# **Exercise: Encapsulation**

Problems for exercise and homework for the Python OOP Course @SoftUni. Submit your solutions in the SoftUni judge system at <a href="https://judge.softuni.bg/Contests/1939">https://judge.softuni.bg/Contests/1939</a>

# 1. Wild Cat Zoo

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

## **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**

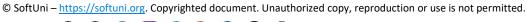
## **Attributes**

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**

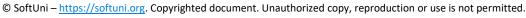
# **Attributes**

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)

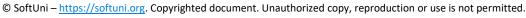
Increase the budget with the given amount of profit

### 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}
      <u>Hint</u>: use the <u>repr</u> 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











