```
#
#
# Aravind Kumar Kaspe
# Banner ID: 001291145
#
# Description : In this program, first a Stack and a Queue are instantiated,
#               which are used to store player's information as player object.
#               The program then calls a method called getPlayers() that reads
#               information from a data file and randomly places each player's
#               details onto either the Stack or the Queue. Following this, the
#               initial status of both the Stack and the Queue is printed. Then
#               code proceeds to execute 20 random moves, which could involve
#               operations such as popping from the Stack and enqueueing into
#               the Queue , popping from the Stack, dequeueing from the Queue and
#               pushing onto the stack, or dequeueing from the Queue. These moves
#               reflect dynamic interactions with the data structures,
#               simulating changes in the arrangement of player information in
#               stacks and Queues.
#
#
#
from ArrayStack import *
from ArrayQueue import *
from Player import *
from Empty import *
from random import *

def getPlayers(myS, myQ):
    #
    # Opens the data file of player info... number, firstName, lastName... reads
    # each line of data as a str, divides the line into the 3 values...
    # str, str, str... converts the 1st str into an int (number)... combines the
    # 2nd & 3rd str objects into 1 str (name)... instantiates a new Player object,
    # using the constructor to set the number & name values... then, randomly
    # pushes the object onto a stack, or enqueues the object into a queue...
    #
    #
    # myS An object of the ArrayStack class
    # Initially empty - Player objects randomly pushed onto myS
    #
    # myQ An object of the ArrayQueue class
    # Intitially empty - Player objects randomly enqueued into myQ
    #
    # No return value
    #
    with open('5930 - MP6 Data.txt') as data:
        player_data = data.readline()
        while player_data:
            player_list = player_data.split()
            number = int(player_list.pop(0))
            name = ' '.join(player_list)
            p = Player(number,name)
            player_choice = ['stack','Queue']
            b = choice(player_choice)
            if b == 'stack':
                myS.push(p)
            else:
                myQ.enqueue(p)

            player_data = data.readline()


def currentStatus(myS, myQ):
    #
    # Prints the contents of the Player object at the top of the stack, along with
the
    # number of elements currently in the stack. If the stack is empty, an
appropriate
    # message is indicated. Then, does the same for the Player object at the front
of
    # the queue. If the queue is empty, an appropriate message is indicated.
    #
    # myS An object of the ArrayStack class - could be empty
    #
    # myQ An object of the ArrayQueue class - could be empty
    #
    # No return value
    #
    if len(myS) == 0:
        print("Stack is empty (0)... Queue Front: {} {}
({}).".format((myQ.first()).getNumber(), (myQ.first()).getName(),len(myQ)))
    elif len(myQ) == 0:
        print("Stack Top: {} {}({})... Queue is empty
(0).".format((myS.top()).getNumber(), (myS.top()).getName(),len(myS)))
    else:
        print("Stack Top: {} {}({})... Queue Front: {} {}
({}).".format((myS.top()).getNumber(),
(myS.top()).getName(),len(myS),(myQ.first()).getNumber(),
(myQ.first()).getName(),len(myQ)))


#
#************Main Program************
#
myS = ArrayStack()
myQ = ArrayQueue()
getPlayers(myS,myQ)

print("Initial Status:")
currentStatus(myS,myQ)
```

```
print()
count = 0

for i in range(20):
    count += 1
    print("Move {}:".format(count))
    operation_list = ["SpopQenqueue","Spop","QdequeueSpush","Qdequeue"]
    operation = choice(operation_list)

    if operation == "SpopQenqueue" :
        if len(myS) == 0:
            print("Pop: Stack is empty.")
            currentStatus(myS, myQ)
            print()
        else:
            stack_top = myS.pop()
            print("Pop: {} {} then Enqueue.".format(stack_top.getNumber(), 
stack_top.getName()))
            myQ.enqueue(stack_top)
            currentStatus(myS, myQ)
            print()

    elif operation == "Spop":
        if len(myS) == 0:
            print("Pop: Stack is empty.")
            currentStatus(myS, myQ)
            print()
        else:
            stack_top = myS.pop()
            print("Pop: {} {}.".format(stack_top.getNumber(), stack_top.getName()))
            currentStatus(myS, myQ)
            print()

    elif operation == "QdequeueSpush":
        if len(myQ) == 0:
            print("Dequeue: Queue is empty.")
            currentStatus(myS, myQ)
            print()
        else:
            queue_first = myQ.dequeue()
            print("Dequeue: {} {} then Push.".format(queue_first.getNumber(), 
queue_first.getName()))
            myS.push(queue_first)
            currentStatus(myS, myQ)
            print()

    else:
        if len(myQ) == 0:
            print("Dequeue: Queue is empty.")
            currentStatus(myS, myQ)
            print()
        else:
            queue_first = myQ.dequeue()
            print("Dequeue: {} {}.".format(queue_first.getNumber(), 
queue_first.getName()))
            currentStatus(myS, myQ)
            print()
```