Static method

Static method is a global method, which define individual operation an operation which does not use class level variables and object level variables.
This method is bind with class name can be called without creating object.

Syntax:

```
@staticmethod
def <method-name>(arg1,arg2,arg3,…):
        statement-1
        statement-2
```

Example:

```
class Math:
   @staticmethod
   def factorial(num):
      f=1
      for i in range(1,num+1):
         f=f*i
      return f

   @staticmethod
   def power(num,p):
      return num**p

def main():
    res1=Math.factorial(4)
    res2=Math.power(4,2)
    print(res1,res2)
main()
```

Example:

```
class Employee:
    def __init__(self):
        self.__empno=None # OLV/Properties/attributes
        self.__ename=None # OLV/Properties/attributes
        self.__salary=None # OLV/Properties/attributes
    def read_emp(self):
        self.__empno=int(input("EmployeeNo"))
        self.__ename=input("EmployeeName")
        self.__salary=float(input("Salary"))
    def print_emp(self):
        print(self.__empno,self.__ename,self.__salary)

def main():
    emp1=Employee()
    emp1.read_emp()
    emp1.print_emp()
main()
```

Output:

```
EmployeeNo101
EmployeeNamenaresh
Salary5000
101 naresh 5000.0
```

Example:

```
class A:
    __x=100 # CLV
    @classmethod
    def m1(cls):
        print(cls.__x)



def main():
    A.m1()
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest26.py =======
100
```

Example:

```
class Person:
    def __init__(self,n,a):
        self.__name=n
        self.__age=a
    def printPerson(self):
        print(f'Name {self.__name}')
        print(f'Age {self.__age}')
    def get_age(self):
        return self.__age
    @staticmethod
    def isAdult(age):
        if age>=18:
            print("Adult")
def main():
    p1=Person("naresh",50)
    p1.printPerson()
    a=p1.get_age()
    Person.isAdult(a)
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest27.py =======
Name naresh
Age 50
Adult
```

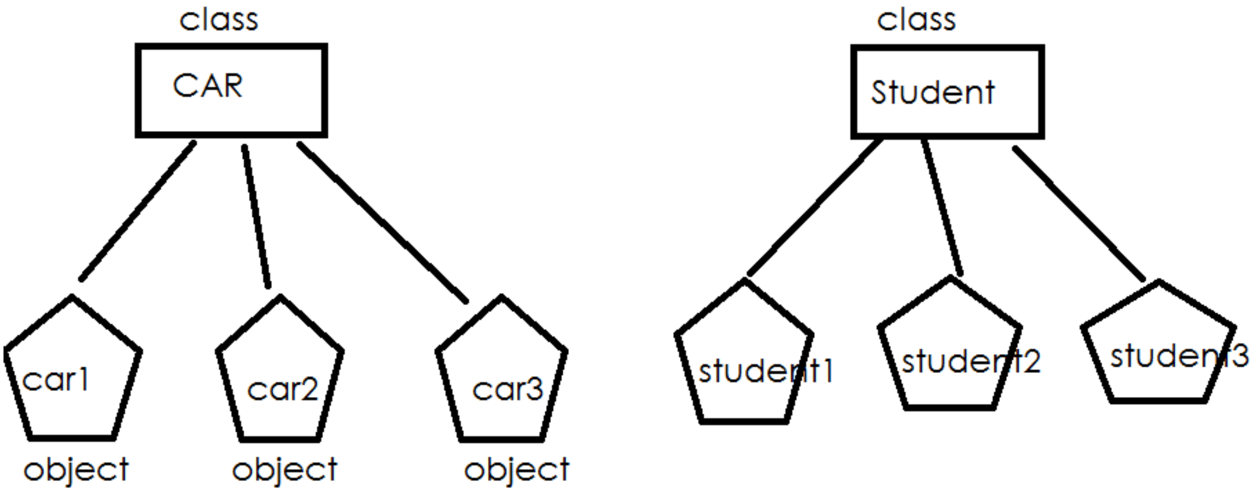
What is difference between object level method, class level method and static method?

Object level method or instance method	**Class level method**	**Static method**
This method is bind with object name and cannot invoked without creating object	This method is bind with class name, it can be called without creating object.	This method is bind with class name it can be called without creating object
It is having first argument as self	It is having first argument cls	It is not having implicit first argument

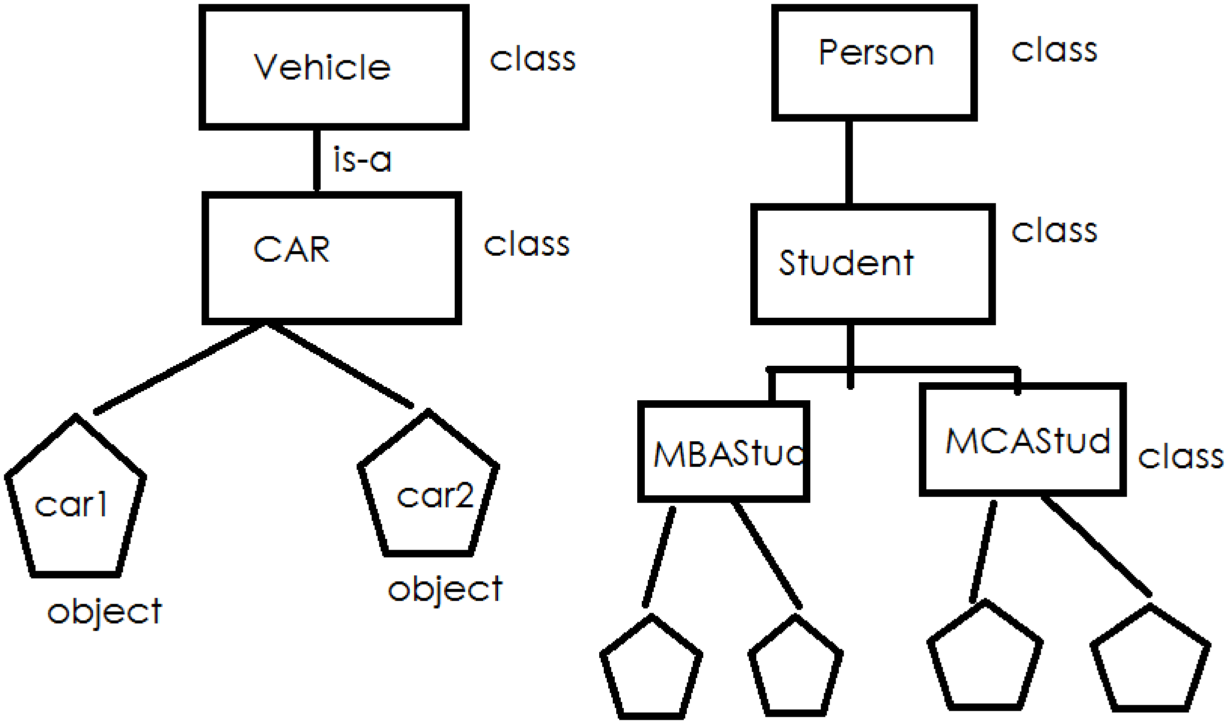
This method can access class level and object level data	This method can access only class level variable	This method cannot access class level data and object level data

Class Reusability

Object oriented application is not developed using one class, it is collection of classes, The content of one class can be used inside another class using different approaches.



Inheritance



Inheritance is a process of acquiring the properties (variables) and behavior (methods) of one class inside another class.

Inheritance is a feature or a process in which, new classes are created from the existing classes.

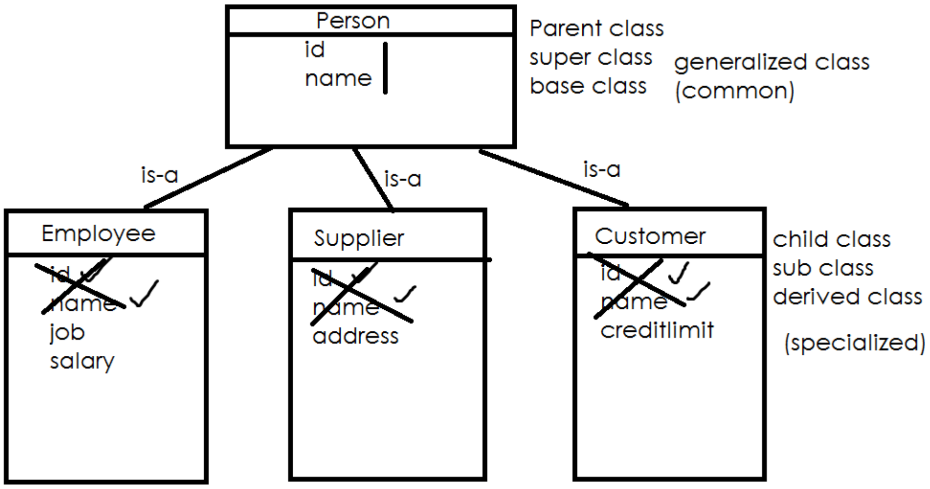
Inheritance allows programmers to create classes that are built upon existing classes.

Advantage of inheritance

The advantage of inheritance is,

1. **Reusability**
   Variables and methods declared inside one class is used inside another class
2. **Extensibility**
   Adding new features to the existing class without modifying it

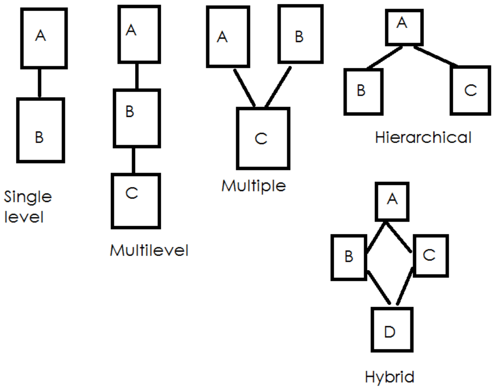
Inheritance is process of creating hierarchy of classes which share common properties and behavior.



Type of inheritances

Based on the reusability of classes or organization of classes there are 5 types of inheritances.

1. Single level inheritance
2. Multi level inheritance
3. Multiple inheritance
4. Hierarchical inheritance
5. Hybrid inheritance



Syntax:

class <derived-class-name>(base-class-name,base-class-name,..):
        variables (instance variables,class variables)
        methods (instance methods, class method, static methods)

1. Methods of parent class are automatically inherited inside child class.

Example

```
class A: # base class
   def m1(self):
      print("m1 of A class")

class B(A): # derived class
   def m2(self):
      print("m2 of B class")

def main():
   objb=B()
   objb.m1()
   objb.m2()
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest28.py ========
m1 of A class
m2 of B class
```

2. Variables or properties of parent class are not inheritance inside child class automatically

In order to inherit the properties of parent class inside child class, child class constructor must call or invoke constructor of parent class.

Example:

```
class A:
   def __init__(self):
      self.x=100

class B(A):
   def __init__(self):
      A.__init__(self)
```

```
        self.y=200


def main():
    objb=B()
    print(objb.y)
    print(objb.x)
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest29.py ========
200
100
```