Insights into OOPS

Object oriented programming is about creating objects that contain both data and methods together.Methods are used to act on the data.OOPS is faster and easier to execute.It provides a clear structure for the programs.It basically provides security, reusability and readability for the programs.

Features of OOPS

Polymorphism- It is one of the features of OOPS.Here, the word is divided into two “Poly” which means “many” and “morphism” means “forms”.Polymorphism is divided into two types, the first one is method overloading and the second one is method overriding.Real life example could be a person. For example: A person can be a son, father, grandfather that means he can take so many forms.

Method overloading- Method overloading is the concept or the technique that can have the methods of same name in a class.It follows three conditions , the number of parameters should be different in all the methods of a class, the data type of the parameter should be different in all the methods of a class and the sequence of the parameter should be different for all methods in a class.If these three conditions are not fulfilled then it will not be a method overloading.Real life example can be a calculator which can be operated and have two functions named Add ,which can take two and three parameters respectively of same method name Add .

```python
class abc:
def Add(self,a,b):#function1
x=a+b
return x

def Add(self,a,b.c):#function2
x=a+b+c
return x
 obj=abc()
print(obj.Add(2,3))#first function is called
print (obj.Add(5,6,7))#second function is called
```

Method overriding-Method overriding is the concept which always deals with two classes.In this, the two classes can be as Parent class and the child class.Here, the implementation of the parent class is overridden with the child class.

```python
class Parent:#parent class
def __init__(self):
self.value="Parent class"
def show(self):
print(self.value)
 class Child(Parent):#child class
def __init__(self):
self.value="Child class"
def show(self):
print(self.value)
obj1=Parent()#object for parent class
obj2=Child()#object for child class
obj1.show()#method show for Parent in called
obj2.show()#method show for Child in called
```

Difference between method overloading and method overriding is that method overloading is a compile time polymorphism and method overriding is the run time polymorphism.

Inheritance-Inheritance is the feature of OOPS.It is defined as the process in which the child class acquires all the properties and behaviours of its parent class.It is basically for reusability.OOPS has two forms of relationship “has-a” , weak relationship and “is-a” ,strong relationship.Inheritance falls under ”is-a” relationship .There are many types of inheritance.

Single inheritance-Here,the child class acquires the properties and behaviours of the parent class.

```python
class Animal:#parent class
def speak (self):
print(“ ”)
class Dog(Animal):#child class
def bark(self):
print(“”)
d=Dog()#object for child class
d.bark()
d.speak()
```

Multilevel inheritance-Here,the child class will inherit a parent class and this parent class acts as a child class for the other parent class above.

```python
class Animal:#first class
Def speak (self):
print(“ ”)
class Dog(Animal):#second class
Def bark(self):
print(“”)
class Cat(Dog):#third class
Def meo(self):
print(“”)
d=Cat()
d.bark()
d.speak()
d.meo()
```

Multiple inheritance- The child class inherits multiple parent classes.Example can be hybrid dogs.

Hierarchical inheritance-Here,more than one child class inherits the single parent class.

Hybrid inheritance-It is the combination of two inheritance.

This parent child can be a real life example for inheritance.

Encapsulation-The data and the behaviour wrapped in a single unit is called encapsulation.It is the feature of OOPS.For instance,the class is an example of encapsulation where all the data members,member functions are in a particular class.

Abstraction-Abstraction is the concept where it will only show the functionality detail and hides the implementation details.if i take a real life example that suppose there is a car and it runs with the help of key ,but we do not know how it is working, we just know that it is working.

What are classes and objects?

Class

Class is a user defined data type.It is a logical entity.It has no space in memory.It does not depend on object,so class is independent entity.It is template for objects.It is a blueprint for creating objects.Class contains only a keyword class with an identifier.It contains the

methods, variables and classname.It is blueprint which defines some properties and behaviour.The syntax for class is" class hello: " in python.In real world example ,the class can be considered as a "Car".

Object

An object is an instance of a class.Object is a physical entity.Object depends on class, so object is dependent entity.Objects are allocated memory space.The real life example of an object is" baleno","Alto","XUV" .So these examples are the name or the brand of cars that comes under the "Car" class.The syntax to create object is objectname=classname().for example hello=Person().

Self and __init__

In python, the self keyword is always passed as parameter to a function.The main purpose of self is that it links the values passed while creating the object to the parameters passed in the function .in other words, self represents the current instance of a class and binds the attributes with the given arguments.The syntax for the self is def _init_(self,name,year): Here __init__ is used to initialise objects of a class.It is also called as a constructor in python.The syntax for constructor is def __init__(self):

Constructor

Since __init__ is a constructor.By constructor,we mean that it is a special type of a method that is used to initialise the object.here in python,we have default and parameterized constructor.

Default constructor-It is a type of constructor where no parameters are passed to function.For instance, def __init__(self):

```python
class Hello:
def __init__(self):# default constructor
 self.val="Snehal"
def result(self):
print (self.val)
d=Hello()
d.result()
```

Parameterized constructor-It is a type of constructor where the function accepts a number of parameter.For instance,def __init__(self,a,b): Here, a and b are the two parameter that are passed in a function which is called a parameterized constructor.

```python
class Hello:
def __init__(self,a,b):#parameterized constructor
 self.first=a
self.second=b
def calculate(self):
self.answer=self.first+self.second
def result(self):
print ("The first number:"+str(self.first))
print ("The second number:"+str(self.second))
print ("The answer is:"+str(self.answer))
d=Hello(5,6)
d.calculate()
d.result()
```

Advantages of OOPS

Enables code reusability-The concept of inheritance can enable code reusability.
Code maintenance-Due to object oriented, code is highly maintained.

Use of OOPS

The object oriented makes the code easier to execute and makes testing and debugging easier.

Summary

OOPS is a programming paradigm that focuses on objects rather than functions unlike procedural oriented programming language like C.It allows for better organisation and encapsulation of code, promotes reusability, and can improve the maintainability of our programs.