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


# Fixed constants
BUFFER_SIZE = 10000


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


def writeFile(fileName, fileData):
    """
    Writing byte string data into the file.
    """
    fOpen = open("Client/"+fileName, 'w')
    fOpen.write(fileData)


def readResponse(soc, data, startTime, msgServer, fileName):
    """
    Read a Response from the Client.
    """
    fixedHeader = data[:8]  # Fixed Header byte array
    acturalData = data[8:]  # Actual Data byte array

    if len(data) <= 8:
        print("File doesnt exist")
        soc.close()  # Closing the socket
        exit()

    # Checks the first 8 bytes
    (magicNum, _type, statusCode, dataLength) = decodeFixedHeader(fixedHeader)

    # Determining the amount of recieved bytes
    if len(acturalData) == 0:
        print("\n- Noting was in the file.")
        soc.close()  # Closing the socket
        exit()
    else:
        print(msgServer)

    # If the time gap is greater then 1, terminate the program
    if (time.clock()-startTime) >= 1.0:
        print("\nERROR: File Response is erroneous...\nTerminating Program")
        soc.close()  # Closing the socket
        exit()

    # Checking the validity of the File Response
    fr = FileResponse(magicNum, statusCode, dataLength, _type)
    if fr.responseChecker():
        print("\nERROR: File Response is erroneous...\nTerminating Program")
        soc.close()  # Closing the socket
        exit()

    # Reading the actual data sent from Server
    if statusCode:
        writeFile(fileName, acturalData.decode('utf-8'))
        soc.close()  # Closing the socket
        exit()
    else:
        print("\nERROR: Unable to write the file you requested..." +
              "\nTerminating Program")
        soc.close()  # Closing the socket
        exit()
```

```
def sendRequest(soc, fileName):
    """
    Sends byte data detailing the information the Client would like to
    retrieve from the Server.
    """
    record = bytearray(0)
    number = 0x497E

    fr = FileRequest(number, fileName)
    fr.encodeFixedHeader(record)

    # Sending Info to the Server
    soc.send(record)
    soc.send(fileName.encode('utf-8'))


def setUpClient():
    """
    Checking for errors and setting up the server.
    """
    # Analysing the entered host name, port number and file name
    try:
        (host, port, fileName) = input("Please enter in a Hosts Name, " +
        "Port Number and a File Name:\n>> ").split()
    except ValueError as e:
        print('\n',str(e))
        exit()

    try:
        addrInfo = socket.getaddrinfo(host, port)
    except socket.error as e:
        print('\n',str(e))
        exit()

    if int(port) < 1024 or 64000 < int(port):
        print("\nERROR: Port number '{0}' is not within values 1,024 and 64,000...".format(port))
        print("Terminating Program")
        exit()

    if os.path.exists("Client/"+fileName):
        print("\nERROR: File '{0}' already exists locally...\nTerminating Program.".format(fileName))
        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()
    startTime = time.clock()  # Start timer

    # Attempting to connect with the server
    try:
        soc.connect(addrInfo[0][-1])
    except socket.error as e:
        print('\n',str(e))
        exit()

    return (soc, fileName, startTime)


def runClient():
    """
    Runs and Controls the program flow of the Client.
    """
    (soc, fileName, startTime) = setUpClient()
    sendRequest(soc, fileName)  # Sending a request
    msgServer = soc.recv(BUFFER_SIZE).decode('utf-8')  # Print server status on file

    # Attempting to read file
```

```
    if msgServer[2:7] != "ERROR":
        data = soc.recv(BUFFER_SIZE)  # Data sent from Client through a socket
        readResponse(soc, data, startTime, msgServer, fileName)  # Read response from server
    else:
        print(msgServer)
        soc.close()  # Closing the socket
        exit()


runClient()
```