# (7.9) Singleton

A class which can only be instantiate once

A class which can have one and only one object

## Uses

**Resource**

File Write

file -> Resoure Acquire

File Write

file -> should not be allowed

**DB Connector**

## 1. DB Connector

In [1]:
```
def connect(*args):
    class A:
        @staticmethod
        def cursor():
            pass
    return A()

# JDBC -> connection -> c1, c2, c3 -> connection.commit()

class DB:
    __instance = None
    def __new__(cls):
        if DB.__instance == None:
            self = super(cls, DB).__new__(cls)
            DB.__instance = self
            return self
        else:
            return DB.__instance
    def __init__(self):
        self.db = connect('hostname', 1234, 'pankaj', 'redhat')
        self.cursor = self.db.cursor()

a = DB() # connection - 1
print(id(a))
b = DB() # connection - 1
print(id(b))
```

```
1866608507056
1866608507056
```

## 2. Assigning one object reference to other object

In [3]:
```
class A:
```

```
    __instance = None
    def __new__(cls, *ags, **kwargs):
        if A.__instance == None:
            self = super(cls, A).__new__(cls)
            A.__instance = self
            return self
        else:
            return A.__instance
    def __init__(self, name):
        self.name = name
    @staticmethod
    def return_instance():
        return A.__instance
    def __del__(self):
        A.__instance = none
        del self
```

In [6]:

```
a = A("Pankaj")
print(id(a), a)
print(a.name)

b = A("Sachin")
print(id(a), a)
print(b.name)
print(a.name)
```

```
1866608508688 <__main__.A object at 0x000001B29A897F10>
Pankaj
1866608508688 <__main__.A object at 0x000001B29A897F10>
Sachin
Sachin
```

In [11]:

```
obj = A.return_instance()
print(obj)

print(obj.name)
```

```
<__main__.A object at 0x000001B29A897F10>
Sachin
```

In [12]:

```
del obj
print(a)
print(b)
```

```
<__main__.A object at 0x000001B29A897F10>
<__main__.A object at 0x000001B29A897F10>
```

In [13]:

```
obj
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_4976/219639173.py in <module>
----> 1 obj

NameError: name 'obj' is not defined
```

## 3. Objects are sharing same state

In [14]:

```
class A:
    __share_dict = dict()
    def __init__(self):
```

```
        self.data = A.__share_dict
    @classmethod
    def update_data(cls, key, value):
        cls.__share_dict[key] = value
    @classmethod
    def get_data(cls):
        return cls.__share_dict
```

In [15]:
```
a = A()
b = A()
```

In [32]:
```
a is b
```

Out[32]:
```
False
```

In [16]:
```
a.data['name'] = 'Pankaj Yadav'
print(a.data)
print(b.data)
```

```
{'name': 'Pankaj Yadav'}
{'name': 'Pankaj Yadav'}
```

In [17]:
```
b.update_data("age", 20)
print(a.data)
print(b.data)
```

```
{'name': 'Pankaj Yadav', 'age': 20}
{'name': 'Pankaj Yadav', 'age': 20}
```

## 4. raise Exception if more than one object created

In [18]:
```
class A:
    __instance = None
    def __init__(self, name):
        if A.__instance == None:
            self.name = name
            A.__instance = self
        else:
            raise Exception("Singleton Class Can not create a new object")
```

In [19]:
```
a = A("Pankaj")
```

In [20]:
```
b = A("Sachin")
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_4976/2317254624.py in <module>
----> 1 b = A("Sachin")

C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_4976/2351104706.py in __init__(self, name)
      6             A.__instance = self
      7         else:
----> 8             raise Exception("Singleton Class Can not create a new object")

Exception: Singleton Class Can not create a new object
```

In [ ]: