Assignment 3 part II - INF-1400

Sigurd Uhre

April 14, 2021

1 Questions

1. What is the difference between a class and and an object?

We can describe an object as a collection of data with associated behaviors. For example: circles, rectangles and polygons are three different shapes that we can draw in our game. We now have three different kinds of objects, to distinguish them from one another we use the term class to separate "this kind of object" from others. We can now say that we have three classes of objects. Classes describe objects. We can look at classes as blueprints for creating an object. If we put the three different circles on the screen, each shape is a distinct object with different appearance, but all three have the attributes and behaviors associated with one class: The general class of circles. [1]

2. What is inheritance? What is the Python syntax for inheritance?

In OOP, one class can inherit attributes and methods from another class. We often refer to inheritance classes as child-class if it is inheriting from another class. The class that provide the inherited properties is often referred to as parent class. The point of this set up is so that the child class can inherit properties from the parent class and in addition add new features specific for this child-class. This result in re-usability of code. The goal is to find features that is desired for different classes and put them in one parent class. In this way we don't need to write the same "basic" properties for each class, we just inherit it from the parent-class. The syntax of inheritance classes is illustrated below in figure 1. [1]

```
4
5   class parenClass():
6       body of parent class
7
8   class childClass(parentClass):
9       body of child class
10
```

Figure 1:

3. What is the difference between a has-a and an is-a relationship?

The "is a" relationship is fully based on inheritance. This can either be of two types class inheritance or interface inheritance. In other words its like saying " A is a B type if object". For example, we can say house is building, but a building is not necessarily a house. If there is an extends keyword or implemented keywords in the class declaration, then this class is said to have is-a relationship. [2]

The has-a composition means the use of instance variables that are referenced to other objects. For example: The car Opel has an Engine, or the House has Bathroom. We can use the car example to illustrate these concepts below: [2]

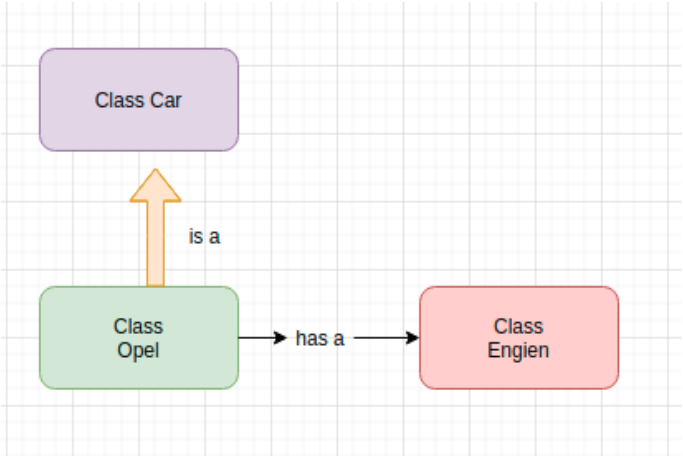

Figure 2:

4. What is encapsulation? How is encapsulation handled in Python?

The idea of encapsulation is wrapping data and the methods used on the data inside one unit "in the same capsule". In this way we get restriction on the access of specific variables and methods directly, making the probability of accidental modification of data less likely. Changing the objects method by changing an objects variable is the only way to make a change. We often refer to those variables as private variable. Encapsulation in python is done by organising variables, methods, objects etc.. in classes. [3]

5. What is Polymorphism? Give examples of polymorphism from the pre-code and the mayhem implementation?

Polymorphism is the ability treat a class differently, depending on which subclass is implemented. we can say that polomorphism means same function name, but with different signatures being used for different types. Polymorphism is a concept rather than one specific method. We can apply polymorphism by using one inbuilt "polymorphism" function such as "len()". This function "does the same job" every time but can be applied differently so that i deliverers different outcome: [2]

```
14  #Python program to demonstrate inbuilt polymorphic functions
15
16  #len() being used for a string
17  ptint(len("mayhem"))
18
19  #len() being used for a list
20  print(len([10, 20, 30]))
21
22  #Output
23  6
24  3
25
```

Figure 3:

We can also use polymorphism with class methods by creating a loop that iterates through a tuple of objects. We can then call the methods without being concerned about which class type each object is. We assume that these methods actually exist in each class, shown in figure 3 below. [2]

```
27
28  class Norway():
29      def capital(self):
30          print("Oslo is the capital of Norway")
31
32  class Sweden():
33      def capital(self):
34          ("Stockholm is the capital of Sweden")
35
36  obj_nor = Norway()
37  obj_swe = Sweden()
38  for country in (obj_nor, obj_swe):
39      country.capital()
40
41
42  #Output
43  Oslo is the capital of Norway
44  Stockholm is the capital of Sweden
45
```

Figure 3:

We can also use polymorphism with inheritance. Polymorphism allows one to create methods in the child class that have the same name as the parent class's methods. This is useful when the method inherited from the parent class don't fit the desired outcome of the child class. We can say that we are re-implementing the method in the child class, this method is often referred to as "method overriding".

An example of polymorphism in the precode is the normalize function. Is it used twice in the intersecting and for different purposes. In the mayhem implementation the function $get_rectisusedindifferentpurposes.$

References:

1.$Phillips, D., 2018. Python3Object-OrientedProgramming-ThirdEdition.$ Place of publication not identified : $PacktPublishing.$

2.$GeeksforGeeks.2021.PolymorphisminPython-GeeksforGeeks.[online]Availableat :< https://www.geeksforgeeks.org/polymorphism-in-python/ > [Accessed14April2021].$

3.$GeeksforGeeks.2021.EncapsulationinPython-GeeksforGeeks.[online]Availableat :< https://www.geeksforgeeks.org/encapsulation-in-python/ > [Accessed14April2021].$

4.w3resource.2021.Inheritance(IS−A)vs.Composition(HAS−A)Relationship−w3resource.[online]Availableat :< https : //www.w3resource.com/java − tutorial/inheritance−composition−relationship.php > [Accessed14April2021].