Questions

Daniel Elisabethsønn Antonsen

April 07 2022

What is the difference between a class and an object?

Classes are building blocks which leads us to object-oriented programming, they gives a definition of how an object works. They gives both the interface and the implementation. It is a user-defined datatype, containing its own data members and functions. We can say that a class is sort of a blueprint for an object. It is used to organize the information or data.

Objects are instances of classes. All data members and member functions can be accessed from objects. An object consists of normally: A name, member data or attributes. That is, data that describes the object such as for example color, position, size, etc. And member functions or methods, functions that are related to the object that describes the object. This could be for example the motion of the object, etc.

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

Inheritance is an important concept in object-oriented programming. It is the concept where one class inherit attributes and methods from a parent class. The child class has the same interface as the parent class, but can extend the interface if it is necessary. The concept provides re-usability when coding, and a lot of repeating of code will be unnecessary when we implement code using inheritance.

The Python syntax for inheritance is demonstrated here

Listing 1: Python Inheritance

```
class University:
    # Parent class
    def__init__(self, name, age, major):
        self.name = name
        self.age = age
```

```
class Student(University):
    # Child class
    def __init__(self, name, age, major):
        super().__init__(name, age)
        self.major = major

STA = Student("Alfred", 20, "Computer Science")
print(STA.name)
```

Here the child class inherit from the parent class by writing the name of the parent class in the "input" of the child class and calling the constructor from the parent class with *super().__init__()*. We can skip the *__init__()* in the child class, then the child class will inherit the *__init__()* form the parent class. Callin git with super().__init__() both calls te contructor from parent class and extends it by it own.

What is the difference between a has-a and is-a relationship?

With the is-a relationship, this is based on inheritance. It is essentially a one-way street, that means for example a house is a building, but a building is not a house.

Has-a relationship or compasition just simply mean the use of instance variable that references to an other object. For example, a house has a bathroom.

What is encapsulation? How is encapsulation handled in python?

Encapsulation is the concept of bundling data and methods into one class, like it is done in Java programming language. But it also refer to the information hiding. That is the concept of hide the internal state of an object from the outside. You can make so that attributes is not visible from the outside, and essentially hide specific information and control access to the internal state. This can also be done with methods, where we can use the idea of setters and getters. The getter method retrives an attribute and an setter method changes it.

The syntax for Python is the convention of using underscores to indicate the encapsulation of attributes. A dobbel underscore indicates that we want the attributes to stay private and a single underscore indicates that we to say "please not access this attribute". But in reality if you know what to do, you can get access to the attributs either way. So the encapsulation in Python is not the best amoung the various amount of programming languanges.

What is polymorphism? Give examples of polymorphism from the precode and the Mayhem implementation?

Polymorphism is a concept that refers to an objects abillity to take on multiple forms. It happens when multiple objects inherit from the same base class. Such as if we have a base class/parent class Animals, and we have child classes/-subclasses Dog, Fish, etc. Which all inherits from Animals. The Animals class have a method Move() which all the subclasses inherits, but all the subclasses can have different way of implementing the method. When the method is called from the Dog class it may stride across the screen. But when the same class is called from the Fish class, if swims across the screen. And so on.

We have som instances of polymorphism in this assignement. All the objects that are illustrated on the screen (spaceships, obstacle, bullets and platforms) all inherit from the pygame.sprite.Sprite object, which contains a update() method. They all inherit from the parent class and when the method is called it does different actions for every child class. For the platform and obsticle class, the update() method blits the image on the screen, but for the bullet and player classes the update() method also contains actions such as movement of the objects.