```python
"""
Class to handle file request packets for socket based file downloader

3/8/21
Alex Burling
88866582
"""


import sys

MAGIC_NO = 0x497E


"""Abstracts request packet operations"""
class FileRequest:

    def __init__(self, magic_no, type, filename_len, filename):
        self.magic_no = magic_no
        self.type = type
        self.filename = filename
        self.filename_len = filename_len


    """Initialises a FileRequest object from a filename,
       used on the client side to generate the request"""
    @classmethod
    def from_filename(cls, filename):
        magic_no = MAGIC_NO
        type = 1
        filename = filename.encode('utf-8')
        filename_len = len(filename)

        if filename_len < 1 or filename_len > 1024:
            sys.exit("SPECIFIED FILENAME IS INVALID")

        return cls(magic_no, type, filename_len, filename)


    """Initialises a FileRequest object from a bytearray,
       used on the server side to reconstruct the request.
       Filename 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')
        filename_len = int.from_bytes([array[3], array[4]], 'big')
        filename = bytearray(array[5:])

        return cls(magic_no, type, filename_len, filename)


    """Converts FileRequest data into a bytearray for
       transmission over socket"""
    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.filename_len.to_bytes(2, 'big'))
        packet.extend(self.filename)
        return packet
```

```
"""Validates packet structure on server side"""
def validate(self):
    if (self.magic_no != hex(MAGIC_NO)):
        raise AssertionError()
    if (self.type != 1):
        raise AssertionError()
    if (self.filename_len < 1 or self.filename_len > 1024):
        raise AssertionError()


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


def check_len(self):
    return self.filename_len == len(self.filename)


def get_filename(self):
    return self.filename.decode("utf-8")
```