```
###############################################################################
#                                                                             #
#   L11.py    Text Games II: Object-oriented Programming                      #
#                                                                             #
#   Course:      CS1021C Computer Science I (Dr. Talaga)                      #
#   Author(s):   Kevin Ernst      (kme)                                       #
#                Allyssa Griffith                                             #
#                Nicholas Leigh   (NTL)                                       #
#                Kyle Rone                                                    #
#                                                                             #
#   Date:        03 April 2013                                                #
#                                                                             #
###############################################################################

###############################################################################
#                          R E F E R E N C E S                                #
#                                                                             #
#  1. "Taipan!" - http://en.m.wikipedia.org/wiki/Taipan!                      #
#                                                                             #
#  2. http://www.taipangame.com/                                              #
#                                                                             #
#  3. 'commafy' function from Peterbe.com:                                    #
#     http://www.peterbe.com/plog/thousands-commafy-large-numbers             #
#                                                                             #
###############################################################################
import sys #import (executable, version_info)
import os #import (system, name)
from random import randrange
from string import ljust, join
from time import sleep
DEBUGGING = True
if DEBUGGING:
  RUTHLESS_ADVERSARIES, DEFLATION, EARLY_RETIREMENT = 1,1,1
  SQUID_SEASON, IMPATIENT = 1,1
else:
  RUTHLESS_ADVERSARIES, DEFLATION, EARLY_RETIREMENT = 0,0,0
  SQUID_SEASON, IMPATIENT = 0,0

###############################################################################
#                      G L O B A L   C O N S T A N T S                        #
###############################################################################
# how many times you can sail around the "world" before it's time to retire
RETIREMENT_AGE         = 10 - 9*EARLY_RETIREMENT
# you're a millionaire. retire to a secluded tropical isle
RICH_ENOUGH_TO_RETIRE  = 1e6 - int(DEFLATION*0.99*1e6)
# how much cash you get for defeating a pirate fleet (updated in seaBattle)
PIRATE_BOOTY           = 200
# Starting repair cost for 100% damage
REPAIR_COST            = 1000
STARTING_GUNS          = 5
SHIP_DAMAGE_SEA_VOYAGE = 3  + 5*RUTHLESS_ADVERSARIES
SHIP_DAMAGE_PIRATES    = 10 + 20*RUTHLESS_ADVERSARIES
SHIP_DAMAGE_SQUID      = 25 + 25*RUTHLESS_ADVERSARIES
PAUSE_MSG              = 2 - IMPATIENT
PAUSE_MSG_SHORT        = 1 - 0.5*IMPATIENT
PAUSE_EVENT            = 5 - 3*IMPATIENT
DEFAULT_FIRM_NAME      = "Permanent Assurance"
DEFAULT_SHIP_NAME      = "The Crimson Permanent Assurance"
DEFAULT_SHANTY         = """
It's fun to charter an accountant,
And sail the wide accountan-cy.
To find, explore the funds offshore,
And skirt the shoals of bankruptcy.
It can be manly in insurance.
We'll up your premium semi-annually.
It's all tax-deductible,
We're fairly incorruptible.
We're sailing on the wide accountan-cy."""

# I decided on a 1-based "array" because the original Tapian! game assigned
# these numbers to the ports, so I wouldn't get confused.
port_names = { 1:'Hong Kong', 2:'Shanghai',  3:'Nagasaki', 4:'Saigon',
               5:'Manila',    6:'Singapore', 7:'Batavia',
```

```
                0:'the open ocean' }
port_descriptions = {
    1:"Hong Kong: All the familiar sights, sounds, and smells of home!",
    2:"Shanghai bustles with traders from east and west.\n" +
      "Be on the lookout for pickpockets!",
    3:"Taipan, I've heard one can obtain Portuguese tobacco here.\n" +
      "And sponge cakes!",
    4:"We've been at sea for many months, Tapian. Let's treat the men\n" +
      "to some nice hot noodles in Saigon.",
    5:"Tapain, the Peruvian silver in Manila will be quite valuable\n" +
      "for trade on the mainland.",
    6:"If we'll be needing any supplies for repair of the ship,\n" +
      "Singapore will be the place to aquire them, sir.",
    7:"You'll be wanting to stay on board in Batavia, sir. Much\n" +
      "bad blood against Westerners here.",
    0:"'I'm on the sea! I'm on the sea!\n" +
      " I am where I would ever be,\n" +
      " With the blue above and the blue below,\n" +
      " And silence wheresoe'er I go.'\n\n" +
      "Nothing quite like it, eh, sir?" }
# Where can you sail from each port (in order of North, South, East, West):
port_routes = { 1:{ 'n':2, 's':5 },
                2:{ 's':1, 'e':3 },
                3:{ 'e':2 },
                4:{ 'n':1, 's':6, 'e':5 },
                5:{ 'n':1, 'w':4 },
                6:{ 'n':4, 's':7 },
                7:{ 'n':4, 'e':6 } }

characters = ['Mc Henry', 'Li Yuen', 'Elder Brother Wu']
directions = { 'n':'north', 's':'south', 'e':'east', 'w':'west' }
conditions = [ 'Poor', 'Fair', 'Good', 'Very Good' ]
help_msg_1 = """
TAIPAN!
=======
Insipred by the Mega Micro Computers game for the TRS-80 and Apple ][.

You are the head of an up-and-coming trading company based in Hong Kong
in the 19th century and the captain of your own ship.

You begin the game in your home port, and can sail to various ports in
and around the South China Sea. Other ports you may visit are Shanghai,
Nagasaki, Saigon (present day Ho Chi Min City in Vietnam), Manila,
Singapore, and Batavia (the old Dutch name for Jakarta, the capital
city of Indonesia).




NOTE: When gameplay begins, be sure to make the command area in JES at least
large enough to see this text and the TAIPAN! title at the top.
"""
# FIXME: Can't travel by cardinal directions.
# you can reference by cardinal direction (North, South, East, or West).
help_msg_2 = """
Traveling close to the mainland or to nearby ports incurs less risk
(of pirate or giant squid attacks) than traveling across open ocean.
Upon arriving at a port, you are presented with a list of neighboring
ports, which you can safely travel to under the protection of
maritime law enforcement. You can travel to these nearby ports by
pressing the letter corresponding to the one of the cardinal
directions: 'n', 's', 'e', or 'w'.

You can travel to any other port (potentially crossing open ocean) by
pressing one of the number keys assigned to the desired port.

If your ship incurs damage during your travels on the high seas, you
may request repairs in your home port of Hong Kong. If your ship is
heavily damaged, it's much to your advantage to stick close to the
coast on the journey home.
```

```
The game ends when your ship is sunk by marauders, you make enough
money to retire a millionaire, or you sail around the world a
specified number of times (which you can check by pressing 'r').
"""
# FIXME: ---or--- you sail to each port ten times

# Fix up a few things if we're not running in JES/Jython:
# -------------------------------------------------------
def runningInJES():
  """Determine whether we're running in JES (sys.platform returns the Java JRE
  version instead of the machine architecture or OS type) and return True if
  we are, False otherwise (kme)"""
  # import sys
  # This seems like an odd combination, and it's probably unique to JES:
  return sys.platform.startswith("java") and sys.executable == None

if runningInJES():
  def cls():
    printNow("\n"*25)
else:
  # We're running in IPython or 'python' in the console
  def cls():
    os.system(['clear','cls'][os.name == 'nt'])

  def printNow(s):
    """Replacement for JES's 'printNow' function when running in CPython
    (kme)"""
    print(s)

  def requestString(message="Enter a value -->"):
    """Replacement for JES's 'requestString' function when running in
    CPython (kme)"""
    #no_input = True
    #while no_input:
    print(message), # no newline
    response = str(raw_input())
    #  if len(response) == 0:
    #    print("That is not an acceptable response. Try again.")
    #  else:
    #    no_input = False
    return response

  def requestInteger(message="Enter a number -->"):
    """Replacement for JES's 'requestInteger' function when running in
    CPython (kme)"""
    bad_input = True
    while bad_input:
      print(message + " "), # no newline
      response = raw_input()
      if len(response) == 0 or not response.isdigit():
        print("That is not an acceptable response. Try again.")
      else:
        bad_input = False
    return int(response)

##############################################################################
#                    U T I L I T Y   F U N C T I O N S                       #
##############################################################################
def commafy(n):
  """Stringify a large number, with commas in the thousands' places (kme,
  based on http://www.peterbe.com/plog/thousands-commafy-large-numbers"""
  rev_n = enumerate(str(n)[::-1])  # a reversed string, enumerated
  r = []
  for i, c in rev_n:
      if i and (not (i % 3)):
          r.insert(0, ',')
      r.insert(0, c)
  return ''.join(r)

##############################################################################
#                          E X C E P T I O N S                               #
##############################################################################
```

```
class CantSailThere(Exception):
  def __init__(self):
    self.message = "You can't sail in that direction."
  def __str__(self):
    return self.message

class ShipSunk(Exception):
  def __init__(self, ship_name="The ship"):
    self.message = "Taipan, we've sustained too much damage. " + \
        "%s is going down!" % ship_name
  def __str__(self):
    return self.message

class EndGame(Exception):
  def __init__(self, ship_name="The ship"):
    self.message = "The game has ended."
  def __str__(self):
    return self.message

class BattleDefeat(Exception):
  def __init__(self):
    self.message = "We were bested by the pirate fleet, Taipan."
  def __str__(self):
    return self.message

class BattleVictory(Exception):
  def __init__(self):
    self.message = "We were victorious against the pirate fleet. Huzzah!"
  def __str__(self):
    return self.message

class ThatWouldBankruptYou(Exception):
  def __init__(self):
    self.message = "You don't have that much cash, Taipan!"
  def __str__(self):
    return self.message


##############################################################################
#                          C L A S S   O B J E C T S                         #
##############################################################################
class Ship:
  """The ship object for Taipain! (kme, with modifications by everyone)"""
  def __init__(self, name=DEFAULT_SHIP_NAME, starting_choice=2,
               shanty=DEFAULT_SHANTY):
    """Initialization for the 'Ship' class"""
    self.condition  = 100     # ship's condition (0-100%)
    self.sea_shanty = shanty  # the ship's official sea shanty

    # initialCash=0, initialDebt=0, initialGuns=5
    # These are the choices from the original Taipan! game:
    if starting_choice == 1: # cash and debt
      self.guns = 0
      self.cash = 400
      self.debt = -5000
    elif starting_choice == 2: # guns and no cash (but no debt)
      if DEBUGGING:
        self.guns = STARTING_GUNS - 2
      else:
        self.guns = STARTING_GUNS
      self.cash = 0
      self.debt = 0
    # The ship starts with 60 holds. A gun takes 10 holds.
    self.holds = 60 - 10*self.guns

  def getName(self):
    """Returns the ships name - NTL"""
    return self.name

  def getCondition(self):
    """Returns the ships condition - NTL"""
    return self.condition
```

```
  def getGuns(self):
    """Returns the number of guns on the ship - NTL"""
    return self.guns

  def getCash(self):
    """Returns the amount of cash on the ship - NTL"""
    return self.cash

  def getDebt(self):
    """Returns the amount of debt you owe - NTL"""
    return self.debt

  def setName(self,name):
    """Sets your ships name, uses a default name if none given - NTL"""
    if name: # Not None or empty string
      self.name     = name    # give this ship a name
    else:
      self.name     = DEFAULT_SHIP_NAME

  def setGuns(self, amt):
    """Sets the amount of guns on the ship - NTL"""
    if amt < 0:
      raise ValueError
    else:
      self.guns = amt

  def addGuns(self, how_many=1):
    """Add a specified number of guns to the ship's armaments (kme)"""
    if how_many < 0:
      raise ValueError
    else:
      self.guns += how_many

  def destroyGun(self):
    """Destroy a gun in battle. Returns the number of guns remaining in the
    arsenal. (kme)"""
    if self.guns >= 1:
      self.guns -= 1
    return self.guns

  def setCash(self, amt):
    """Sets the amount of cash on the ship - NTL"""
    self.cash = amt

  def setDebt(self, amt):
    """Sets the amount of debt you owe - NTL"""
    self.debt = amt

  def setCondition(self, amt):
    """Sets the ships condition - NTL"""
    self.condition = amt

  def printStatus(self):
    """Print a summary of the ship's status: guns, cash, debt, and the
    condition of the ship."""
    pad = len(str(RICH_ENOUGH_TO_RETIRE))
    s = "The status of " + self.name + " is as follows:"
    printNow(s + '\n' + '-'*len(s))
    printNow("CASH:   " + ljust(commafy(str(self.getCash())), pad) +
             "   DEBT:   -" + ljust(commafy(str(self.getDebt())), pad))
    c = self.getCondition()
    if c == 100:
      s = "   REPAIR:   Perfect! (%i%%)" % c
    elif c > 100:
      s = "   REPAIR:   Magically protected! (%i%%)" % c
    else:
      t = conditions[(c/25)%4]
      s = "   REPAIR:   %s (%i%%)" %(t, c)
    printNow("GUNS:   " + ljust(str(self.guns), pad) + s)
    printNow("")

  def causeDamage(self, damage):
    """Cause the ship to sustain battle damage, subtracting 'damage' from
```

```
    self.condition. Raise 'ShipSunk' if this causes condition to go less than
    zero. (kme, modifications by NTL)"""
    self.setCondition(self.getCondition() - damage)
    if self.getCondition() <=0:
      raise ShipSunk

  def doShipRepairs(self):
    """This method gets invoked when the ship's condition is <90%, and
    allows you to pay McHenry from the Hong Kong shipyard (kme, with
    modifications by NTL)"""
    # Should test for location here, but it's no longer local to this class.
    # :/
    if not DEBUGGING:
      assert(self.getCondition() < 100)  # it's a programming error if we get
                                         # here otherwise
    damage = 100 - self.getCondition()
    printNow("Taipan, Mc Henry from the Hong Kong Shipyards has arrived.")
    sleep(PAUSE_MSG_SHORT)
    printNow('')
    printNow("He says, 'I see ye've a wee bit of damage to yer ship.'")
    resp = requestString("Will ye be wanting repairs? [Y/n]")
    if resp == '' or resp[0].lower() == 'y':  # yes or ENTER
      printNow("\nOch, 'tis a pity to be %i%% damaged." % damage)
      # Compute cost for repairs based on how much cash you've got. If you're
      # richer, McHenry's going to charge you more. Like any good capitalist,
      # he's capturing consumer surplus.
      # ref: http://www.joelonsoftware.com/articles/CamelsandRubberDuckies.html
      price = int((REPAIR_COST + min(10000, 0.15*self.getCash())) *
                  damage/100.0)
      printNow("We can fix yer whole ship for " + str(price) +
               ", or make partial repairs if you wish.")
      good_response = False
      while not good_response:
        resp = requestInteger("How much will ye spend? (0-%i)" % price)
        resp = int(resp) # always ignore fractional currency
        if not resp:  # None or 0
          good_response = True
          continue
        elif resp < 0:
          printNow("That isn't a figure I understand, sir.\n")
          continue
        elif resp > self.getCash():
          printNow("That'll bankrupt us, Taipan!\n")
          continue
        else:
          good_response = True
          self.setCash(self.getCash() - resp)
          # McHenry will gladly let you pay more than the agreed-upon price,
          # but won't fix your ship any more than all the way fixed.
          if resp > price:
            self.setCondition(100)
          else:
            self.setCondition(self.getCondition() +
              int((resp / float(price)) * damage))
      if self.guns < STARTING_GUNS:
        busted_guns = STARTING_GUNS - self.guns
        sleep(PAUSE_MSG_SHORT)
        # ---------------------------------------------------------------
        #               Offer to replace damaged guns:
        # ---------------------------------------------------------------
        printNow("\nIt seems %i of yer guns have been damaged " %busted_guns +
                 "in battle.\nYour ship has holds for %i guns.\n" %
                 STARTING_GUNS)
        resp = requestString("Will ye be wanting to purchase " +
                             "replacement guns? [Y/n]")
        if resp == '' or resp[0].lower() == 'y':  # yes or ENTER
          # a tenth the base price of repairing your whole ship plus 1% of
          # your cash on hand up to a max of 1000:
          price = int(0.10*REPAIR_COST + min(1000, 0.01*self.cash))
          printNow("Ay, very well then.\nI can replace yer %i busted " %
                   busted_guns + "guns at a price of %i apiece." % price)
          good_response = False
          while not good_response:
```

```
            resp = requestInteger("How many guns will ye replace? (0-%i)"
              % busted_guns)
            resp = int(resp) # always ignore fractional quantity
            if not resp:  # None or 0
              good_response = True
              continue
            elif resp < 0:
              printNow("That isn't a figure I understand, sir.\n")
              continue
            elif resp > self.cash:
              printNow("That'll bankrupt us, Taipan!\n")
              continue
            else:
              good_response = True
              self.setCash(self.cash - price*resp)
              self.addGuns(resp)
    else:
      # Ye won't be wanting repairs, then.
      return


class Port:
  """A class that contains all the ports of call for Taipan! along with
  methods to determine whether one can sail between two ports directly, and to
  print the "arriving at..." message for the current port. (kme, with
  modifications by everyone)"""
  # Don't put this here. It becomes a class variable which is shared (and
  # modified) by all instances:
  #port_to_the = {}  # keep track of which port lies to n,s,e,w

  def __init__(self, portnum):
    self.port_to_the = {}  # keep track of which port lies to n,s,e,w
    self.name = port_names[portnum]
    self.description = port_descriptions[portnum]
    self.port_number = portnum
    # Populate the "port_to_the[direction]" list:
    #for direction in ['n', 's', 'e', 'w']:
    #  try:
    #    self.port_to_the[direction] = port_routes[portnum][direction]
    #  except KeyError:
    #    self.port_to_the[direction] = 0 # open ocean

  #def neighbors(self):
  #  """Return a list of the neighboring ports in NSEW order"""
  #  #                   actually, no it doesn't  ^^^^
  #  # TODO: make this a generator, so it actually returns in the proper
  #  # order.
  #  t = []
  #  for d in directions.keys():
  #    t.append(self.port_to_the[d].name)
  #  return t

  # def neighboringPortNumbers(self):
  #   """TODO: Prints a list of neighboring port numbers (for use in the
  #   'canSailTo' method"""

  def getPortNumber(self):
    """Returns the port number - NTL"""
    return self.port_number

  def getName(self):
    """Returns the name of the port at [portnum] - NTL"""
    portnum = self.getPortNumber()
    return port_names[portnum]

  def getPortDescription(self):
    """Returns the description of the port at [portnum] - NTL"""
    portnum = self.getPortNumber()
    return port_descriptions[portnum]

  def getPortToThe(self, direction):
    """Returns the port (object) to the [direction] - NTL"""
    # For some odd reason, getPortToThe('whatever').getPortNumber() doesn't
```

```
    # work. So getPortNumberToThe('whatever') was created to work around this.
    return self.port_to_the[direction]

  def getPortNumberToThe(self, direction):
    """Returns the /number/ of the port in the requested direction (kme)"""
    return self.port_to_the[direction].getPortNumber()

  def setPortToThe(self, direction, portref):
    """Set the value at 'direction' in the port_to_the dictionary to the Port
    reference given as the second argument (NTL)"""
    # This function allows the Game class to set the values in the
    # 'port_to_the' dict to references to the other Port class instances.
    # Previously, 'port_to_the' just held integers representing the key in the
    # 'ports' dictionary of the Game class.
    assert(direction in directions.keys())
    self.port_to_the[direction] = portref

  def arrivalMessage(self):
    """Prints the arrival message (and a brief ship's status report) for this
    port (kme)"""
    #self.__printDescription()
    s = "Arriving in the port of " + self.getName() + "...."
    printNow('='*len(s) + '\n' + s + '\n' + '='*len(s))
    printNow(self.getPortDescription() + '\n')
    self.printNeighboringPorts()
    printNow("")

  def canSailTo(self, portnum):
    """Returns False if you can't sail *directly* to that port without putting
    out to sea (kme)"""
    #if portnum in self.port_to_the.values(): # are Port objs now, won't work
    plist = []
    for p in self.port_to_the.values():
      plist.append(p.port_number)
    if portnum in plist:
      return True
    else:
      #raise CantSailThere
      return False

  def printDescription(self):
    """Prints a description of this port. (kme)"""
    printNow("You are in the port of " + self.getName())

  def printNeighboringPorts(self):
    """Prints a list of all neighboring ports (to which you can sail directly
    without having to put to sea. (kme, NTL)"""
    # @NTL: since this is a private method, you can directly access the
    # class's internal variables here, and in a "real" class, there may be
    # speed benefits to doing so. I'm leaving this one as is. --kme
    printNow("To the [N]orth lies " + self.getPortToThe('n').getName() + ".")
    printNow("To the [S]outh lies " + self.getPortToThe('s').getName() + ".")
    printNow("To the [E]ast  lies " + self.getPortToThe('e').getName() + ".")
    printNow("To the [W]est  lies " + self.getPortToThe('w').getName() + ".")

  #def listNeighboringPorts(self):
  #  """Return a Python dictionary of neighboring ports (essentially an
  #  encapsulated copy of port_routes) which can be processed in a list context
  #  to determine whether a cardinal direction ('n', 's', 'e', 'w') given at
  #  the sail prompt is a valid one (kme)"""
  #  # This may duplicate some of the functionality of neighbors() and
  #  # neighboringPortNumbers(), which are unimplemented as of 2013-04-05, but
  #  # marked as 'TODO' above.
  #  for d in self.port_to_the: #.keys() is implicit
  #    if d

class HomePort(Port):
  """Derived Port class that only applies to Hong Kong, where you can get your
  ship repaired, visit the warehouse, and borrow money from Elder Brother
  Wu. Allyssa Griffith."""

  def doBusinessWithBrotherWu(self,ship):
    resp = requestString('Do you have business with Elder Brother Wu, '+
```

```
      'the moneylender? [y/n]')
    if not resp == None and resp[0].lower() == 'y':
      good_response = False
      while not good_response:
        resp = requestInteger('How much would you like to borrow?')
        resp = int(resp) # always ignore fractional quantity
        if not resp:  # None or 0
          good_response = True
          continue
        if ship.getCash() == 0:
          if resp > ship.getGuns() * 100:
            printNow("He won't lend you so much, Taipan!")
            continue
          else:
            good_response = True
            ship.setCash(ship.getCash() + resp)
            ship.setDebt(ship.getDebt() + resp)
        elif resp > ship.getCash():
          printNow("He won't lend you so much, Taipan!")
          continue
        elif resp < 0:
          printNow("That isn't a figure I understand, sir.\n")
          continue
        else:
          good_response = True
          ship.setCash(ship.getCash() + resp)
          ship.setDebt(ship.getDebt() + resp)
    else:
      return

 # def wareHouse(self):
    #warehouse = requestString('Would you like to transfer cargo? yes/no')
    #if warehouse == 'yes':
      #transfer = requestString('What would you like to transfer? o/g/s/a')
      # items: Opium, General, Silk, Arms

 # def retirement(self):
   # if bank + g.ship.cash >= RICH_ENOUGH_TO_RETIRE:
     # r = requestString('You have enough cash to retire. Is this what you'+
       # 'would like to do? yes/no')
      #if r == 'yes':
        #retire


class Game:
  """The map structure of the game. Basically a list of Ports, and the actions
  to act upon them. Always passed in a reference to 'ship' so it can
  manipulate data structures inside it as the ship sails from port to port.
  (kme, with modifications by everyone)"""
  def __init__(self, firm_name=DEFAULT_FIRM_NAME):
    self.ship = Ship()
    if firm_name:  # not None or empty string
      self.firm_name = firm_name
    else:
      self.firm_name = DEFAULT_FIRM_NAME
    self.current_port = None
    self.en_route_to = None
    self.ports = {}
    self.visit_count = {}
    # Initialize the list of ports as a member. The ports already understand
    # their relative position to each other, and set up this mapping in their
    # own __init__ methods.
    for p in port_names.keys():
      if p == 1:
        self.ports[1] = HomePort(1)
      else:
        self.ports[p] = Port(p)
      #if p != 0:
      # I've decided to keep track of visits to port 0 (the open ocean).
      # Could work this into some kind of achievement system, or ranking
      # upon retirement: "Master of the High Seas"
      self.visit_count[p] = 0  # initialize all visit counts to zero.
```

```
    # Now initialize each ports 'port_to_the' dictionary with references to
    # the newly created ports:
    for i in self.ports: #.keys() is apparently implicit
      for d in ['n', 's', 'e', 'w']:
        try:
          # port_routes[i][d] is the port in the direction ('d') of the
          # current port (key 'i' in self.ports):
          self.ports[i].setPortToThe(d, self.ports[port_routes[i][d]])
        except KeyError:
          # if the key 'd' doesn't exist in the port_routes dictionary, then
          # set this direction's destination to 0 = open ocean
          self.ports[i].setPortToThe(d, self.ports[0])

    # start in Hong Kong
    self.current_port = self.ports[1]

  def getName(self):
    """Returns the name of the firm - NTL"""
    return self.firm_name

  def incrementVisitsForPortNumber(self, portnum):
    """Increment the number of visits for the given port number, 'portnum'
    (kme)"""
    if portnum < 0:
      raise ValueError
    else:
      self.visit_count[portnum] += 1

  def getPortVisits(self):
    """Get a list, in numerical order from 0 to (default) 7 of the visit
    counts for all the ports. (kme)"""
    # NB: The sorting probably isn't necessary, since the main game loop is
    # just going to do a min() on this list when deciding whether or not you
    # can retire in Hong Kong.
    visitlist = []
    portnums = self.visit_count.keys()
    portnums.sort() # just in case they're not already (not guaranteed with
                    # the keys of a Python dictionary, I think)
    for i in portnums:
      visitlist.append(self.visit_count[i])
    return visitlist

  def timesAroundTheWorld(self):
    """Return the number of times the player has "sailed around the world" by
    looking a the minimum number of port visits from visit_count (kme)"""
    return min(self.visit_count.values())

  def printPortMenu(self):
    """Print a menu of ports to which you can sail (kme, modifications by NTL)"""
    s = ''
    for i in range(1, len(self.ports)):  # skips '0'!
      s = s + "[%i] %s     " %(i, self.ports[i].getName())
      if i%4 == 0: s = s + '\n'
    printNow(s)

  def sailTo(self, to_port):
    """Set sail from the current port to the given destination port 'to_port'.
    If the destination port is a neighbor, updates self.current_port and
    returns. Otherwise, calls 'putToSea'. (kme, with modificatins by NTL)"""
    to_port = int(to_port) # for my sanity

    # Validate the input first:
    portname = self.ports[to_port].getName()
    if self.current_port.canSailTo(to_port):
      # We can sail to the destination port along the coast without crossing
      # the open ocean.
      printNow("Aye, sir. We'll set out for " + portname + " straight away!")
      sleep(PAUSE_MSG)
      cls()
      #return self.ports[to_port]
      self.current_port = self.ports[to_port]
    else:
      # Not a neighboring port. Must deploy to open ocean.
```

```
      # TODO: Require purchasing supplies first.
      printNow("The journey to " + portname +
               " will require crossing the open ocean, Taipan.")
      printNow("May the sea goddess look favorably on our journey!")
      sleep(PAUSE_EVENT)
      cls()
      #return self.putToSea(to_port)
      self.putToSea(to_port)

  def randomShips(self):
    """Written by Kyle Rone 4/3/2012"""
    # The 'Game' object (which this function is a member of) doesn't have
    # a 'cash' property. But the 'ship' object /inside/ it does. Also, updated
    # to use NTL's getter for the "private" cash variable.
    #
    # Also, in order to even have access to self.ship, you need to pass in
    # 'self' as the first argument to a class member function. ftfy.
    #cash = self.cash()
    cash = self.ship.getCash()
    if cash <= 500:
      # Changed this to 2,5 so I wouldn't have to fix the "1 ships on the
      # horizon!" status message (kme)
      numberShips = randrange(2,5)
    elif (cash > 500) and (cash <= 5000):
      numberShips = randrange(5,10)
    elif (cash > 5000) and (cash <= 50000):
      numberShips = randrange(10,20)
    elif (cash > 50000) and (cash <= 100000):
      numberShips = randrange(20,35)
    elif (cash > 100000) and (cash <= 500000):
      numberShips = randrange(35,50)
    elif (cash > 500000):
      numberShips = randrange(50,100)
    return numberShips

  def seaBattle(self):
    """Initiate a sea battle with a specified number of adversaries
    (defaulting to a random number generated by randomShips(), based on how
    much cash you're carrying). (kme, modified by Kyle Rone)"""
    # (kme) There's a problem with using a function to set default values in
    # the argument list, and JES gripes about it. I'm going to remove the
    # optional 'numberShips' argument and just set it here first thing with
    # randomShips():
    numberShips = self.randomShips()
    printNow("Taipan, there are %i pirate ships on the horizon!\n" %  numberShips)
    sleep(PAUSE_MSG)
    # Run or fight?
    if self.ship.getCondition() > 50:
      printNow("We're seaworthy, there's a good chance we can make it " +
               "if we flee now!")
      resp = requestString("Taipan, should we run? [Y/n]")
      if resp == '' or resp[0].lower() == 'y':  # yes or ENTER
        printNow("\nAye, sir. We'll try to get away...")
        sleep(PAUSE_EVENT)
        printNow("\nWe made it!")
        sleep(PAUSE_MSG)
        raise BattleVictory
      else:
        printNow("\nAye, sir. We'll fight.")
    else:
      # If the ship's not seaworthy enough to escape the battle...
      printNow("Our ship's in poor condition, sir, she might not make it!")
      printNow("We're not seaworthy enough to run... we'll have to fight!")

    # At the moment, every encounter with pirates yields 10% battle damage,
    # the defeat of the pirate fleet, and $500.
    sleep(PAUSE_EVENT)
    try:
      self.ship.causeDamage(SHIP_DAMAGE_PIRATES)
    except ShipSunk:
      printNow("\nWe can't hold 'em off, Tapian! We're being boarded!")
      raise
    else:
```

```
      # Small modification here to use the PIRATE_BOOTY global constant again
      # (I set this to 200 to start with). This guarantees you'll get /at
      # least/ 200 out of defeating pirates.A
      #
      # TODO: Need to make sure that we're making enough money to pay McHenry
      # for the damage incurred by getting the money in the first place!
      booty = PIRATE_BOOTY + (numberShips * randrange(PIRATE_BOOTY))
      printNow("\nWe've taken heavy damage, but the pirates are retreating!")
      printNow("Look at the buggers run, Taipan! Huzzah!\n")
      sleep(PAUSE_MSG)
      printNow("Taipan, we've recovered %i in booty from a captured pirate ship!"
               % booty)
      self.ship.setCash(booty)
      sleep(PAUSE_EVENT)
      cls()
      raise BattleVictory

  def putToSea(self, destination):
    """Puts the ship to sea, with the final destionation of 'destination'.
    As opposed to 'sailTo', which is a short sail to a neighboring port close
    to the coast, 'putToSea' involves preparation for a significant sea
    voyage, buying supplies, and carries with it the inherent risk of pirate
    attacks, storms, and sea monsters. (kme)"""
    destination = int(destination) # for my sanity
    self.en_route_to = destination
    # In the current implementation, there is a 50% chance that you'll be
    # attacked by pirates (and sunk) and a 10% chance that you'll be eaten by
    # a giant squid.
    chance = randrange(1, 101)
    if chance < 50:         # attacked by pirates
      try:
        self.seaBattle()
      except ShipSunk:
        self.endGame()
        return
      except BattleVictory:
        pass
    # attacked by a giant squid (between 10 and 30%, depending on whether it's
    # SQUID_SEASON):
    elif 50 <= chance < 60 + SQUID_SEASON*30:
      printNow("Taipan, the seas are rough. We're likely to be attacked " +
               "by squid.")
      sleep(PAUSE_EVENT)
      printNow("\nGIANT SQUID OFF THE PORT BOW!!!")
      sleep(PAUSE_MSG)
      try:
        self.ship.causeDamage(SHIP_DAMAGE_SQUID)
      except ShipSunk:
        printNow("\nTaipan, the squid is overtaking the ship!")
        self.endGame()
        return
      else:
        printNow("\nTaipan, we've managed to stave off the squid...")
        printNow("We've taken a lot of damage, but we'll " +
                 "make it back to port this time.\n")
        sleep(PAUSE_EVENT)

    # Travel across open sea incurs a 5% penalty on ship's status:
    # FIXME: Your crew won't even try to warn you if this last 5% will sink
    # your ship!
    try:
      self.ship.causeDamage(SHIP_DAMAGE_SEA_VOYAGE)
    except ShipSunk:
      printNow("\nTaipan, bad storm brewing ahead!")
      sleep(PAUSE_EVENT)
      printNow("\nI don't think she'll hold together in this weather...")
      sleep(PAUSE_MSG_SHORT)
      cls()
      printNow("\nWe're losing her, Taipan. " + self.ship.getName() +
               " is sinking!")
      self.endGame()
      return
```

```
    # Made it this far...
    self.en_route_to=None
    #return self.ports[destination]
    self.current_port=self.ports[destination]

  def endGame(self):
    """Update the global quit_game so that the main game loop will quit.
    (kme)"""
    printNow("\n\nIt's been a pleasure serving with you, sir.\n")
    #printNow("ALL HANDS ABANDON SHIP!\n\n")
    printNow("~~~~~~~~~~~~~~~~~~~~\n" +
             " G A M E    O V E R\n" +
             "~~~~~~~~~~~~~~~~~~~~\n")
    #printNow("We know this call to exit() is going to fail because it " +
    #         "isn't properly implemented in JES\n")
    #exit()
    # Try this instead:
    #quit_game = True
    # Try this instead:
    raise EndGame
    return  # unnecessary?

  # def goHome(self, ship):
  #   """Immediately sail home (e.g., in case of severe battle damage)"""


################################################################################
#                        I N I T I A L I Z A T I O N                           #
################################################################################
def screenSizeAdjust():
  """Since we used requestStrings() for everything, JES makes it difficult to
  adjust screen height while the game is already running, so
  screenSizeAdjust() falls back on a 'raw_input' to allow the user to adjust
  her screen height before the game starts."""
  cls()
  printNow("RESIZE JES COMMAND WINDOW TO HERE\n" + '-'*70 + '\n'* 18 +
      "For the best experience, please resize the JES command window so " +
      "that the message\n'RESIZE JES COMMAND WINDOW TO HERE' above " +
      "is visible on your screen.\n\n" +
      "Press ENTER when ready to begin or 'q' to quit ...")
  resp = raw_input()
  if resp and resp[0].lower() == 'q':
    return False
  else:
    return True

def printHelp():
  """Print the two screens of help, pausing in between (kme)"""
  cls()  # clear the screen
  printNow(help_msg_1)
  requestString("Press ENTER for the next page of help.")
  cls()
  printNow(help_msg_2)

def runGame():
  """Run Taipan! (kme, with modifications by everyone else)"""
  quit_game = False
  # Allow user to adjust screen hard before we start in with the
  # requestStrings:
  if runningInJES():
    if not screenSizeAdjust():
      return  # bail out now

  printHelp()
  printNow('~'*70)
  resp = requestString("Let's begin, Taipan. What will you name your firm?")
  g = Game(firm_name=resp)
  printNow("Very well, sir. " + g.getName() + " will need a ship.\n")
  resp = requestString("What will you name your ship?")
  g.ship.setName(resp)
  sleep(PAUSE_MSG_SHORT)
  cls()
```

```
  while not quit_game:
    g.current_port.arrivalMessage()
    g.incrementVisitsForPortNumber(g.current_port.getPortNumber())
    g.ship.printStatus()
    g.printPortMenu()
    printNow('')

    # Allow ship repairs if ship status is <90% and you're in Hong Kong
    # TODO: fix this to be instanceof(HomePort) once the HomePort class is
    # fleshed out.
    if g.current_port.getName() == "Hong Kong":
      g.current_port.doBusinessWithBrotherWu(g.ship)
      if g.ship.getCash() >= RICH_ENOUGH_TO_RETIRE:
        cls()
        printNow("Taipan, you have had a successful career and amassed " +
                 "great wealth.\nI think it's high time you retired to a " +
                 "quiet home in the country!")
        try:
          # This isn't pretty, but we want to ignore the exception that
          # endGame() raises and just 'return' (quitting the game):
          g.endGame()
        except EndGame:
          return #quit the program

      elif g.timesAroundTheWorld() >= RETIREMENT_AGE:
        cls()
        printNow("Taipan, you have had a successful career and bravely " +
                 "sailed the high seas for\nmany a year. I think it's " +
                 "high time you retired to a quiet home in the country!")
        try:
          g.endGame()
        except EndGame:
          return #quit the program
      elif g.ship.getCondition() < 90:
        g.ship.doShipRepairs()
      cls()
      g.current_port.arrivalMessage()
      g.ship.printStatus()
      g.printPortMenu()

    printNow('')

    resp = None
    # Process input for the 'Where shall we sail to?' prompt, allowing N,S,E,W
    # as well as ports 1-7, [H]elp and [Q]uit.
    while not resp:  # loop while the user keeps giving bad input
      resp = requestString("Where shall we sail to, Taipan?\n" +
                           "(or [H]elp or [Q]uit)")
      # Have to check this first, otherwise string methods below will fail.
      if not resp: # empty or None
        continue
      # Now, remove leading and trailing spaces, take first letter of
      # response, and make it lower case:
      resp = resp.strip()[0].lower()
      if resp == 'q':
        quit_game = True
        continue
      elif resp == 'h':
        printHelp()
        requestString("Press ENTER to continue.")
        resp = True  # just to make sure we break out of this while loop in
                     # case the user pressed "Cancel"
      elif resp in directions.keys():
        # Go in that direction.
        resp = g.current_port.getPortToThe(resp).getPortNumber()
        #resp = g.current_port.getPortNumberToThe('n')  # oops (kme)
        try:
          g.sailTo(resp)
        except EndGame:
          quit_game = True
          continue
      elif resp == 'r':
        # Check retirement status:
```

```
        cls()
        printNow("Port visit count: %s" % str(g.getPortVisits()))
        printNow("You have sailed around the world %i time(s)."
                 % g.timesAroundTheWorld())
        printNow("You may retire after %i time(s) around the world."
            % RETIREMENT_AGE)
        requestString("Press ENTER to continue.")
        resp = True # ensure we re-print the port arrival message
      elif resp.isdigit():
        #g.current_port = g.sailTo(resp)
        try:
          g.sailTo(resp)
        except EndGame:
          quit_game = True
          continue
      else: # invalid response
        resp = None # clear the response, so the while loop won't stop
        continue

if __name__ == "__main__":
  # If we're running interactively:
  runGame()

# End L11.py
# vim:tw=78 sw=2 ts=2 expandtab
```