

Object Oriented Programming

Key Definitions:

Keyword	Definition
Class	A blueprint for creating objects that defines attributes (data) and methods (behaviour).
Object	An instance of a class with specific values for its attributes.
Encapsulation	Hiding an object's data and only allowing controlled access through methods (getters and setters).
Inheritance	A child class inherits attributes and methods from a parent class, promoting code reuse.
Polymorphism	The ability for methods to take multiple forms, either by method overloading or method overriding. Method Overriding - When a child class redefines a method inherited from the parent class to change its behaviour. Method Overloading- When multiple methods in the same class have the same name but different parameters
Abstraction	Hiding complex implementation details and exposing only essential features to the user.
Constructor	A special method that is automatically called when an object is created, used to initialise attributes.
Getter Method	A method that retrieves (gets) the value of an attribute.
Setter Method	A method that updates (sets) the value of an attribute.
Public	Allows attributes and methods to be accessed from anywhere.
Private	Restricts access to attributes and methods within the same class.
Protected	Allows access within the class and its subclasses.

Below is an example of a Pet Class and inheritance

Inheritance allows a class to inherit properties and behaviours from another class, reducing code duplication.

Class: Pet

```
private petName: string  
private petType: string  
private petAge: int
```

```
Constructor new (pName,pType,pAge)  
getName()  
setName(pName)
```

```
class Pet()  
    private petName  
    private petType  
    private petAge
```

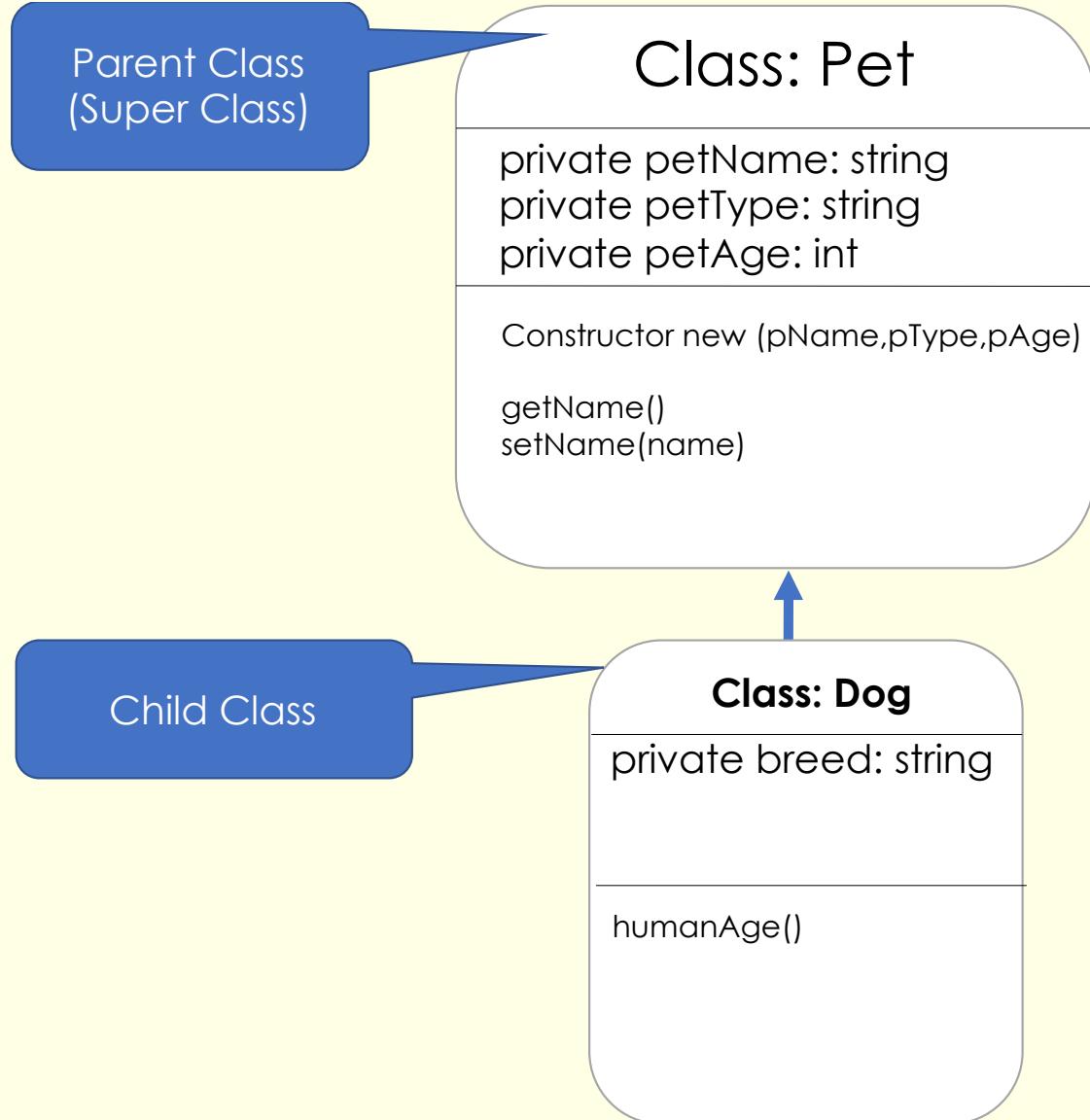
```
#Constructor  
public procedure new(pName, pType, pAge)  
    petName = pName  
    petType = pType  
    petAge = pAge  
end procedure  
#Getter methods return the value of an attribute  
public function getName()  
    return petName  
end function  
# Setter methods let you change the value of an attribute  
public procedure setName(pName)  
    petName = pName  
end procedure
```

```
#main program  
# pet object  
myPet = new Pet("Buster","dog",5)  
#Access an attribute  
print(myPet.getName())  
#Change an attribute  
myPet.setName("Bob")
```

Constructor method

To create an instance of an object from a class

Encapsulation is the concept of restricting direct access to an object's data and allowing interaction only through getter and setter methods.



class dog **inherits** pet
 private breed
 """"constructor class""""
 public procedure new (pName, pType, pAge, pBreed)
super(pName, pType, pAge)
 breed = pBreed
 end procedure

public procedure humanAge()
 humanAge = petAge * 7
 print(humanAge)
 end procedure
 end class

Class: Pet

```
private petName: string  
private petType: string  
private petAge: int
```

```
Constructor new (pName,pType,pAge)
```

```
getName()  
setName(name)  
display()
```

Class: Dog

```
private breed: string
```

```
humanAge()  
display()
```

Polymorphism

Polymorphism lets the same method name work in different ways, either in different classes or with different parameters in the same class.

Method Overriding

A child class replaces a method from the parent class with its own version.

Parent Class – Pets

```
procedure display()  
    print( "Name", petName)  
    print("Type", petType)  
    print("Age", petAge)  
end procedure
```

Child Class – Dogs

```
procedure display()  
    print( "Name", self.name)  
    print("Type", petType)  
    print("Age", petAge)  
    print("Breed", breed)  
end procedure
```

Object Oriented Programming

Below is an example of a Pet Class:

Object pseudocode	Explanation	Python Code
class Pet	A class will contain class variables, constructor method and methods	class Pet(object):
private petName private petType private petAge	Define the class variables: public or private Private – means only accessible within the class Public – means outside the class Protected - means can be shared with any child classes	Not required in python
public procedure new(pName, pType, age) petName = pName petType = pType petAge = pAge end procedure	Constructor method for pets class This is what will happen when you create a new instance of pet (known as an object) Must use correct declaration appropriate name and new Include any parameters required for creating a new object	def __init__(self,pName,pType,pAge): self.petName = pName self.petType = pType self.age = pAge
public function getName() return petName end function public procedure setName(pName) petName = pName end procedure	Method are usually public. It can be a procedure or function (return something). getter methods are used to return an objects attribute setter methods are used to change an attribute	def getName(self): return self.name def setName(self,name): self.name = name
myPet = new Pet("Buster", "Dog", 3)	This is used to create a new instance of Pet (class instantiation) in the main program	myPet = Pet("lily", "dog", 3)
print(myPet.getName())	This is how you use a method within a class	print(myPet.getName()) myPet.setName("Fred")

Inheritance

OOP pseudocode	Explanation	Python Code
class Dog inherits Pet	The dog class inherits the methods and attributes from Pet The Pet class is the Parent class (superclass) The Dog class is the child class	class Dog(Pet): """pet class""""
Class Dog inherits Pet private breed #Constructor Method public procedure new(pName, pType, pAge, breed) super (pName,pType, pAge) breed = breed end procedure end class	In the child class, you can declare new class variables, in this case , dogbreed In the constructor method list all the parameters and then in the method to refer to attributes inherited form the parent class use the <i>super.inheritedvariable</i>	class Dog(Pet): def __init__(self,pName,pType,pAge,breed): super().__init__(pName,pType,pAge) self.breed = breed
private procedure humanAge() humanAge = petAge * 7 print (humanAge) end procedure	This method private as it is only to accessed within the Dog class	def humanAge(self): humanAge = self.petAge * 7 print(humanAge)
myDog= new Dog("Buster", "Dog", 3, "Border terrier")	Main Program: Create a new instance of dog using all the constructor parameters.	myDog = dog("Buster","Dog", 4, "Border terrier")
print(myDog.getName()) myDog.humanAge()	In the main program, you can then use methods in the dog class and pet class.	print(myDog.getName()) mydog.humanAge()