Discussion-8: Modular Programming in Python

Discussion Topic:

PLEASE USE THIS QUESTION FOR PARTICIPATION, WHICH EXTENDS THE STANDARD ONE ABOVE: Select **only one** of the following scenarios and offer a good, complete solution to it. Some repetition is inevitable, but please aim to reply to a scenario that has not already been replied to. It is essential that you include a Python code sample with your work. Please include the scenario number you reply to, so we all can better understand your post, such as "#1", "#2", etc.:

Scenario #1: A national builder has hired your software development firm to create a home inventory program. You first need to develop a class that represents the inventory of the homes in the nation. What components and attributes will need to be included in your home class? Why? *Please include a Python code sample.*

Scenario #2: Your local zoo is opening a new "Serengeti" exhibit and has asked your software firm to create an animal inventory program. You first have to develop a class that represents the inventory of African animals (lions, elephants, wildebeest, etc.) who live in the great African Serengeti plains. What components and attributes will need to be included in the Animal class? Why? *Please include a Python code sample.*

Scenario #3: Your local community college is opening a new campus in a different city and has asked your software firm to create a student roster program. You first have to develop a class that represents the roster of students who are currently studying in this local community college. What components and attributes will need to be included in the Student class? Why? *Please include a Python code sample.*

Scenario #4: A car dealership has hired your software development firm to create a new car inventory program. You first need to develop a class that represents the inventory of the dealership. What components and attributes will need to be included in your automobile class? Why? *Please include a Python code sample.*

My Post:

Hello class,

For this discussion, I chose (I love going to the zoo):
Scenario #2: Your local zoo is opening a new "Serengeti" exhibit and has asked your software firm to create an animal inventory program. You first have to develop a class that represents the inventory of African animals (lions, elephants, wildebeest, etc.) who live in the great African Serengeti plains. What components and attributes will need to be included in the Animal class? Why? *Please include a Python code sample.*

First, Class attributes are a fundamental concept in object-oriented programming (OOP) and play a vital role in creating well-structured and efficient code in Python (Yonkeu, 2023). For a class that would represent the inventory of African animals for the "Serengeti" exhibit, I define the essential components and attributes that define each animal species as follows:

1. species: the species of the animal (e.g., lion, elephant, wildebeest). It helps to identify and categorize the animals.
2. name: the name of the individual animal. It helps in identifying and referring to specific animals.
3. age: the age of the animal in years. It is important to monitor the animal health.
4. gender: stores the gender of the animal (e.g., male, female), for managing breeding programs and maintaining a balanced population in the zoo.
5. weight: stores the weight of the animal. It helps for monitoring the health and growth of the animals.
6. health status: stores the current health status of the animal (e.g., healthy, sick, injured). It helps to monitor the well-being of the animals.
7. location: stores the current location of the animal within the exhibit (e.g., enclosure number, specific area). It helps to locate the animal within the exhibit.
8. diet: stores the dietary classification of the animal (e.g., carnivore, omnivore, herbivore). It helps to determine the appropriate food and nutrition requirements for each animal.
9. preferred food: stores the preferred food items of the animal. It helps in providing a diet that meets the animal's specific preferences.
10. feeding schedule: the feeding schedule of the animal, indicating the times of the day when the animal should be fed. It helps to ensure that the animals receive their meals at appropriate intervals.
11. exhibit: the exhibit the animal belongs to (e.g., "Serengeti"). It helps in managing and organizing animals according to their specific habitat.

The Animal class should include getter and setter methods for each attribute, allowing access and manipulation of the attribute values. Another method is the __str__ method overrides to defaulted representation of the Animal object; this provides a 'string' representation of the Animal object, An additional method is the __del__ method is a destructor method that is called when an Animal object needs to be destroyed.

Here's a sample Python code:

```python
class Animal:
    def __init__(self, species, name, age, gender, weight, health_status,
                 location, diet, preferred_food, feeding_schedule, exhibit):
        """ Initialize an Animal object with the given attributes. """
        self.__species = species
        self.__name = name
        self.__age = age
        self.__gender = gender
        self.__weight = weight
        self.__health_status = health_status
        self.__location = location
        self.__diet = diet
        self.__preferred_food = preferred_food
        self.__feeding_schedule = feeding_schedule
        self.__exhibit = exhibit

    # Getter methods
    def get_species(self):
        """ Return the species of the animal. """
        return self.__species
    def get_name(self):
        """ Return the name of the animal. """
        return self.__name
    def get_age(self):
        """ Return the age of the animal. """
        return self.__age
    def get_gender(self):
        """ Return the gender of the animal. """
        return self.__gender
    def get_weight(self):
        """ Return the weight of the animal. """
        return self.__weight
    def get_health_status(self):
        """ Return the health status of the animal. """
        return self.__health_status
    def get_location(self):
        """ Return the location of the animal. """
        return self.__location
    def get_diet(self):
        """ Return the diet of the animal. """
        return self.__diet
    def get_preferred_food(self):
        """ Return the preferred food of the animal. """
        return self.__preferred_food
    def get_feeding_schedule(self):
        """ Return the feeding schedule of the animal. """
        return self.__feeding_schedule
    def get_exhibit(self):
        """ Return the exhibit the animal belongs to. """
        return self.__exhibit

    # Setter methods
    def set_age(self, age):
        """ Set the age of the animal. """
        self.__age = age
    def set_weight(self, weight):
        """ Set the weight of the animal. """
        self.__weight = weight
    def set_health_status(self, health_status):
        """ Set the health status of the animal. """
        self.__health_status = health_status
    def set_location(self, location):
        """ Set the location of the animal. """
        self.__location = location
    def set_diet(self, diet):
        """ Set the diet of the animal. """
        self.__diet = diet
    def set_preferred_food(self, preferred_food):
        """ Set the preferred food of the animal. """
        self.__preferred_food = preferred_food
    def set_feeding_schedule(self, feeding_schedule):
        """ Set the feeding schedule of the animal. """
        self.__feeding_schedule = feeding_schedule
    def set_exhibit(self, exhibit):
        """ Set the exhibit the animal belongs to. """
        self.__exhibit = exhibit

    def __str__(self):
        """ Return a string representation of the Animal object. """
        return (f"Species: {self.__species}, Name: {self.__name}, Age: {self.__age}, "
                f"Gender: {self.__gender}, Weight: {self.__weight}, Health Status: {self.__health_status},
"
                f"Location: {self.__location}, Diet: {self.__diet}, Preferred Food: {self.__preferred_food
}, "
                f"Feeding Schedule: {self.__feeding_schedule}, Exhibit: {self.__exhibit}")
    def __del__(self):
        """ Print a message when the Animal object is being destroyed. """
        print(f"Animal object '{self.__name}' is being destroyed.")

def main():
    """ Main function. """

    # Create an Animal object
    lion = Animal("Lion", "Simba", 5, "Male", 200, "Healthy", "Enclosure 1", "Carnivore", "Meat",
                  ["Morning", "Evening"], "Serengeti")

    # Access animal attributes using getter methods
    print(lion.get_name())  # Output: Simba
    print(lion.get_exhibit())  # Output: Serengeti

    # Modify animal attributes using setter methods
    lion.set_age(6)
    lion.set_exhibit("African Savanna")
    # Print the animal object
    print(lion)

    # Destroy the animal object
    del lion

if __name__ == "__main__": main()
```

- Alex

References:

Yonkeu, S. (2023, April 4). *A guide to Python class attributes and class methods*. A Guide to Python Class Attributes and Class Methods. https://www.turing.com/kb/introduction-to-python-class-attributes