# Lab: Objects and Classes

Problems for lab for the "PHP Fundamentals" course @ SoftUni.

You can check your solutions in Judge.

## 1. Day of Week

You are given a **date** in format **"{day}-{month}-{year}"**. Calculate and print the **day of week** in **English**.

### Examples

| Input | Output |
|---|---|
| 18-04-2016 | Monday |
| 27-11-1996 | Wednesday |

### Hints

- **Read the date as string** from the Console.
- Use **DateTime()** to convert the input string to object of type **DateTime**
- Format the received date with **'l'** - this will return a full textual representation of the day of the week
  You can see here for more details

```php
<?php
$dateAsString = readline();
$date = new DateTime($dateAsString);

echo $date->format( format: 'l') . PHP_EOL;
```

## 2. Person Info

Create a person class that receives **first name**, **last name** and **age**. Print the entries of a given object.

### Examples

| Input | Output |
|---|---|
| Peter<br>Pan<br>20 | firstName: Peter<br>lastName: Pan<br>age: 20 |

### Hints

- Create the Person class and create fields

```php
class Person
{
    private $firstName;
    private $lastName;
    private $age;
```

- Create getters and setters for **firstName**, **lastName** and **age**

```php
public function getFirstName()
{
    return $this->firstName;
}

public function setFirstName($firstName)
{
    $this->firstName = $firstName;
}
//TODO: create the rest
```

- Create new Person Object and print the result

```php
$person = new Person ();
$person->setFirstName(readline());
$person->setLastName(readline());
$person->setAge(readline());

echo "firstName:{$person->getFirstName()}" . PHP_EOL;
echo "lastName:{$person->getLastName()}" . PHP_EOL;
echo "age:{$person->getAge()}" . PHP_EOL;
```

## 3. Songs

Define a class **Song**, which holds the following information about songs: **Type List**, **Name** and **Time**.

On the first line you will receive the **number of songs - N**.

On the **next N-lines** you will be receiving data in the following format: **"{typeList}_{name}_{time}".**

On the last line you will receive **"{Type List}"** or **"all"**. Print only the **Names of the songs** which are from that **"{Type List}"** or **"all"**.

### Examples

| Input | Output |
|---|---|
| 3<br>favourite_DownTown_3:14<br>favourite_Kiss_4:16<br>favourite_Smooth Criminal_4:01<br>favourite | DownTown<br>Kiss<br>Smooth Criminal |
| 4<br>favourite_DownTown_3:14<br>listenLater_Andalouse_3:24<br>favourite_In To The Night_3:58<br>favourite_Live It Up_3:48<br>listenLater | Andalouse |
| 2<br>like_Replay_3:15<br>ban_Photoshop_3:48<br>all | Replay<br>Photoshop |

## Solution

Define class Song with properties: **Type List**, **Name** and **Time**.

```
class Song
{
    private $typeSong;
    private $name;
    private $time;
```

Create constructor and getters:

```
public function __construct($typeSong, $name, $time)
{
    $this->typeSong = $typeSong;
    $this->name = $name;
    $this->time = $time;
}

public function getTypeSong()
{
    return $this->typeSong;
}
```

Read the input lines, make collection and store the data.

```
for ($i = 0; $i < $songs; $i++) {
    $current = explode( delimiter: '_', readline());
    $type = $current[0];
    $name = $current[1];
    $time = $current[2];
    $song = new Song($type, $name, $time);
    array_push( &array: $arraySongs, $song);
}
```

Finally read your last line – **Type List** and **print** the result.

```
$typeList = readline();

if ($typeList == 'all') {
    foreach ($arraySongs as $value) {
        echo $value->getName() . PHP_EOL;
    }
} else {
    foreach ($arraySongs as $value) {

        if ($value->getTypeSong() === $typeList) {
            echo $value->getName() . PHP_EOL;
        }
    }
}
```

## 4. Students

Define a class **`Student`**, which holds the following information about students: **first name**, **last name**, **age** and **hometown**.

Read list of students until you receive "**end**" command. After that, you will receive a **city name**. Print only students which are from the given city, in the following format: **`"{firstName} {lastName} is {age} years old."`**.

### Examples

| Input | Output |
|---|---|
| `John Smith 15 Sofia`<br>`Peter Ivanov 14 Plovdiv`<br>`Linda Bridge 16 Sofia`<br>`Simon Stone 12 Varna`<br>`end`<br>`Sofia` | `John Smith is 15 years old.`<br>`Linda Bridge is 16 years old.` |
| `Anthony Taylor 15 Chicago`<br>`David Anderson 16 Washington`<br>`Jack Lewis 14 Chicago`<br>`David Lee 14 Chicago`<br>`end`<br>`Chicago` | `Anthony Taylor is 15 years old.`<br>`Jack Lewis is 14 years old.`<br>`David Lee is 14 years old.` |

### Hints

- Define a class Student with the following fields: **`firstName`**, **`lastName`**, **`age`** and **`city.`**
- Generate constructor in class Student.
- Read a list of students.
- Read a city name and print only students which are from the given city.

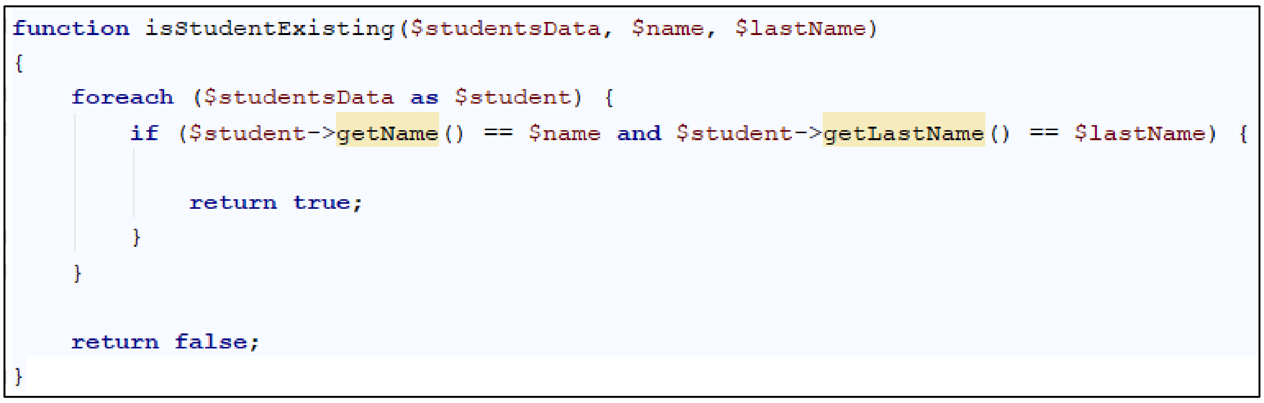
## 5. Students 2.0

Use the class from the previous problem. If you receive a student which already exists (**first name** and **last name** should be **unique**) overwrite the information.

| Input | Output |
|---|---|
| `John Smith 15 Sofia`<br>`Peter Ivanov 14 Plovdiv`<br>`Peter Ivanov 25 Plovdiv`<br>`Linda Bridge 16 Sofia`<br>`Linda Bridge 27 Sofia`<br>`Simon Stone 12 Varna`<br>`end`<br>`Sofia` | `John Smith is 15 years old.`<br>`Linda Bridge is 27 years old.` |
| `Anthony Taylor 15 Chicago`<br>`David Anderson 16 Washington`<br>`Jack Lewis 14 Chicago`<br>`David Lee 14 Chicago`<br>`Jack Lewis 26 Chicago`<br>`David Lee 18 Chicago`<br>`end`<br>`Chicago` | `Anthony Taylor is 15 years old.`<br>`Jack Lewis is 26 years old.`<br>`David Lee is 18 years old.` |

## Hints

Check if the given student already exists.

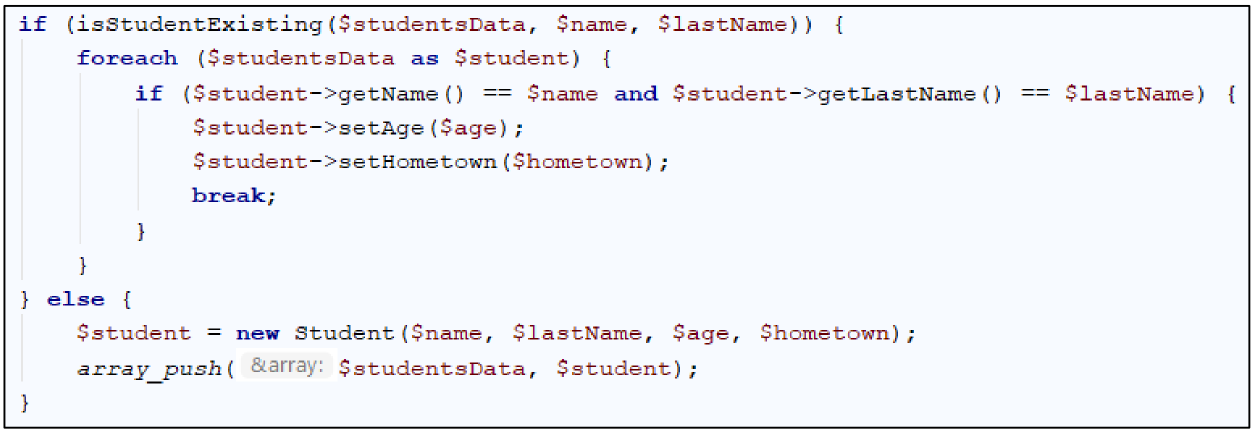


```
function isStudentExisting($studentsData, $name, $lastName)
{
    foreach ($studentsData as $student) {
        if ($student->getName() == $name and $student->getLastName() == $lastName) {

            return true;
        }
    }

    return false;
}
```

Overwrite the student information if the student exists.



```
if (isStudentExisting($studentsData, $name, $lastName)) {
    foreach ($studentsData as $student) {
        if ($student->getName() == $name and $student->getLastName() == $lastName) {
            $student->setAge($age);
            $student->setHometown($hometown);
            break;
        }
    }
} else {
    $student = new Student($name, $lastName, $age, $hometown);
    array_push( &array: $studentsData, $student);
}
```

## 6. Store Boxes

Define a class **Item** which contains these properties: **Name and Price.**

Define a class **Box** which contains these properties: **Serial Number, Item, Item Quantity and Price for a Box.**

Until you receive **"end"** you will be receiving data in the following format:
"**`{Serial Number} {Item Name} {Item Quantity} {itemPrice}`**"

The **Price of one box** has to be calculated: **`itemQuantity * itemPrice`.**

Print all the boxes, ordered descending by price for a box, in the following format:

```
{boxSerialNumber}

-- {boxItemName} – ${boxItemPrice}: {boxItemQuantity}

-- ${boxPrice}
```

Price should be **formatted to the 2nd character after the decimal point**.

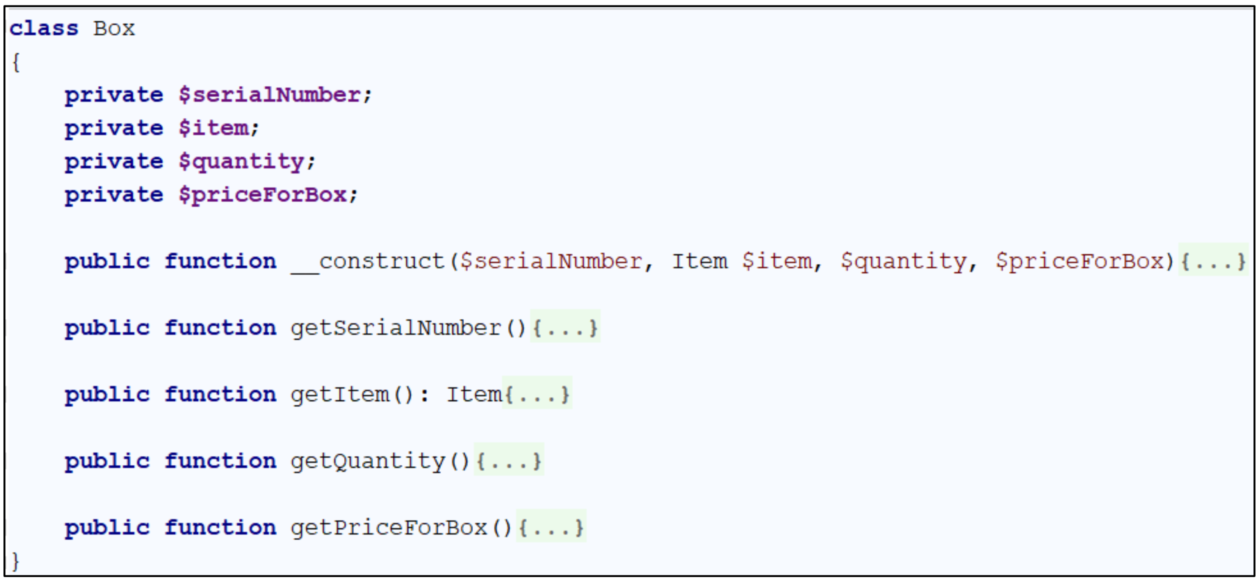
### Examples

| Input | Output |
|---|---|
| 19861519 Dove 15 2.50<br>86757035 Butter 7 3.20<br>39393891 Orbit 16 1.60 | 37741865<br>-- Samsung - $1000.00: 10<br>-- $10000.00 |

| | |
|---|---|
| 37741865 Samsung 10 1000<br>end | 19861519<br>-- Dove - $2.50: 15<br>-- $37.50<br>39393891<br>-- Orbit - $1.60: 16<br>-- $25.60<br>86757035<br>-- Butter - $3.20: 7<br>-- $22.40 |
| 48760766 Alcatel 8 100<br>97617240 Intel 2 500<br>83840873 Milka 20 2.75<br>35056501 SneakersXL 15 1.50<br>end | 97617240<br>-- Intel - $500.00: 2<br>-- $1000.00<br>48760766<br>-- Alcatel - $100.00: 8<br>-- $800.00<br>83840873<br>-- Milka - $2.75: 20<br>-- $55.00<br>35056501<br>-- SneakersXL - $1.50: 15<br>-- $22.50 |

## Hints

This is how your class Box and class Items should look like.



```
class Box
{
    private $serialNumber;
    private $item;
    private $quantity;
    private $priceForBox;

    public function __construct($serialNumber, Item $item, $quantity, $priceForBox){...}

    public function getSerialNumber(){...}

    public function getItem(): Item{...}

    public function getQuantity(){...}

    public function getPriceForBox(){...}
}
```

```
class Item
{
    private $name;
    private $price;

    public function __construct($name, $price)
    {
        $this->name = $name;
        $this->price = $price;
    }

    public function getName(){...}


    public function getPrice(){...}
}
```

## 7. Vehicle Catalogue

Your task is to **create Vehicle catalogue** which contains only **Trucks and Cars**.

Define class **Truck** with these fields: **Brand, Model and Weight**.

Define class **Car** with these fields: **Brand, Model and Horse Power**.

Define class **Catalog** with these fields: **Collections of Trucks and Cars**.

You have to read your input until you receive the "**end**" command.

The input will be in following format: **`"{type}/{brand}/{model}/{horse power / weight}"`**

In the end you have **to print all vehicles ordered alphabetical by brand,** in the following format:

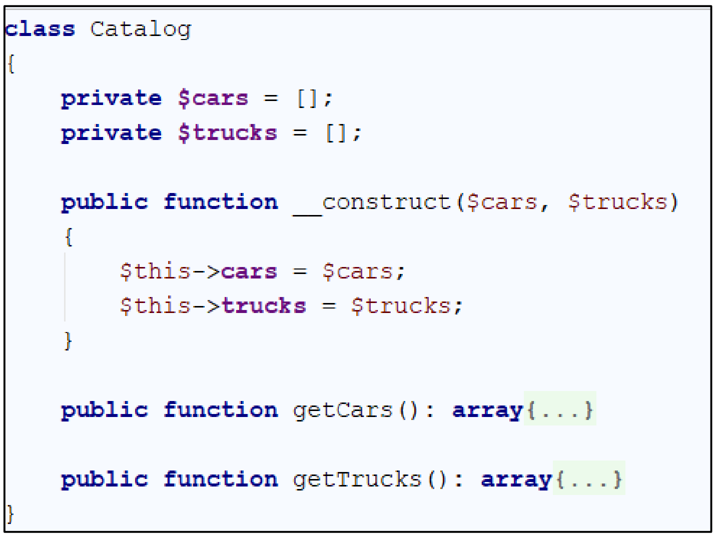
```
Cars:
{Brand}: {Model} - {Horse Power}hp
Trucks:
{Brand}: {Model} - {Weight}kg
```

### Examples

| Input | Output |
|---|---|
| Car/Audi/A3/110<br>Car/Maserati/Levante/350<br>Truck/Mercedes/Actros/9019<br>Car/Porsche/Panamera/375<br>end | Cars:<br>Audi: A3 - 110hp<br>Maserati: Levante - 350hp<br>Porsche: Panamera - 375hp<br>Trucks:<br>Mercedes: Actros - 9019kg |
| Car/Subaru/Impreza/152<br>Car/Peugeot/307/109<br>end | Cars:<br>Peugeot: 307 - 109hp<br>Subaru: Impreza - 152hp |

### Hints

This is how your class **Catalog** should look like.



```
class Catalog
{
    private $cars = [];
    private $trucks = [];

    public function __construct($cars, $trucks)
    {
        $this->cars = $cars;
        $this->trucks = $trucks;
    }

    public function getCars(): array{...}

    public function getTrucks(): array{...}
}
```