Въведение в ООП
Класове

Основни понятия

- Class -> Представлява моделиран обект от реалния свят Например човек, къща и т.н.Всеки клас има полета -> атрибути на класа и функции на класа. Това е основната конструкция на клас за всеки ООП език – Python, C++, Java.В императивното C/C++ най близкото до клас е структура
- Обект -> Представлява променлива от тип на класа. В C/C++ най близкото до обект преди ООП е променлива от тип структура. Обекта се нарича инстанция на класа.

Как е в друг език - Java

- Penson.java MainClass.java
- Person mitko=new Person("Mitko",30,);
- Mitko.name=”Mitko”;
- Mitko.age=30;
- System.out.println(miko.name);
- class Person{
- Char[30] name;
- Int age;
- }

Основни понятия

Конструктор - > функция която получава като входен параметър външни стойности и в тялото на конструктора полетата на класа приемат като стойност входните параметри на конструктора.

Setter - > функция която получава външна стойност и съответната стойтост на полето става равна на тази стойност.

Getter - > функция която връща като стойност стойността на съответното поле.

Setter и Getter служат за да могат стойностите на полетата да се достъпват от външен клас.

Пример за обект

Нека разгледаме h=”hello”

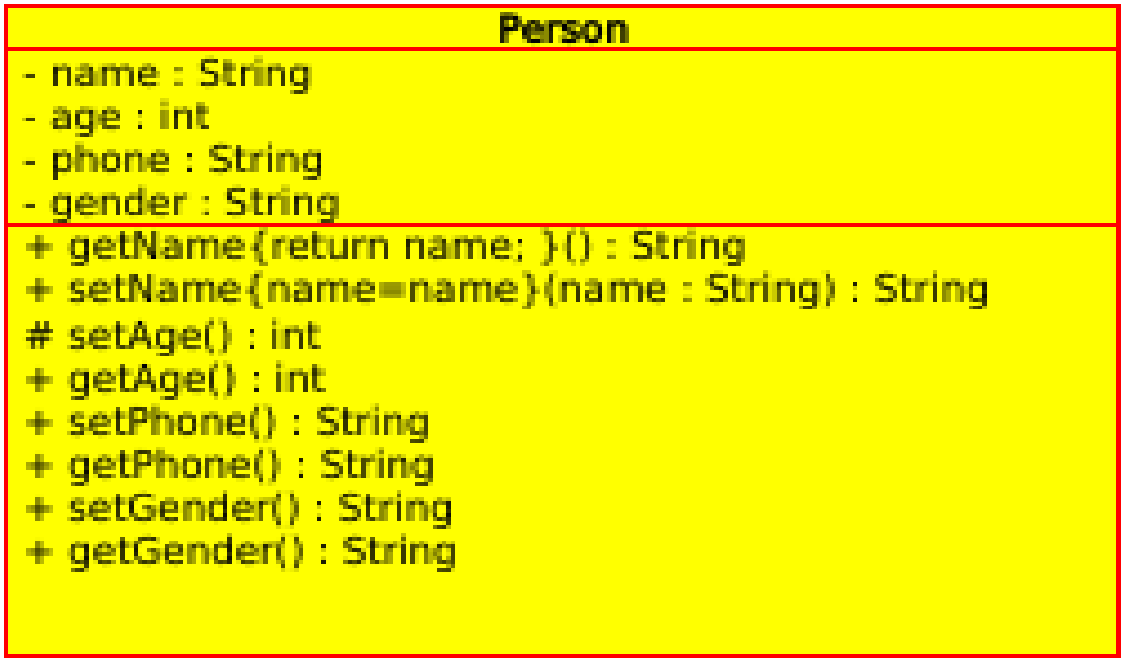
От императивна гл. т. това е променливата h.

Вече разгледахме че от гл. т. на хардуера в клетка от паметта се заделя място за “hello”.

А от гл. т. на ООП h представлява обект. В Python типовете на променливите int, double… са клас.

Общ вид на клас

Нека разгледаме клас Person.



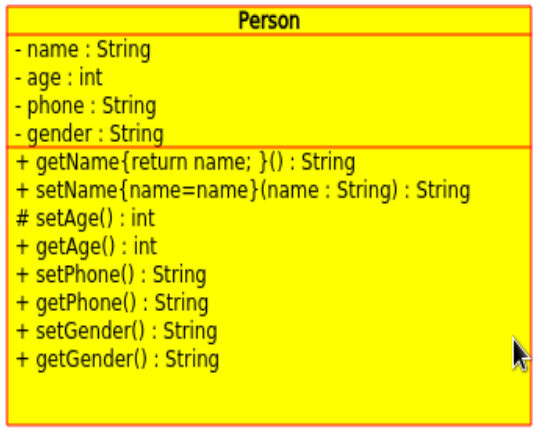
set()
Get()
private String name;
Private int age;

person — Umbrello UML Modeller

File Edit Diagram Code Settings Help

New Open... Save Cut Copy Paste Find...

Person



Ready. 100% Fit 100%

Дефиниране на клас

```
class Human:
    #constructor
def __init__(self,name,age,phone):
        self.name=name
        self.age=age
        self.phone=phone


   def printName(self):
            print(self.name)
    def printAge(self):
                print(self.age)
    def printPhone(self):
            print(self.phone)
p1 = Human("Pesho",18,"0899")
print(p1.printName())
print(p1.printAge())
print(p1.printPhone())
```

Rpi class

```
class RPi:
    def __init__(self,model,cpu,mem,ports):
        self.model=model
        self.cpu=cpu
        self.mem=mem
        self.ports=ports
rasp1 = RPi("3","ARM Cortex-72","4GB RAM","USB2.0/3.0 LAN")
Rasp2=RPi(“4”,”ARM”,”8GB RAM”,”USB 3.0 Flash SSD”)

print(rasp1.model)
print(rasp1.cpu)
print(rasp1.mem)
print(rasp1.ports)
```

Rpi Getter and Setter

```
public class Rpi:
Private String model;
Private String

    def __init__(self,model,cpu,mem,ports):
        self.model=model
        self.cpu=cpu
        self.mem=mem
        self.ports=ports
    def setModel(self,model):
        self.model=model
    def getModel():
        return model
rasp1 = RPi("3","ARM Cortex-72","4GB RAM","USB2.0/3.0 LAN")
print(rasp1.model)
print(rasp1.cpu)
print(rasp1.mem)
print(rasp1.ports)
Public class Mai{
Rpi 1= new Rpi;
1.setModel(“3”);
1.setCpu(cortex);
}
```