INF-1400: Object-oriented programming
Assignment 3 - Part II

Trym Varland

Git: tva050

UiT - Norges Arktiske Universitet

April 12, 2023

Questions

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

We can think of a class, as a template to create objects. The class contains attributes and methods, which describes the behavior of the object. So then we can say that, an object is the instance of the class. So for example, under we have a class called `"Person"` with two attributes `"name"` and `"age"`, it also have a method called `"name_age"`. We than create a object by calling on the `"Person"` class, where `"person1"` is the object of the class.

```
class Person: # Class with two attributes and one mathod
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def name_age(self):
        print("The name is " + self.name + " and is "+ self.age +
            "years old.")

person1 = Person("Myrt", 34) # object of the Person class

person1.name_age() # output: The name is Myrt and is 34 years old.
```

Listing 1: Class and object example

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

Inheritance, is what it says. A class (child class) can inherits properties and methods, from a already existing class (parent class). it is also possible to override the existing ones from the parent class, by adding new properties and methods in the child class.

```
class Parent:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def name_age(self):
        print("The name is " + self.name + " and is "+ self.age +
```

```
            "years old.")

class Child(Parent):
    def __init__(self, name, age, birth_year):
        super().__init__(name, age): # inhertis all params and
    methods
        self.birth_year = birth_year

    def birth_year(self):
        print("Name: " + self.name + "\n Age: " + self.age + "\n
    Birth year: " + self.birth_year)

person1 = Child("Myrt", 34, 1989)
person1.birth_year()
"""
output:

Name: Myrt
Age: 34
Birth year: 1989
"""
```

Listing 2: Python syntax for inheritance

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

The difference between a has-a and an is-a relationship, is that a has-a relationship describes the relationship where one object has another object as part of it's self, often implemented with e.g aggregation, where a class have instance of a another class as a variable. E.g a class called `Car` has a engine, the engine can be represented as it own `Engine` class. So the `Car` has a `Engine`. An is-a relationship is where we have inheritance, so a class is a subclass of another class which inherits all the attributes and methods. E.g we have a class called `Animal`, than we can create a subclass of animal for example a class `Cat` which inherits attributes and behaviors of from the `Animal` class. So than we can say that `Cat` is a `Animal`.

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

Encapsulation is when we secure some variables and methods that works on some type of data, which works inside one unit. This means, that you can only accesses this through public methods and is hidden from outside. In python a class is a example of encapsulation, where all methods, variables and so on, which lay in side of the class (encapsulated).

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

Polymorphism is when objects from different classes is allowed to be used with one another (vice versa), although they have different implementation of the same attributes and methods. That will say, a method in a child class, have equal name in the parent class.
In the Mayhem implementation, polymorphism is not explicit used. But polymorphism is that a object have the ability to take on any forms or have many different types. So in the program we have the class called `Spaceships`, which represent any spaceship object created from this class. Each spaceship object will have the same attributes and methods, but they can have different values for those attributes and methods. So that will say when we create a new spaceship object using the `Spaceships` class, we pass in different values for the image, position and velocity, that will create a new spaceship object but they will still have the same attributes and methods. Which comes clear when the game is running, since we see that the two players are behaving different (considering position, velocity and so on). This is called object polymorphism.