Object Oriented Programming, Exercise 7

Topics: Inheritance, exceptions

Make a Git commit at least after every coding task.

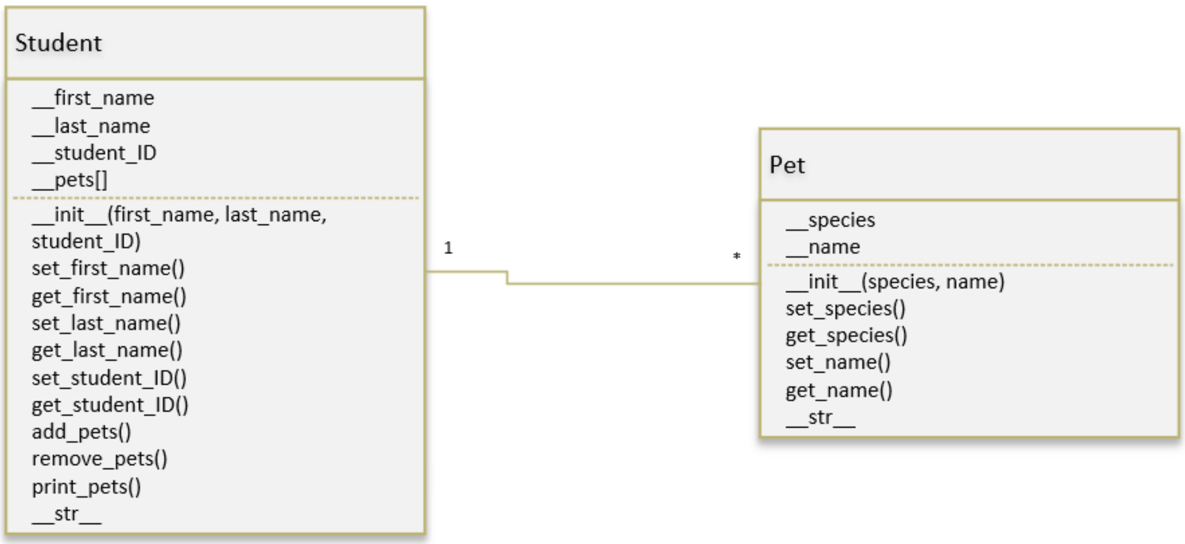
Code in Python3 and follow the style guide.

1 point from every task (1-7) (unless otherwise mentioned)

1. Answer the following questions.
    a. What does polymorphism (in object-oriented programming) mean? Also give a short (coding) example, e.g. google for examples).
    b. What is a class variable and how are they used?
    c. What is an instance variable and how is it different from the class variable?
    d. What is a UML sequence diagram used for?
    e. What is a lifeline in UML sequence diagrams?
2. More theory tasks
    a. Multiple choice
        i. In an inheritance relationship, the ______ is the general class.
            1. Child class
            2. Subclass
            3. Superclass
            4. Specialized class
        ii. In an inheritance relationship, the ______ is the specialized class:
            1. Superclass
            2. Master class
            3. Parent class
            4. Subclass
        iii. Let's say we have two classes in our program: BankAccount and SavingsAccount. Which one of them would most likely be the subclass?
            1. BankAccount
            2. SavingsAccount
            3. Neither of them
            4. Both of them.
        iv. Which one of the options you will use if you want to check whether an object is an instance of a class?
            1. The *instance* operator
            2. The *is_object_of* function
            3. The *isinstance* function
            4. There is not a way to check that at all.
        v. Which one of the UML diagrams is a behavioral diagram?
            1. Class diagram
            2. Sequence diagram

        3. Object diagram
        4. Deployment diagram
    vi. Which one of the UML diagrams is a structural diagram?
        1. Use case diagram
        2. State machine diagram
        3. Activity diagram
        4. Composite structure diagram
    vii. In UML class diagrams, what does the notation * mean.
        1. Multiplication operation
        2. Power of operation
        3. Multiplicity 0..n
        4. Multiplicity 0..1
  b. True or false?
    i. It is not possible to call a superclass's __init__ method from a subclass's __init__ method.
    ii. A subclass never inherits any methods or attributes from the superclass.
    iii. A superclass can inherit methods from subclass, if they have been denoted with *pass_to_super* function.
    iv. In a subclass it is possible to have methods and attributes in addition to those that the subclass inherits from superclass.
    v. In Python, multiple inheritance does not exist.
    vi. Aggregation and composition shall never be used in UML class diagrams.
    vii. Aggregation and composition mean exactly the same thing in UML class diagrams.

3. Implement the following UML diagram. Try to figure out the best way to have animals in appropriate data structure in the Student class (see the link in Exercise 5 Task 7, there is an example of Deck class creating a deck of cards, pay attention to the relationship between the Deck class and Card class and think how that information can be applied in the relationship between the Student class and Animal class). Pet Class can also be called Animal/Domestic animal (you most likely have implemented that in previous exercises). Think (carefully), do you need to have the owner of the pet as e.g. data attribute in the Pet/Animal class (in order to make the relationship between Student and Pet). If yes, add that. To clarify: Student and Pet are classes. Then create a few objects (=instances of the classes). Print out the states and information, who is the owner of the pet and which pets (if any) a student has. (2 points)



4. Add a class to the class diagram of task 4: Class is car (which you most likely have already implemented). Think of the multiplicity (can a student have more than one car, can a car have more than one owner or not)? Hint: see the Student <> Pet association multiplicity... You can use any SW you like for drawing UML diagrams or even pen and paper (and take a picture of it to return document).
5. Code the relationship you just drew in the class diagram. Then check if the pet fits into your car or do you need a truck or a trailer. Give informative output prints. See which pet-related methods are in the Student class and think which car related methods you need to add to the Student class. (2 points)
6. Still practicing the use of dictionary. Implement a simple quiz where the user is asked the capitals of countries. First, make a **text file** with at least 50 countries with their capitals. The information is read at the beginning of the program into the dictionary. In the quiz itself, the user is asked the capitals of ten countries. When correctly answered by the user, proceed to the next question. If the user answers incorrectly, they will be shown the correct answer and then proceed to the next question. After ten questions, the user is informed of the number of correct answers. (1 point)
    a. Use exception handling (= poikkeuskäsittely in Finnish) in checking, that the file exists. (1 point)
7. Draw a simple class diagram (at least one inheritance and one association relationship with multiplicity), a few data attributes and methods and implement the diagram. Use informative output prints. You can use any SW you like for drawing UML diagrams or even pen and paper (and take a picture of it to the return document).