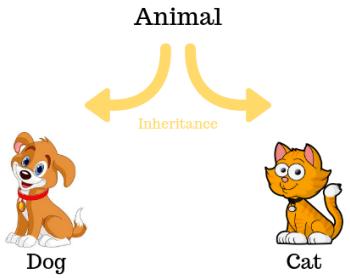
Inheritance

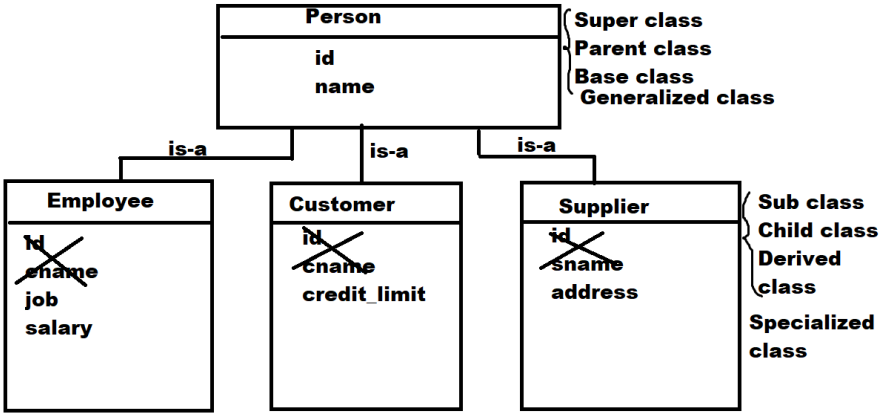
Inheritance is process of acquiring the properties and behavior of one class inside another class.
Inheritance allows developing a new class or data type using existing class or data type.
The relationship between class in inheritance is (IS-A)
IS-A relation is between related classes but not between unrelated classes.
Inheritance allows grouping all the class which share common properties and behavior.



Advantage of inheritance is reusability

1. Variables
2. Methods

It allows reusing the variables and methods of one class inside another class.

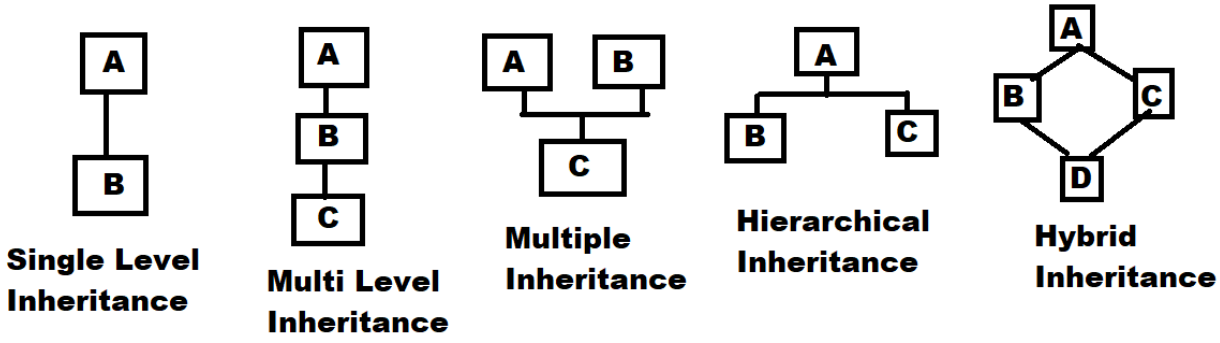


The class having common properties and behavior is called generalized class.

The class derived from generalized class is called specialized class.

Based on the reusability and organization of classes, there are various of types of inheritance

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



Syntax of inheritance

```
class <derived-class>(base-class,base-class,base-class):
        variables
        methods
```

Points to remember

1. Methods of super class/base class/parent class are automatically inherited within sub class

Example:

```
class A: # Base class
    def m1(self):
        print("m1 of A")
    def m2(self):
        print("m2 of A")

class B(A):
```

```
    def m3(self):
        print("m3 of B")
    def m4(self):
        print("m4 of B")

objb=B()
objb.m1()
objb.m2()
objb.m3()
objb.m4()
```

Output

```
m1 of A
m2 of A
m3 of B
m4 of B
```

2. Variables of parent are not inherited automatically within child class.

Example:

```
class A:
    def __init__(self):
        self.x=100
        self.y=200

class B(A):
    def __init__(self):
        super().__init__()
        self.p=300
        self.q=400


objb=B()
print(objb.x)
print(objb.y)
print(objb.p)
print(objb.q)
```

Output
100
200
300
400

What is super()?
super() type, which return an object of super class.
Sub class refers to the members of super class using super() object.

Single Level Inheritance
In single level inheritance there is one base and derived class.

Example:
```
# Single Level Inheritance

class Person:
    def __init__(self):
        self.__name=None
    def set_name(self,n):
        self.__name=n
    def get_name(self):
        return self.__name

class Employee(Person):
    def __init__(self):
        super().__init__()
        self.__salary=None
    def set_salary(self,s):
        self.__salary=s
    def get_salary(self):
        return self.__salary


emp1=Employee()
emp1.set_name("naresh")
emp1.set_salary(50000)
print(emp1.get_name())
```

```
print(emp1.get_salary())
```

Output

```
naresh
50000
```

Multi Level Inheritance

If a class derived from another derived class, it is called multi level inheritance.

Example:

```
# Multi Level Inheritance

class Person:
    def __init__(self):
        self.__name=None
    def set_name(self,n):
        self.__name=n
    def get_name(self):
        return self.__name

class Employee(Person):
    def __init__(self):
        super().__init__()
        self.__job=None
    def set_job(self,j):
        self.__job=j
    def get_job(self):
        return self.__job

class SalariedEmployee(Employee):
    def __init__(self):
        super().__init__()
        self.__salary=None
    def set_salary(self,s):
        self.__salary=s
    def get_salary(self):
        return self.__salary
```

```
emp1=SalariedEmployee()
emp1.set_name("naresh")
emp1.set_job("CEO")
emp1.set_salary(5000000)
print(emp1.get_name())
print(emp1.get_job())
print(emp1.get_salary())
```

Output

```
naresh
CEO
5000000
```

Multiple Inheritance

If a class derived from more than one base class, it is called multiple inheritance.

Example:

```
# Multiple Inheritance

class A:
    def __init__(self):
        self.__x=0
    def set_x(self,x):
        self.__x=x
    def get_x(self):
        return self.__x

class B:
    def __init__(self):
        self.__y=0
    def set_y(self,y):
        self.__y=y
    def get_y(self):
        return self.__y

class C(A,B):  # Multiple Inheritance
```

```
    def __init__(self):
        super().__init__()
        B.__init__(self)
        self.__z=0
    def set_z(self,z):
        self.__z=z
    def get_z(self):
        return self.__z

objc=C()
objc.set_x(100)
objc.set_y(200)
objc.set_z(300)
print(objc.get_x(),objc.get_y(),objc.get_z())
```

Output

```
100 200 300
```

Hierarchical Inheritance

If more than one class derived from same base class, it is called hierarchical inheritance.