Lab: Inheritance

This document defines the exercises for the "Python OOP" course at @Software University.

Please submit your solutions (source code) to all the below-described problems in Judge.

Part I: Inheritance

1. Food

In a folder called **`project`** create two files: **`food.py`** and **`fruit.py`**:

- In the **`food.py`** file, create a class called **`Food`** which will receive an **`expiration_date`** (str) upon initialization.
- In the **`fruit.py`** file, create a class called **`Fruit`** which will receive a **`name`** (str) and an **`expiration_date`** (str) upon initialization.

`Fruit` should be inherited from **`Food`**.

Submit in Judge a **zip file** of the folder **`project`**.

2. Single Inheritance

In a folder called **`project`** create two files: **`animal.py`** and **`dog.py`**:

- In the **`animal.py`** file, create a class called **`Animal`** with a single method **`eat()`** that returns: **"eating..."**.
- In the **`dog.py`** file, create a class called **`Dog`** with a single method **`bark()`** that returns: **"barking..."**.

The **`Dog`** should inherit from **`Animal`**.

Submit in Judge a **zip file** of the folder **`project`**.

3. Multiple Inheritance

In a folder called **`project`** create three files: **`person.py`**, **`employee.py`**, and **`teacher.py`**.

In each file, create its corresponding class - **`Person`**, **`Employee`**, and **`Teacher`**:

- **`Person`** with a single method **`sleep()`** that returns: **"sleeping..."**
- **`Employee`** with a single method **`get_fired()`** that returns: **"fired..."**
- **`Teacher`** with a single method **`teach()`** that returns: **"teaching..."**.

`Teacher` should inherit from **`Person`** and **`Employee`**.

Submit in Judge a **zip file** of the folder **`project`**.

4. Multilevel Inheritance

In a folder called **`project`** create three files: **`vehicle.py`** and **`car.py`**, and **`sports_car.py`**.

In each file, create its corresponding class - **`Vehicle`**, **`Car`**, and **`SportsCar`**:

- **`Vehicle`** with a single method **`move()`** that returns: **"moving..."**
- **`Car`** with a single method **`drive()`** that returns: **"driving..."**
- **`SportsCar`** with a single method **`race()`** that returns: **"racing..."**.

`SportsCar` should inherit from **`Car`** and **`Car`** should inherit from **`Vehicle`**.

Submit in Judge a **zip file** of the folder **`project`**.

5. Hierarchical Inheritance

In a folder called **`project`** create three files: **`animal.py`**, **`dog.py`**, and **`cat.py`**.

In each file, create its corresponding class - **`Animal`**, **`Dog`**, and **`Cat`**:

- **`Animal`** with a single method **`eat()`** that returns: **"eating..."**
- **`Dog`** with a single method **`bark()`** that returns: **"barking..."**
- **`Cat`** with a single method **`meow()`** that returns: **"meowing..."**

Both **`Dog`** and **`Cat`** should inherit from **`Animal`**.

Submit in Judge a **zip file** of the folder **`project`**.

Part II: Reusing Classes

6. Stack of Strings

Create a class **`Stack`** that can store **only strings** and has the following functionality:

- Instance attribute: **`data: list`**
- Method: **`push(element)`** – adds an element at the end of the stack
- Method: **`pop()`** – removes and returns the last element in the stack
- Method: **`top()`** - returns a reference to the topmost element of the stack
- Method: **`is_empty()`** - returns boolean **`True/False`**
- Override the **string method** to return the stack data in the format:
  `"[{element(N)}, {element(N-1)} ... {element(0)}]"`