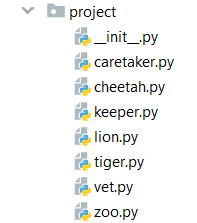
# Exercise: Encapsulation

Problems for exercise and homework for the [Python OOP Course @SoftUni](https://softuni.bg/courses/python-oop). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1939>

## Wild Cat Zoo

In this exercise we are going to create a whole project called "**Wild Cat Zoo**". We are going to create the project step-by-step starting with the project structure:



Please create separate file for each class as shown above and submit a zip file containing all files (zip the whole project folder/module) - it is important to include all files in project module in order to be able to make proper imports.

### Class Lion

#### Attributes

Public attribute **name: string**

Public attribute **gender: string**

Public attribute **age: number**

#### Methods

**\_\_init\_\_(name, gender, age)** - set all the attributes to the given ones

**get\_needs()** - returns the number **50** (amount of money needed to tend the animal)

**\_\_repr\_\_()** - returns string representation of the lion in the format: **"Name: {name}, Age: {age}, Gender: {gender}"**

### Class Tiger

#### Attributes

Public attribute **name: string**

Public attribute **gender: string**

Public attribute **age: number**

#### Methods

**\_\_init\_\_(name, gender, age)** - set all the attributes to the given ones

**get\_needs()** - returns the number **45** (amount of money needed to tend the animal)

**\_\_repr\_\_()** - returns string representation of the tiger in the format: **"Name: {name}, Age: {age}, Gender: {gender}"**

### Class Cheetah

#### Attributes

Public attribute **name: string**

Public attribute **gender: string**

Public attribute **age: number**

#### Methods

**\_\_init\_\_(name, gender, age)** - set all the attributes to the given ones

**get\_needs()** - returns the number **60** (amount of money needed to tend the animal)

**\_\_repr\_\_()** - returns string representation of the cheetah in the format: **"Name: {name}, Age: {age}, Gender: {gender}"**

### Class Keeper

#### Attributes

Public attribute **name: string**

Public attribute **age: number**

Public attribute **salary: number**

#### Methods

**\_\_init\_\_(name, age, salary)** - set all the attributes to the given ones

**\_\_repr\_\_()** - returns string representation of the keeper in the format: **"Name: {name}, Age: {age}, Salary: {salary}"**

### Class Caretaker

#### Attributes

Public attribute **name: string**

Public attribute **age: number**

Public attribute **salary: number**

#### Methods

**\_\_init\_\_(name, age, salary)** - set all the attributes to the given ones

**\_\_repr\_\_()** - returns string representation of the caretaker in the format: **"Name: {name}, Age: {age}, Salary: {salary}"**

### Class Vet

#### Attributes

Public attribute **name: string**

Public attribute **age: number**

Public attribute **salary: number**

#### Methods

**\_\_init\_\_(name, age, salary)** - set all the attributes to the given ones

**\_\_repr\_\_()** - returns string representation of the vet in the format: **"Name: {name}, Age: {age}, Salary: {salary}"**

### Class Zoo

#### Attributes

**Private** attribute **animal\_capacity: number**

**Private** attribute **workers\_capacity: number**

**Private** attribute **budget: number**

Public attribute **name: string**

Public attribute **animals: list (empty upon initialization)**

Public attribute **workers: list (empty upon initialization)**

#### Methods

**\_\_init\_\_(name, budget, animlal\_capacity, workers\_capacity)** - set the attributes to the given ones

**add\_animal(animal, price)**

* If you have **enough budget** and **capacity** **add** the animal (instance of Lion/Tiger/Cheetah) to the **animals list**, **reduce** the **budget** and **return** **"{name} the {type of animal (Lion/Tiger/Cheetah)} added to the zoo"**
* If you have capacity, but **no budget**, return **"Not enough budget"**
* In any other case, you **don't have space** and you should return **"Not enough space for animal"**

**hire\_worker(worker)**

* If you have **enough space** for the worker (instance of Keeper/Caretaker/Vet), **add him** to the workers and return **"{name} the {type(Keeper/Vet/Caretaker)} hired successfully"**
* Otherwise return **"Not enough space for worker"**

**fire\_worker(worker\_name)**

* If there **is a worker** with that name in the workers list, **remove** him and return **"{worker\_name} fired successfully"**
* Otherwise return **"There is no {worker\_name} in the zoo"**

**pay\_workers()**

* If you have **enough budget** to pay the workers (sum their salaries) **pay them** and return **"You payed your workers. They are happy. Budget left: {left\_budget}"**
* Otherwise return **"You have no budget to pay your workers. They are unhappy"**

**tend\_animals()**

* If you have **enough budget** to tend the animals **reduce the budget** and return **"You tended all the animals. They are happy. Budget left: {left\_budget}"**
* Otherwise return **"You have no budget to tend the animals. They are unhappy."**

**profit(amount)**

* **Increase the budget** with the given amount of profit

**animals\_status()**

* Returns the following string:

**You have {total\_animals\_count} animals****----- {amount\_of\_lions} Lions:  
{lion1}  
…  
----- {amount\_of\_tigers} Tigers:  
{tiger1}  
…  
----- {amount\_of\_cheetahs} Cheetahs:  
{cheetah1}  
…**

* ***Hint***: use the **\_\_repr\_\_** methods of the animals to print them on the console

**workers\_status()**

* Returns the following string:

**You have {total\_workers\_count} workers  
----- {amount\_of\_keepers} Keepers:  
{keeper1}  
…  
----- {amount\_of\_caretakers} Caretakers:  
{caretaker1}  
…  
----- {amount\_of\_vetes} Vets:  
{vet1}  
…**

* ***Hint***: use the **\_\_repr\_\_** methods of the workers to print them on the console

### Examples

|  |
| --- |
| **Test Code** |
| zoo = Zoo("Zootopia", 3000, 5, 8)  # Animals creation  animals = [Cheetah("Cheeto", "Male", 2), Cheetah("Cheetia", "Female", 1), Lion("Simba", "Male", 4), Tiger("Zuba", "Male", 3), Tiger("Tigeria", "Female", 1), Lion("Nala", "Female", 4)]  # Animal prices  prices = [200, 190, 204, 156, 211, 140]  # Workers creation  workers = [Keeper("John", 26, 100), Keeper("Adam", 29, 80), Keeper("Anna", 31, 95), Caretaker("Bill", 21, 68), Caretaker("Marie", 32, 105), Caretaker("Stacy", 35, 140), Vet("Peter", 40, 300), Vet("Kasey", 37, 280), Vet("Sam", 29, 220)]  # Adding all animals  for i in range(len(animals)):  animal = animals[i]  price = prices[i]  print(zoo.add\_animal(animal, price))  # Adding all workers  for worker in workers:  print(zoo.hire\_worker(worker))  # Tending animals  print(zoo.tend\_animals())  # Paying keepers  print(zoo.pay\_workers())  # Fireing worker  print(zoo.fire\_worker("Adam"))  # Printing statuses  print(zoo.animals\_status())  print(zoo.workers\_status()) |
| **Output** |
| Cheeto the Cheetah added to the zoo  Cheetia the Cheetah added to the zoo  Simba the Lion added to the zoo  Zuba the Tiger added to the zoo  Tigeria the Tiger added to the zoo  Not enough space for animal  John the Keeper hired successfully  Adam the Keeper hired successfully  Anna the Keeper hired successfully  Bill the Caretaker hired successfully  Marie the Caretaker hired successfully  Stacy the Caretaker hired successfully  Peter the Vet hired successfully  Kasey the Vet hired successfully  Not enough space for worker  You tended all the animals. They are happy. Budget left: 1779  You payed your workers. They are happy. Budget left: 611  Adam fired successfully  You have 5 animals  ----- 1 Lions:  Name: Simba, Age: 4, Gender: Male  ----- 2 Tigers:  Name: Zuba, Age: 3, Gender: Male  Name: Tigeria, Age: 1, Gender: Female  ----- 2 Cheetahs:  Name: Cheeto, Age: 2, Gender: Male  Name: Cheetia, Age: 1, Gender: Female  You have 7 workers  ----- 2 Keepers:  Name: John, Age: 26, Salary: 100  Name: Anna, Age: 31, Salary: 95  ----- 3 Caretakers:  Name: Bill, Age: 21, Salary: 68  Name: Marie, Age: 32, Salary: 105  Name: Stacy, Age: 35, Salary: 140  ----- 2 Vets:  Name: Peter, Age: 40, Salary: 300  Name: Kasey, Age: 37, Salary: 280 |

## Pizza Calories

### Class Toppings

#### Attributes

Private attribute **topping\_type: string**

Private attribute **weight: double**

#### Methods

**\_\_init\_\_( topping\_type, weight)** - set all the attributes to the given ones

**Getters** and **Setters** to all of the private attributes

### Class Dough

#### Attributes

Private attribute **flour\_type: string**

Private attribute **baking\_technique: string**

Private attribute **weight: double**

#### Methods

**\_\_init\_\_(flour\_type, baking\_technique, weight)** - set all the attributes to the given ones

**Getters** and **Setters** to all of the private attributes

### Class Pizza

#### Attributes

Private attribute **name: string**

Private attribute **dough: Dough**

Private attribute **toppings: dictionary**

Private attribute **toppings\_capacity: number**

#### Methods

**\_\_init\_\_(name, dough, toppings\_capacity)** - set all the attributes to the given ones. Also, **initialize an empty toppings dictionary**. It will contain the **topping type as a key** and the **topping's weight as a value**.

**Getters** and **Setters** to all of the private attributes

**add\_topping(topping: Topping)** - Adds a new topping to the dictionary.

* If there is **no space left** for **a new topping**, raise a **ValueError**: **"Not enough space for another topping"**
* If the topping is **already in the dictionary**, **increase the value of its weight**.

**calculate\_total\_weight()** - returns the total weight of the pizza.

## Football Team Generator

### Class Player

#### Attributes

Private attribute **name: string**

Private attribute **endurance: number**

Private attribute **sprint: number**

Private attribute **dribble: number**

Private attribute **passing: number**

Private attribute **shooting: number**

#### Methods

**\_\_init\_\_(name, endurance, sprint, dribble, passing, shooting)** - set all the attributes to the given ones.

**Getters** and **Setters** to all of the private attributes

**\_\_str\_\_()** - should return:

**"Player: {name}**

**Endurance: {endurance}**

**Sprint: {sprint}**

**Dribble: {dribble}**

**Passing: {passing}**

**Shooting: {shooting}**

**"**

**Note:** There is a **new line at the end** of the **\_\_str\_\_()**!!!

### Class Team

#### Attributes

Private attribute **name: string**

Private attribute **rating: number**

Private attribute **players: list**

#### Methods

**\_\_init\_\_( name, rating)** - set all the attributes to the given ones. Also, initialize a **new collection, containing** all of the **players**.

**Getters** and **Setters** to all of the private attributes

**add\_player(player: Player)** - adds a new player to the team.

* If the player **is already in the team**, return **"Player {name} has already joined"**
* **Add the player** to the team and return **"Player {name} joined team {team\_name}"**

**remove\_player(player\_name: str)** - removes a player by its given name

* **Remove the player** and **return him**
* If the player **is not in the team**, return **"Player {player\_name} not found"**