```
class A:
  def __init__(self):
    print('class A constructor')

class B(A):
  def __init__(self):
    print('class B constructor')

b = B()
```

```
class B constructor
```

```
class A:
  def __init__(self):
    print('class A constructor')

class B(A):
  def __init__(self):
    A.__init__(self)
    print('class B constructor')

b = B()
```

```
class A constructor
class B constructor
```

```
class A:
  def __init__(self):
    print('class A constructor')

class B:
  def __init__(self):
    A.__init__(self)
    print('class B constructor')

b = B()
```

```
class A constructor
class B constructor
```

```
class A:
  def __init__(self):
    print('class A constructor')

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

```
    A.__init__(self)
    print('class B constructor')

class C(A):
  def __init__(self):
    A.__init__(self)
    print('class C constructor')

class D(B,C):
  def __init__(self):
    B.__init__(self)
    C.__init__(self)
    print('class D constructor')

d = D()
```

```
class A constructor
class B constructor
class A constructor
class C constructor
class D constructor
```

```
class A:
  def __init__(self):
    print('class A constructor')

class B:
  def __init__(self):
    A.__init__(self)
    print('class B constructor')

class C:
  def __init__(self):
    A.__init__(self)
    print('class C constructor')

class D:
  def __init__(self):
    B.__init__(self)
    C.__init__(self)
    print('class D constructor')

d = D()
```

```
class A constructor
class B constructor
class A constructor
class C constructor
class D constructor
```

```
##############################################
class A:
  def __init__(self):
    print('class A constructor')

class B(A):
  def __init__(self):
    super().__init__()
    print('class B constructor')

class C(A):
  def __init__(self):
    super().__init__()
    print('class C constructor')

class D(B,C):
  def __init__(self):
    super().__init__()
    print('class D constructor')

d = D()
```

```
class A constructor
class C constructor
class B constructor
class D constructor
```

```
class sample:
  def f1(self):
    print('sample instance method f1( )')

class example(sample):
  def f1(self):
    print('example instance method f1( )')

e = example()
e.f1()
```

```
example instance method f1( )
```

```
class sample:
  def f1(self):
    print('sample instance method f1( )')

class example(sample):
  def f1(self):
    super().f1()
```

```
    print('example instance method f1( )')

e = example()
e.f1()
```

```
    sample instance method f1( )
    example instance method f1( )
```

```
################################################
class test:
  def __init__(self,f):
    self.f = f

  def __call__(self):
    print(' start '.center(30,'|'))
    self.f()
    print(' end '.center(30,'|'))

@test
def f1():
  print('f1 function defintion')
@test
def f2():
  print('f2 function definition')
@test
def f3():
  print('f3 function definition')

f1()
```

```
    |||||||||||| start ||||||||||||
    f1 function defintion
    |||||||||||| end ||||||||||||
```

```
f2()
```

```
    |||||||||||| start ||||||||||||
    f2 function definition
    |||||||||||| end ||||||||||||
```

```
f3()
```

```
    |||||||||||| start ||||||||||||
    f3 function definition
    |||||||||||| end ||||||||||||
```

```
################################################
class test:
```

```
    def __init__(self,f):
      self.f = f

    def __call__(self):
      print(' start '.center(30,'|'))
      self.f()
      print(' end '.center(30,'|'))

    def showdata(self):
      self.f()

  @test
  def f1():
    print('f1 function defintion')
  @test
  def f2():
    print('f2 function definition')
  @test
  def f3():
    print('f3 function definition')


  f1.showdata()
```

```
     f1 function defintion
```

```
  f2.showdata()
```

```
     f2 function definition
```

```
  f3.showdata()
```

```
     f3 function definition
```

```
  class chair:
    pass

  class room:
    chairs = []
    def __init__(self):
      print('Room object created')

    def addChair(self,obj):
      if len(self.chairs) < 10:
        self.chairs.append(obj)
      else:
        print('sorry, room is full')
```

```
  def removeChair(self):
    if len(self.chairs) > 0:
      self.chairs.pop()
    else:
      print('sorry, room is empty')

  def availableSpace(self):
    print(10 - len(self.chairs),'chairs')

  def occupiedSpace(self):
    print(len(self.chairs),'chairs')

  def clearRoom(self):
    self.chairs.clear()
    print('all chairs are removed')
```

```
r1 = room()
```

```
Room object created
```

```
r1.availableSpace()
```

```
10 chairs
```

```
r1.occupiedSpace()
```

```
0 chairs
```

```
r1.addChair(chair())
```

```
r1.availableSpace()
```

```
9 chairs
```

```
r1.occupiedSpace()
```

```
1 chairs
```

```
r1.addChair(chair())
```

```
r1.occupiedSpace()
```

```
2 chairs
```

```
r1.availableSpace()
```

```
8 chairs
```

```
r1.clearRoom()
```

```
all chairs are removed
```

```
r1.availableSpace()
```

```
10 chairs
```

✓ 0s completed at 1:32 PM