PROJECT 1 REPORT

NFTNet server starts the welcoming socket and GeckoClient access to this socket. After arriving to NFTNet, NFTNet sends the client to ClientHandler class. In this ClientHandler class and in the GeckoClient class they are under some protocols. To request data or parse the response they are modifying their methods for this protocol.

For example, the client implements the protocol as to send request, they enter the Method as REQUEST and change their Commands relative to the usage.

```
public void sendListRequest() throws IOException, JSONException {
    JSONObject request = new JSONObject();
    request.put("Method", "REQUEST");
    request.put("Command", "LIST");
    out.println(request.toString());
}

public void sendNFTRequest(String id) throws IOException, JSONException {
    JSONObject request = new JSONObject();
    request.put("Method", "REQUEST");
    request.put("Command", "NFT");
    request.put("ID", id);
    out.println(request.toString());
}
```

In also parsing the client class apply the protocol rules:

```
//print statements
public static void printList(String jsonResponse) throws JSONException {
    JSONObject response = new JSONObject(jsonResponse);
    //System.out.println(response);
    JSONArray nfts = response.getJSONArray("Body");
    for (int i = 0; i < nfts.length(); i++) {
        JSONObject nft = nfts.getJSONObject(i);
        String symbol = nft.optString("symbol", "N/A");
        String name = nft.optString("name", "N/A");
        String platformId = nft.optString("asset_platform_id", "N/A");
        String contractAddress = nft.optString("contract_address", "N/A");
        System.out.println("Symbol: "+ symbol+ " Name: "+ name+ " Platform ID: "+ platformId+ " Contract Address: "+contractAddress);
    }
}

public static void printNFT(String jsonResponse) throws JSONException {
    JSONObject nft = new JSONObject(jsonResponse);
    //System.out.println(nft);
    String name = nft.getJSONObject("Body").optString("name", "N/A");
    String platformId = nft.getJSONObject("Body").optString("asset_platform_id", "N/A");
    double priceInUsd = nft.getJSONObject("Body").getJSONObject("floor_price").optDouble("usd", 0.00);
    System.out.println("Name: " + name + "\nPlatform ID: " + platformId + "\nPrice in USD: $" + String.format("%.2f", priceInUsd));
}
```

In ClientHandler, we also check for the Protocol format:

```
private void processRequest(JSONObject requestJson) throws JSONException {
    String method = requestJson.getString("Method");
    //System.out.println("method:" +method);
    if (method.equalsIgnoreCase("REQUEST")) {
        handleRequest(requestJson);
    }
    else {sendResponse("404", "Unknown message type");}
}
```

And handle the request relative to command:

```
private void handleRequest(JSONObject requestJson) throws JSONException {
    if (serverRate.attemptLogin()) {
        //System.out.println("we are in handlerequest");
        String command = requestJson.getString("Command");
        //System.out.println("no error in command");
        //System.out.println("we are in handlerequest");
        try {
            switch (command) {
                case "LIST":
                    JSONArray nft_list = fetchListOfNFTs();
                    //System.out.println(nft_list);
                    sendResponse(nft_list);
                    break;
                case "NFT":
                    System.out.println(requestJson);
                    String id = requestJson.getString("ID");
                    JSONObject nftDetails = fetchNFTData(id);
                    System.out.println("data: "+ nftDetails );
                    sendResponse(nftDetails);;
                    break;
                default:
                    sendResponse("NO_ACTION", "Invalid action");
                    break;
            }
        } catch (Exception e) {
            sendResponse("HANDLING ERROR",e.getMessage());
        }
    } else {
        sendResponse("LIMIT EXCEEDED", "There are more than 50 clients");
```

Also sending response is another example for protocol usage:

```
private void sendResponse(String errorCode, String errorMessage) throws JSONException {
    JSONObject errorResponse = new JSONObject();
    errorResponse.put("Method", "ERROR");
    errorResponse.put("ErrorCode", errorCode);
    errorResponse.put("ErrorMessage", errorMessage);
    out.println(errorResponse.toString());
}

private void sendResponse(JSONObject responseData) throws JSONException {
    JSONObject response = new JSONObject();
    response.put("Header", new JSONObject().put("Method", "RESPONSE"));
    response.put("Body", responseData);
    out.println(response.toString());
}

private void sendResponse(JSONArray responseData) throws JSONException {
    JSONObject response = new JSONObject();
    response.put("Method", "RESPONSE");
    response.put("Body", responseData);
    out.println(response.toString());
}
```