```
"""
Class to handle file response packets for socket based file downloader

3/8/21
Alex Burling
88866582
"""


import sys
import os

MAGIC_NO = 0x497E


"""Abstracts response packet operations"""
class FileResponse:

    def __init__(self, magic_no, type, status_code, data_len, data):
        self.magic_no = magic_no
        self.type = type
        self.status_code = status_code
        self.data_len = data_len
        self.data = data


    """Initialises a FileResponse object from a filename,
       used on the server side to generate the response"""
    @classmethod
    def from_filename(cls, filename):
        magic_no = MAGIC_NO
        type = 2
        if (filename == None): #If req validation throws an error
            status_code = 2
            data = ""
            data_len = 0
        elif (os.path.exists(filename)):
            status_code = 1
            file = open(filename, "rb")
            data = file.read()
            file.close()
            data_len = len(data)
        else: #Server can't find file
            status_code = 0
            data = ""
            data_len = 0
        return cls(magic_no, type, status_code, data_len, data)


    """Initialises a FileResponse object from a bytearray,
       used on the client side to reconstruct the response.
       Data remains as a bytearray in case a complete
       request wasn't recieved in the first chunk"""
    @classmethod
    def from_bytearray(cls, array):
        magic_no = hex(int.from_bytes([array[0], array[1]], 'big'))
        type = int.from_bytes([array[2]], 'big')
        status_code = int.from_bytes([array[3]], 'big')
        data_len = int.from_bytes([array[4], array[5], array[6], array[7]], 'big')
        data = bytearray(array[8:])
        return cls(magic_no, type, status_code, data_len, data)
```

```
"""Converts FileResponse data into a bytearray for
   transmission over socket and returns length for logging"""
def generate_packet(self):
    packet = bytearray()
    packet.extend(self.magic_no.to_bytes(2, 'big'))
    packet.extend(self.type.to_bytes(1, 'big'))
    packet.extend(self.status_code.to_bytes(1, 'big'))
    packet.extend(self.data_len.to_bytes(4, 'big'))
    packet.extend(self.data)
    packet_len = len(packet)
    return packet, packet_len


"""Validates packet structure on client side"""
def validate(self):
    if (self.magic_no != hex(MAGIC_NO)):
        sys.exit("RESPONSE PACKET MALFORMED")
    if (self.type != 2):
        sys.exit("RESPONSE PACKET MALFORMED")
    if (self.status_code == 0):
        sys.exit("SERVER COULDN'T FIND REQUESTED FILE")
    if (self.status_code == 2):
        sys.exit("SERVER RECIEVED MALFORMED PACKET")
    if (not self.check_len()):
        sys.exit("COULDN'T RECIEVE COMPLETE RESPONSE")


"""Used to append data to FileResponse in case a complete request
wasn't recieved in the first chunk """
def append_data(self, data):
    self.data.extend(data)


def check_len(self):
    return self.data_len == len(self.data)


def get_data(self):
    return self.data
```