Object Oriented Programming

Svarny Petr

Katedra logiky FF UK

16. března 2021

Overview

Připomínka OOP

Python

C++

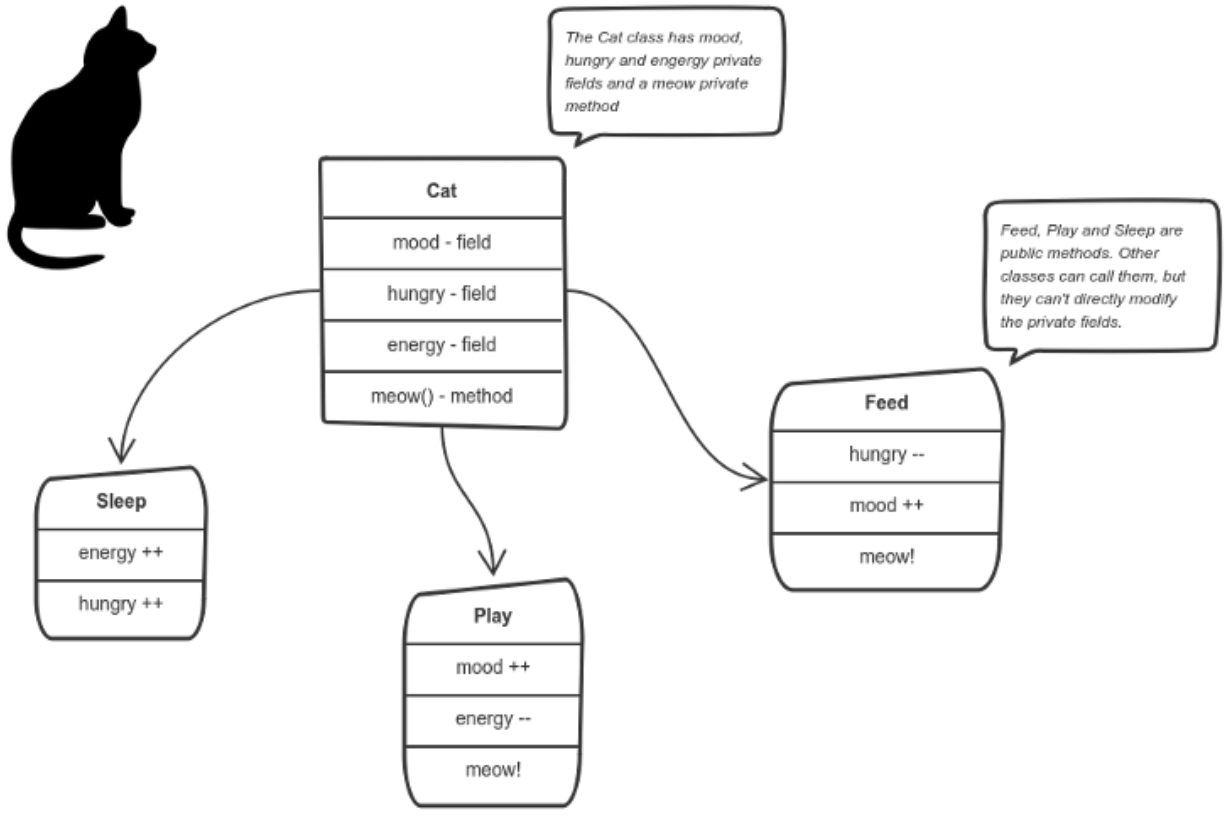
JAVA

Kotlin

Visual Basic

ORK

OOP



=Concepts=

=Hate=

Reminder

- public vs. private
- constructor and destructor
- attributes and methods

Don't Repeat Yourself (DRY)

- inheritance
- polymorphism
- static methods
- factories

Python

=RealPython tutorial=

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

Private Python

=GeeksforGeeks=

```
self._something
>> class._something

self.__something
>> class._Class__something
```

Inheritance

=W3School=

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

class Dog(Animal):
    pass
```

Polymorphism

=GeeksforGeeks=

```
class Cat:
    def __init__(self):
        self.noise = 'Mnau'

    def make_noise(self):
        print(self.noise)

class Dog:
    def __init__(self):
        self.noise = 'Haf'

    def make_noise(self):
        print(self.noise)
```

Static

=RealPython=

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

    @staticmethod
    def make_noise():
        print('Haf haf')
```

Factory

=RealPython=

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

    def __repr__(self):
        return f'This is a {self.breed}.'

    @classmethod
    def pug(cls, name, age):
        return cls(name, age, 'pug')
```

```
>>> Dog.pug("MrScruffle", 4)
This is a pug.
```

C++

=W3Schools=

```
class Dog {
    public:
        void setAge(int s) {
            age = s;
        }
        void setName(string n) {
            name = n;
        }
    protected:
        int age;
        string name;
};
int main(void) {
    Dog Doggo;
    Doggo.setAge(5);
    Doggo.setName("Alik");
    cout<<Doggo.name;
```

JAVA

=W3Schools= “I see classes everywhere.”

```
public class Example{
    public static class Dog {
        int age = 5;

        String name = "Alik";
    }

     public static void main(String []args){

        Dog myDog = new Dog();

        System.out.println(myDog.name);
     }
}
```

JAVA

=Oracle=

```
public class Dog {
  public int age = 5;
  private String name = "Alik";
}
```

Kotlin

=Kotlinlang=

```
fun main() {
        class Dog(var age: Int, var name: String) {
    }
        val my_dog = Dog(5, "Alik")
        print(my_dog.name)
}
```

Visual Basic

=Microsoft=

```
Public Class Dog
    Public Property Name As String
    Public Property Age As Integer
End Class

Dim myDog As New Dog
myDog.name = "Alik"
```

Objekts R Kool

https://esolangs.org/wiki/ORK

```
When this program starts:
There is a scribe called Writer.
Writer is to write "Hello, world!"
```

OOP Advanced topics

- =Multiple inheritance=
- =Overloading=