In [50]:

```
class Vegetable(object):
    veg_type=None
    def __init__(self):
        self.is_raw=True
        self.peels=True

class Carrot(Vegetable):
    veg_type="carrot"

class Potato(Vegetable):
    veg_type="potato"

class Leek(Vegetable):
    veg_type="leek"
```

```
In [44]: class PieceOfVeggie(Vegetable):
             def __init__(self, origin_veggie=None):
                 self.origin=origin_veggie
                 self.is_raw=origin_veggie.is_raw
                 self.peels=origin_veggie.peels
             @staticmethod
             def from_veggie(veggie):
                 return PieceOfVeggie(origin_veggie=veggie)


         class MashedVeggie(object):
             def __init__(self, veggie_list):
                 self.veggie_list=veggie_list
             @staticmethod
             def from_veggies(veggie_list):
                 return MashedVeggie(veggie_list=veggie_list)
             def __add__(self, other):
                 return MashedVeggie.from_veggies(veggie_list=self.veggie_list+other.vegg
         ie_list)

         class Soup(object):
             def __init__(self, mashed_veggie):
                 self.mashed_veggie = mashed_veggie
             @staticmethod
             def from_mashed_veggies(mash_veggie):
                 return Soup(mashed_veggie=mash_veggie)
```

In [61]:
```
class Boil(object):
    def __init__(self, time):
        self.time=time

    def __call__(self, veggies):
        for veggie in veggies:
            veggie.is_raw=False
        return veggies

class Peel(object):
    def __call__(self, veggie):
        veggie.peels=False
        return veggie

class Cut(object):
    def __init__(self, n):
        self.n=n
    def __call__(self, veggie):
        return [PieceOfVeggie.from_veggie(veggie)]*self.n


class Blend(object):
    def __call__(self, veggies):
        return MashedVeggie(veggies)
```

```
In [59]: c =Carrot()
         print(c.peels)
         Peel()(c)
         print(c.peels)
         print(c.is_raw)
         Boil()([c])
         print(c.is_raw)
         print(Cut(10)(c)[0].is_raw)
         Soup.from_mashed_veggies(Blend()(Cut(10)(c)) + Blend()(Cut(10)(c)))

         True
         False
         True
         False
         False

Out[59]: <__main__.Soup at 0x7f83a815dc50>
```

```
In [63]: potatos = [Potato()]*5
         carrots = [Carrot()]*5
         leeks = [Leek()]*3

         carrots_ready = list()
         for veggie in carrots:
             Peel()(veggie)
             carrots_ready.extend(Cut(10)(veggie))

         potatos_ready = list()
         for veggie in carrots:
             Peel()(veggie)
             potatos_ready.extend(Cut(10)(veggie))

         leeks_ready = list()
         for veggie in carrots:
             Peel()(veggie)
             leeks_ready.extend(Cut(10)(veggie))

         Boil(25)(carrots_ready)
         Boil(20)(potatos_ready)
         Boil(15)(leeks_ready)

         Soup.from_mashed_veggies(
             Blend()(carrots_ready)+Blend()(potatos_ready)+Blend()(leeks_ready)
         )

Out[63]: <__main__.Soup at 0x7f83a81565c0>
```

```
In [65]: # Import apache spark in python
         import pyspark
         # Main object to create RDD within Python
         sc = pyspark.SparkContext()
```

In [77]:

```
potatos = [Potato()]*5
carrots = [Carrot()]*5
leeks = [Leek()]*3
ingredients = sc.parallelize(potatos+carrots+leeks)
ingredients = ingredients\
                    .map(Peel())\
                    .flatMap(Cut(10))

def custom_boil(veggie_list):

    time_to_boil_table = {
        'carrot': 25,
        'potato': 20,
        'leek': 15
    }

    return Boil(
        time_to_boil_table[veggie_list[0].veg_type]
    )(veggie_list)



ingredients = ingredients\
                    .map(lambda veggie: (veggie.veg_type, [veggie]))\
                    .reduceByKey(lambda x,y:x+y)\
                    .map(lambda x: x[1])\
                    .map(custom_boil)\
                    .sum()
```