```python
import os
import time
import socket
from FileRequest import FileRequest, decodeFixedHeader
from FileResponse import FileResponse



# Fixed constants
BUFFER_SIZE = 10000


def currentTime():
    """
    Returns the current time
    """
    return time.strftime("%H:%M:%S", time.localtime())


def checkFile(soc, fd, fileName):
    """
    Attempting to read the file to see if it exists
    locally in the Server and is readable.
    - Returns Status Code of 1 when file exists and
      can be opened, otherwise 0.
    """
    errorMessage = "\n-ERROR: File doesn't exist locally in" \
        " the Server, closing and aborting...".encode('utf-8')
    successMessage =  "\n-SUCCESS: File does exist locally in" \
        " the Server, Transferring data now...".encode('utf-8')

    if not os.path.exists("Server/"+fileName):
        fd.send(errorMessage)
        fd.close()      # Closing th File Directory (fd) socket
        runServer(soc)  # Restating the loop process

    fd.send(successMessage)

    return True


def readFile(soc, fd, fileName, close):
    """
    Reading byte string data from the file.
    """
    try:
        fOpen = open("Server/"+fileName, 'rb')
        fRead = fOpen.readlines()
        if close:
            fOpen.close()
    except FileNotFoundError as err:
        print(str(err)+'\n')
        fd.close()      # Closing th File Directory (fd) socket
        runServer(soc)  # Restating the loop process

    return fRead


def sendResponse(soc, fd, fRead, fileName):
    """
    Sends byte data detailing the information the Client would like to
    retrieve from the Server.
    """
    record = bytearray(0)
    number = 0x497E
    dataLength = 0
    _type = 2

    # Gets the status code
    statusCode = checkFile(soc, fd, fileName)

    if statusCode == 0:
        dataLength = 0

    fr = FileResponse(number, statusCode, dataLength, _type)
    fr.encodeFixedHeader(record)

    # Concatenating string byte data with the 8 byte fixed header
    for line in fRead:
        record += line
        dataLength += len(line)

    fd.send(record)  # Sending file to Client

    # Clean up with message
    readFile(soc, fd, fileName, True)
    fd.close()      # Closing th File Directory (fd) socket

    if dataLength == 0:
        print("- Nothing to send from file.")
    else:
        print("A total of {0} bytes transferred succesfully " \
        "from the Server to the Client.\n".format(dataLength))
    runServer(soc)  # Restating the loop process


def fileRequest(soc, fd, data, startTime):
    """
    Checking to see if the server can can send the
    client a certain file if it exists.
    """
    # Checks the 5 byte fixed header
    (magicNum, _type, fileNameLen) = decodeFixedHeader(data)

    # If the time gap is greater then 1, restart process
    if (time.clock()-startTime) >= 1.0:
        print("\nERROR: File Request is erroneous, aborting...")
        print("Please try again.\n")
        fd.close()      # Closing th File Directory (fd) socket
        runServer(soc)  # Restating the loop process

    # Checking the validity of the File Request
    fr = FileRequest(magicNum, fileNameLen, _type)
    if fr.requestChecker():
        print("\nERROR: Couldn't read the record from the socket...")
        print("Please try again.\n")
        fd.close()      # Closing th File Directory (fd) socket
        runServer(soc)  # Restating the loop process

    fileName = data[5:].decode('utf-8')  # decoding the file name string byte data

    if len(fileName) != fileNameLen:
        print("\nERROR: The file could not be read properly...")
        print("Please try again.\n")
        fd.close()      # Closing th File Directory (fd) socket
        runServer(soc)  # Restating the loop process

    return fileName


def acceptSocket(soc):
    """
    Printing server acceptance message.
    """
    port = soc.getsockname()[1]
    fd, addr = soc.accept()
    print("-- {0}  IP = {1}  Port = {2}".format(currentTime(), addr[0], port))

    return fd


def setUpServer():
    """
    Checking for errors and setting up the server.
    """
    # Analysing the entered port number
    port = int(input("Please enter in a Port Number:\n>> "))
    if port < 1024 or 64000 < port:
        print("\nERROR: Port number '{0}' is not within values 1,024 and 64,000...".format(port))
        print("Terminating Program")
        exit()

    # Attempting to create a socket
    try:
        soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    except socket.error as e:
        print('\n',str(e))
        exit()

    # Attempting to bind to the port number
    try:
        soc.bind(('', port))
    except socket.error as e:
        print('\n',str(e))
        exit()

    # Attempting to listen for the socket
    try:
        soc.listen(1)
    except socket.error as e:
        print('\n',str(e))
        soc.close()
        exit()

    return soc


def runServer(soc):
    """
    Runs the server until closed/exited.
    """
    while True:
        fd = acceptSocket(soc)
        startTime = time.clock()  # Start timer
        data = fd.recv(BUFFER_SIZE)  # Data sent from Client through a socket
        fileName = fileRequest(soc, fd, data, startTime)
        fRead = readFile(soc, fd, fileName, False)
        sendResponse(soc, fd, fRead, fileName)


def main():
    """
    Runs and Controls the program flow of the server.
    """
    soc = setUpServer()
    print("Waiting for Client to connect...\n")
    runServer(soc)


main()
```