```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

#
# fichier: monome.py
#    date: 2011/05/02
#
# (tous les symboles non internationaux sont volontairement omis)
#

import string

from rationnel import *

class monome(object):
  """ classe pour un monome (rationnel et indeterminee) """

  def __init__(self, coefficient =rationnel(0), indeterminee ="", valide =True):
    """ constructeur """
    self.__valide = valide and coefficient.est_valide()
    s = ""
    if self.__valide:
      l = []
      for i in indeterminee:
        self.__valide = self.__valide and (i in string.letters)
        if self.__valide:
          l.append(i)
      if self.__valide:
        l.sort()
        for c in l:
          s += c
      else:
        self.__coefficient = rationnel()
    self.__coefficient = coefficient
    self.__indeterminee = s



  def __formater_indeterminee(self):
    """ formate l’indeterminee pour l’impression """
    t = str(self.__indeterminee)
    if len(t) > 1:
      v = []
      n = 0
      i = 0
      for ch in t:
        if i == 0:
          v.append(ch)
          n = 1
        else:
          if ch == v[-1]:
            n += 1
          else:
            if n > 1:
              v.append("^")
              v.append(str(n))
              n = 1
            v.append(" * ")
            v.append(ch)
        i = i + 1
      if n > 1:
        v.append("^")
        v.append(str(n))
      t = "".join(v)
    return t
  def __str__(self):
    """ representation en chaine de caracteres """
    if not self.__valide:
      return "(monome invalide)"
    t = str(self.__coefficient)
    if not self.__coefficient.est_positif():
      t = "(" + t + ")"
    if len(self.__indeterminee) == 0:
#     return "(" + str(self.__coefficient) + ")"
      return t
    else:
      s = self.__formater_indeterminee()
#     return "(" + str(self.__coefficient) + ") * " + s
      if self.__coefficient.est_unite():
        return s
      else:
        return t + " * " + s



  def est_valide(self):
    """ indique l’etat de validite """
    return self.__valide



  def valider(self):
    """ valider l’objet """
    self.__valide = True



  def invalider(self):
    """ invalider l’objet """
    self.__valide = False



  def __cmp__(self, autre):
    """ comparaison (ordre sur les indeterminees) """
    u = len(self.__indeterminee)
    v = len(autre.__indeterminee)
    if u != v:
      return -cmp(u, v)
    else:
      return cmp(self.__indeterminee, autre.__indeterminee)



  def get_coefficient(self):
    """ accesseur coefficient """
    return self.__coefficient



  def get_indeterminee(self):
    """ accesseur indeterminee """
    return self.__indeterminee



  def __neg__(self):
    """ monome oppose """
    return monome(-(self.__coefficient), self.__indeterminee, self.__valide)



  def produit(self, autre):
    """ produit de deux monomes """
    if self.__valide and autre.__valide:
      c = self.__coefficient * autre.__coefficient
      s = str(self.__indeterminee) + str(autre.__indeterminee)
      return monome(c, s)
    else:
      return monome(rationnel(0, 1, False), "")



if __name__ == "__main__":

  x = monome(rationnel(8, -14),"$")
  print x
  print x.est_valide()

  y = monome(rationnel(8, 0),"a")
  print y
  print y.est_valide()

  z = -x
  print z
  print z.est_valide()

  x = monome(rationnel(8, -14),"aaaxxaacccddxxddx")
  print x
  print x.est_valide()
```