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

#
# fichier: entier.py
#    date: 2011/05/07
#
# (tous les symboles non internationaux sont volontairement omis)
#

#def pgcd_naturel(a, b):
# """ pgcd de deux nombres entiers naturels """
# if b == 0:
#   return a
# else:
#   return pgcd_naturel(b, a % b)



def pgcd_naturel(a, b):
  """ pgcd (iteratif) de deux nombres entiers naturels """
  if b == 0:
    return a
  r = long(a) % long(b)
  while r > 0:
    a = b
    b = r
    r = long(a) % long(b)
  return b



def pgcd_liste(l):
  """ pgcd d'une liste d'entiers """
  if l == []:
    return 1
  if len(l) == 1:
    return abs(l[0])
  g = pgcd_naturel(abs(l[0]), abs(l[1]))
  if len(l) == 2:
    return g
  for n in l[1:]:
    g = pgcd_naturel(g, abs(n))
  return g



def ppcm_naturel(a, b):
  """ pgcd de deux nombres entiers naturels """
  if a == 0 or b == 0:
    return 0
  else:
    return (a*b) / pgcd_naturel(a, b)



class entier(object):
  """ classe pour un nombre entier """

  def __init__(self, valeur =long(0), valide =True):
    """ constructeur """
    self.__valide = valide
    if self.__valide:
      self.__valeur = long(valeur)
    else:
      self.__valeur = long(0)
```

```
  def __str__(self):
    """ representation en chaine de caracteres """
    if self.__valide:
      return str(self.__valeur)
    else:
      return "(nombre entier invalide)"



  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
    self.__valeur = long(0)



  def valeur(self):
    """ donne la valeur (type natif long) de l'entier """
    return long(self.__valeur)



  def __add__(self, autre):
    """ addition """
    if self.__valide and autre.__valide:
      a = long(self.__valeur)
      b = long(autre.__valeur)
      return entier(a + b)
    else:
      return entier(0, False)



  def __neg__(self):
    """ nombre oppose """
    if self.__valide:
      a = long(self.__valeur)
      return entier(-a)
    else:
      return entier(0, False)



  def __sub__(self, autre):
    """ soustraction (par addition de l'oppose) """
    return self.__add__(autre.__neg__())



  def __mul__(self, autre):
    """ multiplication """
    if self.__valide and autre.__valide:
      a = long(self.__valeur)
      b = long(autre.__valeur)
      return entier(a * b)
```

```
    else:
      return entier(0, False)



  def __div__(self, autre):
    """ division (si le quotient est entier) """
    if not(self.__valide and autre.__valide):
      return entier(0, False)
    if autre.__valeur == 0:
      return entier(0, False)
    a = long(self.__valeur)
    b = long(autre.__valeur)
    if (a % b) != 0:
      return entier(0, False)
    return entier(a / b)



  def __pow__(self, autre):
    """ exponentiation (si exposant entier naturel) """
    if not(self.__valide and autre.__valide):
      return entier(0, False)
    a = self.__valeur
    n = autre.__valeur
    if a == 0:
      if n <= 0:
        return entier(0, False)
      else:
        return entier()
    if n < 0:
      return entier(0, False)
    p = 1
    while n > 0:
      if n % 2 == 1:
        p *= a
      n //= 2
      a *= a
    return entier(p)



  def est_nul(self):
    """ l'entier est-il nul ? """
    if self.__valide:
      return self.__valeur == 0
    else:
      return False



  def est_positif(self):
    """ l'entier est-il positif ? """
    if self.__valide:
      return self.__valeur > 0
    else:
      return False



def pgcd_entier(a, b):
  """ pgcd pour le type entier """
  if isinstance(a, entier) and isinstance(b, entier):
    if a.est_valide() and b.est_valide():
      return entier(pgcd_naturel(abs(a.valeur()), abs(b.valeur())))
  return entier(0, False)
```

```
def ppcm_entier(a, b):
  """ ppcm (positif) pour le type entier """
  if isinstance(a, entier) and isinstance(b, entier):
    if a.est_valide() and b.est_valide():
      return entier(ppcm_naturel(abs(a.valeur()), abs(b.valeur())))
  return entier(0, False)



if __name__ == "__main__":

  a = entier(-5)
  b = entier(-20, False)
  x = pgcd_entier(a, b)

  print x.est_valide()
  print x
  print x.valeur()

  a = entier(-5)
  b = entier(-20)
  x = pgcd_entier(a, b)

  print x.est_valide()
  print x
  print x.valeur()

  x = ppcm_entier(a, b)

  print x.est_valide()
  print x
  print x.valeur()
```