and Area + Circumference Interfaces

## One Class and Two Interfaces

A class can implement more than one interface. Let’s get back to our Circle. Besides an area, a circle can have a circumference (the length of the line of the circle which is designated by c). See the picture bellow.

Now, let the **class Circle** implement the interface **Area** and implement the interface **Circumference**. Area defines a method **getSurface()** and Circumference defines a method **getCircumference().**

How would the two methods look for a Circle? Calculate a circle’s surface/area and circumference. See the input /output table.

For the calculation of circumference look at: <http://mathworld.wolfram.com/Circle.html>

You should be able to use the class like this:

|  |
| --- |
| inteface  Area |
| **getSurface()** |

|  |
| --- |
| interface  Circumference |
| **getCircumference()** |

|  |
| --- |
| **Circle** |
| **radius** |

|  |  |
| --- | --- |
| **Input** | **Output** |
| $myCircle with radius 10 mm. | Circle with radius = 10 mm:  Area = … mm  Circumference = …. mm |

**How many interfaces can a class implement?**

## Part III. Advanced Use of Interfaces

## 3.1. Hardware Devices and Their Interfaces

## Multiple Interfaces and Multiple Implementations

Imagine that you work in an office and you have to **model** the behaviour of the **hardware devices** in the office. Different hardware devices have different interfaces or ways they connect to people that use them. What can they be:

1. Interface Keyboard with method **pressKey()** and **changeStatus()**
2. Interface Mouse with methods **move()** and **click()**
3. Interface TouchPad with methods **moveFinger()** and **click()**;

Of course there can be much more interfaces than this. But **different devices implement different interfaces**. Some of the devices may implement **all, some or none** of these. So you have at your office the following classes of devices:

1.Class DesktopComputer, 2.Class Laptop, 3.Class Tablet, 4.Class MobilePhone

But these classes are children of the following **abstract classes**: **Computer** and **Mobile**. There is no abstract class for Printer. So DesktopComputer and Laptop inherit abstract Computer. Tablet and MobilePhone inherit Mobile.

See the chart how classes and abstract classes and interfaces are connected. The **orange lines** represent implementations of interfaces. **Blue lines** represent **inheritance**.

|  |
| --- |
| **Abstract class**  **Computer** |
| **processor**  **ram** |

|  |
| --- |
| **Abstract class**  **Mobile** |
| **operator**  **canCall (true / false)**  **battery(%)** |

|  |
| --- |
| **DesktopComputer** |
| **keyboardConnected** |

|  |
| --- |
| inteface  Keyboard |
| **pressKey()**  **changeStatus()** |

|  |
| --- |
| **Laptop** |
| **battery (%)** |

|  |
| --- |
| interface  Mouse |
| **click()**  **move()** |

|  |
| --- |
| **Tablet** |
| **stdResolution** |

|  |
| --- |
| interface  TouchScreen |
| **moveFinger()**  **clickFinger()**  **writeText()** |

|  |
| --- |
| **MobilePhone** |
| **size** |

Create the class hierarchy. If you do the lab at home then create every class in a separate file. Now in the particular office we have:

**One desktop computer - $d, two laptops - $l1 and $l2, one tablet - $t, three mobile phones - $m1, $m2, $m3.**

**How many interfaces** can a class implement and **how many abstract classes** can it inherit**?** (Discuss if you are in a group)

## 3.2. Notebook++

In the office you have a new device which is a **notebook** by which you can also make **phone calls**. It has a touchscreen and can have also a keyboard and a mouse attached to it. It has the characteristics of a laptop and of a tablet.

1. **What abstract class** can Notebook++ extend?
2. **Which interfaces** should Notebook++ implement?
3. Create an instance in the variable **$n**
4. Introduce in the hierarchy a property called **writtenText** and write the two methods **pressKey()** and **writeText()** to **put the text data** in the property writtenTextand **return** it to the caller**. Where** should the property **be put and why?** (Give different options)

## 3.2. The Mouse

Write the methods **click()** and **move()** for a desktop computer and a laptop. **Where** would you put the bodies of the methods? The click method can have a **leftClick** (true|false) and **rightClick** (true|false) arguments. The **move()** method receives as arguments: **currentX** and **currentY** coordinates (in pixels) and **offesetX, offsetY** (in pixels). It returns the **newX**, **newY** coordinates.