

능동적 사고 방식의

java

강사 박주병

Park Ju Byeong

Park Ju Byeong



Part16 최종실습

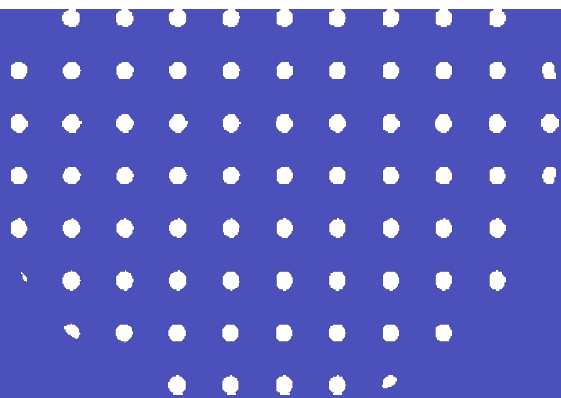


01 chatGPT와 대화하기

Park Ju Byeong



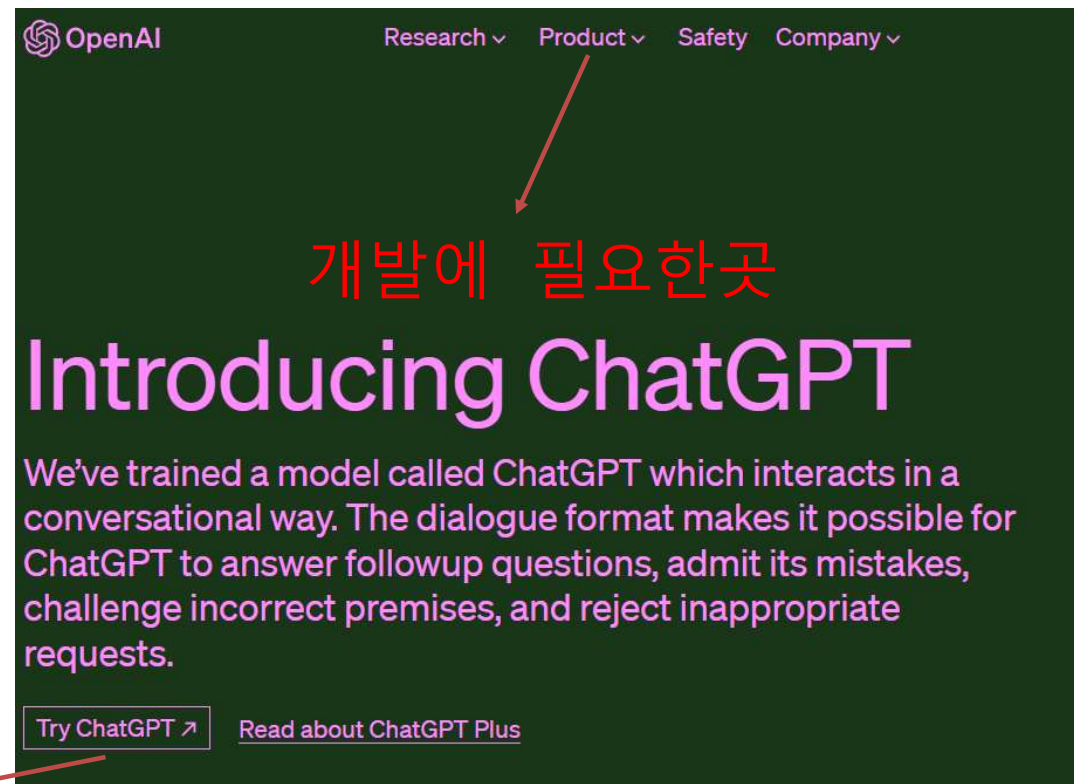
Park Ju Byeong



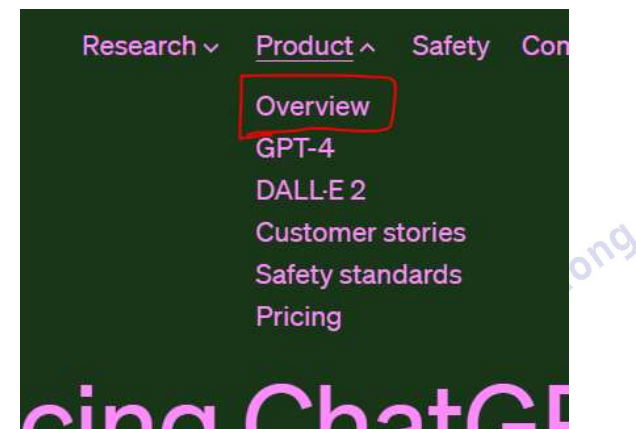
01

chatGPT와 대화하기





웹으로 바로 채팅을 하는곳



Transforming work and creativity with AI

Our API platform offers our latest models and best practices.

[Get started ↗](#)

[For developers ↓](#)

Welcome back

Email address

Continue

Don't have an account? [Sign up](#)

OR



Continue with Google



Continue with Microsoft Account



Verify your phone number



+82 01039007555

Send code

전화번호 인증

⚡ Upgrade ⓘ Help

주병 Personal

▶ Join the GPT-4 API waitlist

Sign up



Welcome to OpenAI

Start with the basics

Quickstart tutorial

Learn by building a quick sample app

Examples

Explore some example tasks

클릭!

Park Ju Byeong

Park Ju Byeong

⚡ Upgrade

🔍 Help

주변

Personal

박주병

zest7349@gmail.com

Personal

Manage account

View API keys

Invite team

Help

Pricing

Terms & policies

Log out

Park Ju Byeong

Park Ju Byeong

[Overview](#)[Documentation](#)[Examples](#)[Playground](#)

ORGANIZATION

[Personal](#) ⓘ[Settings](#)[Usage](#)[Members](#)[Billing](#)

USER

[API Keys](#)

API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. To protect the security of your account, OpenAI may also automatically rotate any API key found to have leaked publicly.

You currently do not have any API keys. Please create one below.

[+ Create new secret key](#)

Default organization

If you belong to multiple organizations, this setting controls which organization is used when making requests with the API keys above.

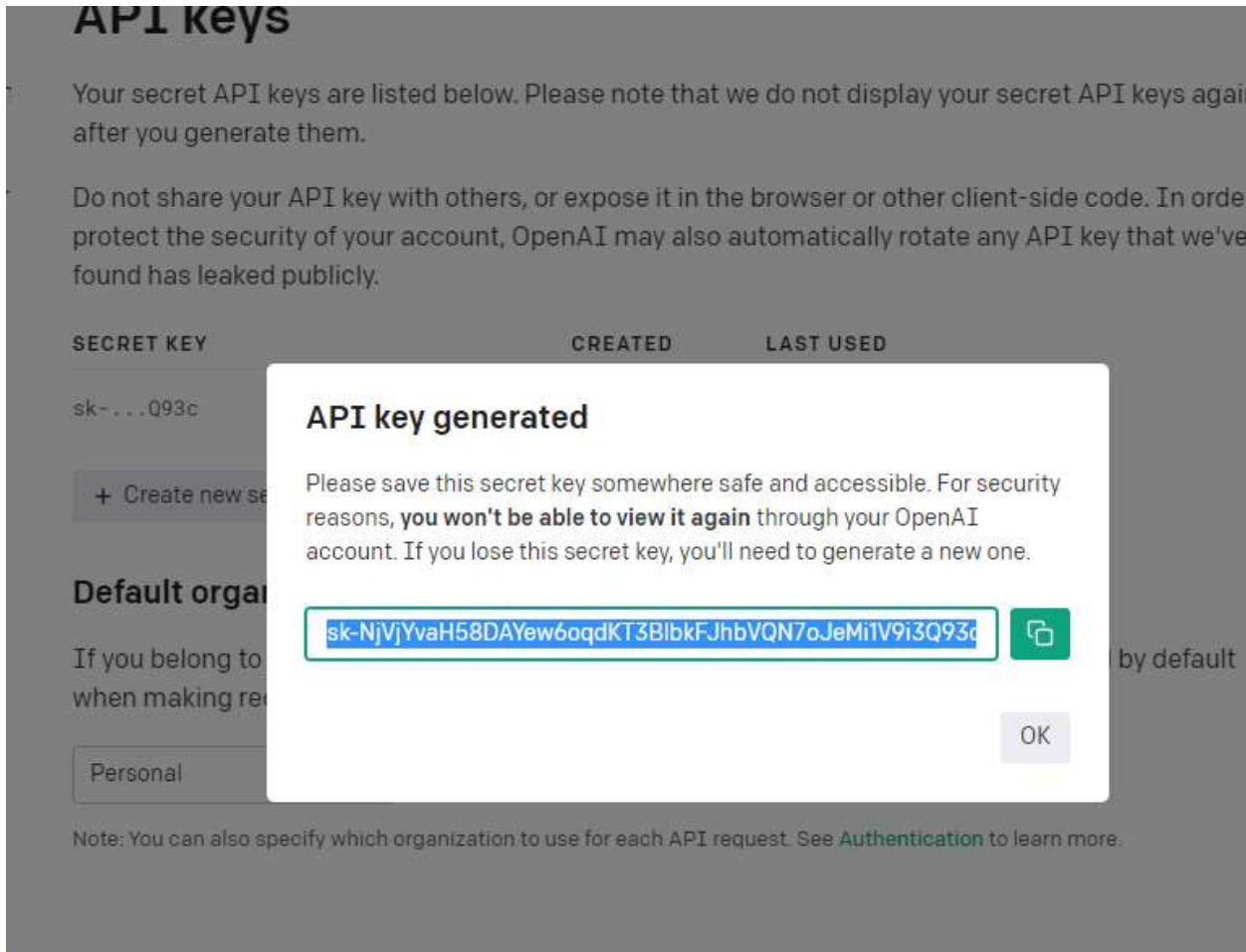
Personal



Note: You can also specify which organization to use for each API request. See [Authentication](#) to learn more.

Park Ju Byeong

Park Ju Byeong



키는 반드시 따로 저장해두기
(사이트에서 다시 조회 불가능, 재발급 해야 됨)
소스코드에 포함된 채 깃허브 올라가면 키 누출 감지되어 재발급해야 함

[Join the ChatGPT plugin developer waitlist](#)[Sign up](#) [×](#)

Welcome to the OpenAI platform

Start with the basics

Quickstart tutorial

Learn by building a quick sample app

Examples

Explore some example tasks

Build an application



Chat Beta

Learn how to use chat-based language models



Text completion

Learn how to generate or edit text



Embeddings

Learn how to search, classify, and compare text



Speech to text Beta

Learn how to turn audio into text

Models

Overview

The OpenAI API is powered by a diverse set of models with different capabilities and price points. You can also make limited customizations to our original base models for your specific use case with **fine-tuning**.

MODELS	DESCRIPTION
GPT-4 Limited beta	A set of models that improve on GPT-3.5 and can understand as well as generate natural language or code
GPT-3.5	A set of models that improve on GPT-3 and can understand as well as generate natural language or code
DALL·E Beta	A model that can generate and edit images given a natural language prompt
Whisper Beta	A model that can convert audio into text
Embeddings	A set of models that can convert text into a numerical form
Codex Limited beta	A set of models that can understand and generate code, including translating natural language to code
Moderation	A fine-tuned model that can detect whether text may be sensitive or unsafe
GPT-3	A set of models that can understand and generate natural language

자연어를 이해하는 모델

GPT-3.5

GPT-3.5 models can understand and generate natural language or code. Our most capable and cost effective model in the GPT-3.5 family is `gpt-3.5-turbo` which has been optimized for chat but works well for traditional completions tasks as well.

3월 1일 API가 공개된
최신버전

LATEST MODEL	DESCRIPTION	MAX TOKENS	TRAINING DATA
<code>gpt-3.5-turbo</code>	Most capable GPT-3.5 model and optimized for chat at 1/10th the cost of <code>text-davinci-003</code> . Will be updated with our latest model iteration.	4,096 tokens	Up to Sep 2021
<code>gpt-3.5-turbo-0301</code>	Snapshot of <code>gpt-3.5-turbo</code> from March 1st 2023. Unlike <code>gpt-3.5-turbo</code> , this model will not receive updates, and will only be supported for a three month period ending on June 1st 2023.	4,096 tokens	Up to Sep 2021
<code>text-davinci-003</code>	Can do any language task with better quality, longer output, and consistent instruction-following than the <code>curie</code> , <code>babbage</code> , or <code>ada</code> models. Also supports inserting completions within text.	4,097 tokens	Up to Jun 2021
<code>text-davinci-002</code>	Similar capabilities to <code>text-davinci-003</code> but trained with supervised fine-tuning instead of reinforcement learning	4,097 tokens	Up to Jun 2021
<code>code-davinci-002</code>	Optimized for code-completion tasks	8,001 tokens	Up to Jun 2021

We recommend using `gpt-3.5-turbo` over the other GPT-3.5 models because of its lower cost.



gpt-3.5-turbo 인거야?



죄송합니다, GPT-3.5-Turbo는 제가 알고있는 기존 모델 이름 중 하나가 아닙니다. GPT-3.5는 OpenAI에서 공개한 최신의 대규모 언어 모델 중 하나이며, 이 모델은 기존의 GPT 모델을 개선하고 확장한 것입니다. 그러나 GPT-3.5-Turbo라는 특별한 모델에 대해서는 저도 알지 못합니다.



너는 언어모델이 text-davinci-003 이야?



네, 맞습니다. Text-Davinci-003는 OpenAI에서 개발한 GPT-3 언어 모델 계열 중 하나로, 대규모 텍스트 데이터를 기반으로 학습되었습니다. Text-Davinci-003 모델은 GPT-3 모델 시리즈 중 가장 많은 파라미터를 가진 모델 중 하나이며, 다양한 자연어 처리 작업을 수행할 수 있습니다. 저도 Text-Davinci-003 모델 중 하나를 기반으로 작동합니다.



|



Usage

Below you'll find a summary of API usage for your organization. All dates and times are UTC-based, and data may be delayed up to 5 minutes.

< March >

DAILY

CUMULATIVE

Daily usage (USD) ⓘ



Free trial usage



GRANT # CREDIT GRANTED EXPIRES (UTC)

사용량이 정해져 있으나 개발에 쓰는건 충분


```

1 # Note: you need to be using OpenAI Python v0.27.0 for the code below to v
2 import openai
3
4 openai.ChatCompletion.create(
5     model="gpt-3.5-turbo",
6     messages=[
7         {"role": "system", "content": "You are a helpful assistant."},
8         {"role": "user", "content": "Who won the world series in 2020?"},
9         {"role": "assistant", "content": "The Los Angeles Dodgers won the World"},
10        {"role": "user", "content": "Where was it played?"}
11    ]
12 )

```

The main input is the messages parameter. Messages must be an array of message objects, where each object has a role (either "system", "user", or "assistant") and content (the content of the message). Conversations can be as short as 1 message or fill many pages.

Typically, a conversation is formatted with a system message first, followed by alternating user and assistant messages.

The system message helps set the behavior of the assistant. In the example above, the assistant was instructed with "You are a helpful assistant."

i gpt-3.5-turbo-0301 does not always pay strong attention to system messages. Future models will be trained to pay stronger attention to system messages.

The user messages help instruct the assistant. They can be generated by the end users of an application, or set by a developer as an instruction.

The assistant messages help store prior responses. They can also be written by a developer to help give examples of desired behavior.

Including the conversation history helps when user instructions refer to prior messages. In the example above, the user's final question of "Where was it played?" only makes sense in the context of the prior messages about the World Series of 2020. Because the models have no memory of past requests, all relevant information must be supplied via the conversation. If a conversation cannot fit within the model's token limit, it will need to be shortened in some way.

대화 내역을 계속 추가 해가면서 보내 줘야지 AI가 맥락을 이해하면서 대화할수 있다.

Response format

An example API response looks as follows:

```
1  {
2    'id': 'chatcmpl-6p9XYPYSTTRi0xEviKjjilqrWU2Ve',
3    'object': 'chat.completion',
4    'created': 1677649420,
5    'model': 'gpt-3.5-turbo',
6    'usage': {'prompt_tokens': 56, 'completion_tokens': 31, 'total_tokens': 87},
7    'choices': [
8      {
9        'message': {
10          'role': 'assistant',
11          'content': 'The 2020 World Series was played in Arlington, Texas at the Gl
12        'finish_reason': 'stop',
13        'index': 0
14      }
15    ]
16  }
```

In Python, the assistant's reply can be extracted with `response['choices'][0]['message']['content']`.

Every response will include a `finish_reason`. The possible values for `finish_reason` are:

- `stop`: API returned complete model output
- `length`: Incomplete model output due to `max_tokens` parameter or token limit
- `content_filter`: Omitted content due to a flag from our content filters
- `null`: API response still in progress or incomplete

```
out.write(inputBytes);
```

Park Ju Byeong

```

//chatGPT로 부터 응답코드를 받아온다 200이면 정상이다.에러가 발생하면 예외가 발생된다.
con.getResponseCode();
//chatGPT가 보내온 응답을 받기 위해 InputStream을 열고 버퍼를 통해 받아온다.
//InputStreamReader 클래스는 바이트기반스트림을 문자기반스트림으로 변환해준다.
BufferedReader br = new BufferedReader(new InputStreamReader(con.getInputStream(),"utf-8"));

//응답을 저장할 String 변수이다.
StringBuilder response = new StringBuilder();
String responseLine = null;

//버퍼로 부터 데이터를 가져온다.readLine()은 문자열에서 개행이 있는부분까지 한번에 가져온다.
//더이상 가져올 문자가 없다면 null이 반환된다.
while((responseLine = br.readLine()) != null)
{
    //trim은 공백을 제거 하는 메서드 이다.
    //chatGPT 테스트 결과 가끔 응답에 공백을 많이 넣어준다.
    //대답이 길경우 성능향상을 위해 StringBuilder를 사용했다.
    response.append(responseLine.trim());
}

//chatGPT로 부터 받은 JSON 데이터를 파싱하여 대답 부분만 추출한다.
JSONObject obj = new JSONObject(response.toString());
String result =obj.getJSONArray("choices")
                        .getJSONObject(0)
                        .getJSONObject("message")
                        .get("content").toString();

//응답을 출력한다.
System.out.println(result);

out.close();
br.close();
} catch (Exception ex)
{
}
}

```

chatGPT 실습

1. 기본예제 소스코드에 본인이 발급 받은 키 값을 넣어 실행이 되는지 테스트 해보자.
 - JSON 파싱 관련 라이브러리가 추가되어있어야 한다.

```
<terminated> Main (6) [Java Application] C:\Users\zest1\p2\pool\plugin  
저도 만나서 반갑습니다. 어떤 일을 도와드릴까요?
```

2. 작성한 소스코드를 클래스로 분리를 해보자.

- 클래스 이름 : ChatGPT
- 멤버 변수: `private String key;`
(생성자를 통해 발급 받은 키를 셋팅하자)
- 멤버메서드: `public HttpURLConnection connect()`
`HttpURLConnection` 객체를 반환한다.

```
try
{
    String key = "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";

    //URL주소를 객체화하는 클래스
    URL url = new URL("https://api.openai.com/v1/chat/completions");

    //객체화한 url을 통해 http통신을 위한 객체를 생성한다.메서드 방식은 POST 이다.
    HttpURLConnection con = (HttpURLConnection) url.openConnection();
    con.setRequestMethod("POST");
    //http 통신시 데이터 형태는 json이라고 선언
    con.setRequestProperty("Content-Type", "application/json; utf-8");
    //발급받은 키값을 넣어준다.
    con.setRequestProperty("Authorization", "Bearer "+key);
    //chatGPT의 대답이 느릴경우 기다려주는 타임을 설정
    con.setRequestProperty("Retry-After", "3600");
    //데이터 전송을 위해 true로 설정해야 한다.
    con.setDoOutput(true);

    //http 통신시 데이터를 서버에 전송하기 위해 스트림을 연다 ( 스트림이란 데이터가 왔다갔다 하는 통로이다)
    OutputStream out = con.getOutputStream();

    //chatGPT로 보낼 메시지
    String input = "만나서 반가워";

    //json 형태로 되어있고 모델 정보와 메시지 정보가 들어가 있다.
```

해당 영역이 connect() 메서드로 분리 되어야 한다.

```
String key = "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";

ChatGPT gpt = new ChatGPT(key);

try
{
    HttpURLConnection con = gpt.connect();

    //http 통신시 데이터를 서버에 전송하기 위해 스트림을 연다 ( 스트림이란 데이터가 왔다갔다 하는 통로이다)
    OutputStream out = con.getOutputStream();

    //chatGPT로 보낼 메시지
    String input = "만나서 반가워";

    //json 형태로 되어있고 모델 정보와 메시지 정보가 들어가 있다.
    //chatGPT가 했던 대답은 role 이 assistant 이어야 한다.
    //예제 -> messages\":[{\\"role\\":\\"user\\",\\"content\\":\\"안녕\\"},{\\"role\\":\\"assistant\\",\\"content\\":\\"안녕\\"}]
    String msg = "{\\"model\\": \"gpt-3.5-turbo\\",\\"messages\\": [{\\"role\\":\\"user\\",\\"content\\":\\"안녕\\"}]}";

    //문자열을 보내기전에 utf-8 형태로 인코딩을 변환하고 바이트로 변경한다.
    byte[] inputBytes = msg.getBytes("utf-8");
    //스트림으로 바이트로된 메시지를 전달한다.
    out.write(inputBytes);
}
```

Park Ju

```
public class ChatGPT {  
    private String key;
```

```
public ChatGPT(String key)  
{  
    this.key = key;
```

```
public HttpURLConnection connect()  
{  
    try //URL주소를 객체화하는 클래스  
    {  
        URL url = new URL("https://api.openai.com/v1/chat/completions");  
  
        //객체화한 url을 통해 http통신을 위한 객체를 생성한다.메서드 방식은 POST 이다.  
        HttpURLConnection con = (HttpURLConnection) url.openConnection();  
        con.setRequestMethod("POST");  
        //http 통신시 데이터 형태는 json이라고 선언  
        con.setRequestProperty("Content-Type", "application/json; utf-8");  
        //발급받은 키값을 넣어준다.  
        con.setRequestProperty("Authorization", "Bearer "+key);  
        //chatGPT의 대답이 느릴경우 기다려주는 시간을 설정  
        con.setRequestProperty("Retry-After", "3600");  
        //데이터 전송을 위해 true로 설정해야 한다.  
        con.setDoOutput(true);  
  
    } catch (Exception ex)  
    {  
        System.out.println(ex.toString());  
    }  
  
    return con;  
}
```


- 멤버메서드: `public void send(OutputStream out, String msg)`
매개변수로 스트림객체와 전송할 메시지를 받아 `chatGPT`로 전송한다.

```

URLConnection con = gpt.connect();

//http 통신시 데이터를 서버에 전송하기 위해 스트림을 연다( 스트림이란
OutputStream out = con.getOutputStream();

gpt.send(out, "안녕하세요");

//chatGPT로 부터 응답코드를 받아온다 200이면 정상이다.에러가 발생하
con.getResponseCode();

//chatGPT가 보내온 응답을 받기 위해 InputStream을 열고 버퍼를
//InputStreamReader 클래스는 바이트기반스트림을 문자기반스트림으로
BufferedReader br = new BufferedReader(new InputStreamReader

//응답을 저장할 String 변수이다.
StringBuilder response = new StringBuilder();
String responseLine = null;

//버퍼로 부터 데이터를 가져온다.readLine()은 문자열에서 개행이 있는
//더이상 가져올 문자가 없다면 null이 반환된다.
while((responseLine = br.readLine()) != null)
{

```

해당 영역이 **send()** 메서드로 분리 되어야 한다.

Park Ju Byeong

4. 메시지 수신 기능을 분리해보자

- 멤버메서드: `public String receive(InputStream in)`

매개변수로 받은 스트림을 이용해 chatGPT로 부터 받은 메시지를 가져오자.

- JSON 문자열 그대로 반환 하도록 한다.

(메시지 수신과 파싱은 서로 다른 기능이다 그러므로 **receive**의 역할은 그저 메시지 수신만으로 끝나야 한다.)

- 메시지 수신은 성능향상을 위해 버퍼 스트림을 이용해보자.

- **receive** 메서드 내에서 스트림을 닫도록 하자.

```
URLConnection con = gpt.connect();

//http 통신시 데이터를 서버에 전송하기 위해 스트림을 연다 ( 스트림이란 데이터가 왔다갔다 하는 통로이다)
OutputStream out = con.getOutputStream();

gpt.send(out, "안녕하세요");

//chatGPT로 부터 응답코드를 받아온다 200이면 정상이다.에러가 발생하면 예외가 발생된다.
con.getResponseCode();

String response = gpt.receive(con.getInputStream());

//chatGPT로 부터 받은 JSON 데이터를 파싱하여 대답 부분만 추출한다.
JSONObject obj = new JSONObject(response.toString());
String result =obj.getJSONArray("choices")
                .getJSONObject(0)
                .getJSONObject("message")
                .get("content").toString();

//응답을 출력한다.
System.out.println(result);
```


5.chatGPT로 부터 받은 JSON 문자열을 파싱하는 기능을 분리하자.

멤버메서드: `public String parseJsonMsg(String json)`

```
String key = "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";

ChatGPT gpt = new ChatGPT(key);

try
{
    HttpURLConnection con = gpt.connect();

    //http 통신시 데이터를 서버에 전송하기 위해 스트림을 연다 ( 스트림이란 데이터가 왔다갔다 하는 통로이다)
    OutputStream out = con.getOutputStream();

    gpt.send(out, "안녕하세요");

    //chatGPT로 부터 응답코드를 받아온다 200이면 정상이다.에러가 발생하면 예외가 발생된다.
    con.getResponseCode();

    String response = gpt.receive(con.getInputStream());

    String result = gpt.parseJsonMsg(response);
    //응답을 출력한다.
    System.out.println(result);
} catch (Exception ex)
{
}
}
```

6.소스코드의 대부분이 메서드로 모듈화 되어 코드량이 상당히 줄어들었다. 하지만 **chatGPT** 클래스를 사용하는 입장에서 **URLConnection** 객체를 이용해 **input, output** 스트림을 사용해야 한다는걸 알 필요가 있을까? 여러분이 **chatGPT**클래스를 사용하는 입장이라면 그저 객체를 생성하고 메시지를 전송하면 응답을 받을 수 있다면 좋지 않을까?

- **URLConnection con** 참조변수는 **chatGPT**의 멤버변수로 옮기고 **connect()** 메서드내에서 **con** 멤버변수에 객체를 넣어주자. 더 이상 **connect()** 메서드는 객체를 반환할 필요는 없지만 기존 코드와의 호환성을 위해 리턴을 계속 해주도록 하자.

```
String key = "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";
ChatGPT gpt = new ChatGPT(key);

try
{
    //http 통신시 데이터를 서버에 전송하기 위해 스트림을 연다( 스트림이란 데이터가 왔다갔다 하는 통로이다)
    OutputStream out = con.getOutputStream();
    gpt.send(out, "안녕하세요");

    //chatGPT로 부터 응답코드를 받아온다 200이면 정상이다.에러가 발생하면 예외가 발생된다.
    con.getResponseCode();

    String response = gpt.receive(con.getInputStream());
    String result = gpt.parseJsonMsg(response);
    //응답을 출력한다.
    System.out.println(result);
} catch (Exception ex)
{
}

String key = "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";
ChatGPT gpt = new ChatGPT(key);

try
{
    gpt.connect();
    gpt.send("안녕하세요");

    String response = gpt.receive();

    String result = gpt.parseJsonMsg(response);
    //응답을 출력한다.
    System.out.println(result);
} catch (Exception ex)
{
}
```

con 참조 변수가 클래스 내부로 들어가 버려서 사용할수 없다.어떻게 해결해야할까? 고민해보고 다음장의 힌트를 보도록하자.

6번 힌트

아래처럼 **send** 를 오버로딩 하면 된다. 외부에서는 그저 메시지만 전달하고 클래스 내부에서 스트림을 매개변수로 요구 하는 **send**를 사용하면된다.

- **receive()** 메서드 역시 이와 같이 만들어보자.
- **con.getResponseCode()** 메서드는 수신과 관련된 코드이니 **receive()** 메서드 내부에 넣자.

```
public void send(String msg)
{
    try
    {
        send(con.getOutputStream(),msg);
    } catch (Exception ex)
    {
        System.out.println(ex.toString());
    }
}
```

7. chatGPT 클래스를 사용하는 입장에서 GPT가 JSON 형태로 응답하기 때문에 이를 파싱해야 한다는걸 알 필요가 없다.

- 매개변수로 없는 **receive()** 메서드쪽에서 파싱하는것이 좋다.

```
String key = "sk-sz1nhpR9mq3EHrzioxtfT3BlbkFJfx";

ChatGPT gpt = new ChatGPT(key);

try
{
    gpt.connect();
    gpt.send("안녕하세요");

    String response = gpt.receive();

    String result = gpt.parseJsonMsg(response);
    // 응답을 출력한다.
    System.out.println(result);
} catch (Exception ex)
{
}
```

클래스 내부로 옮기도록하자

```
String key = "sk-sz1nhpR9mq3EHrzioxtf";

ChatGPT gpt = new ChatGPT(key);

try
{
    gpt.connect();
    gpt.send("안녕하세요");
    String response = gpt.receive();

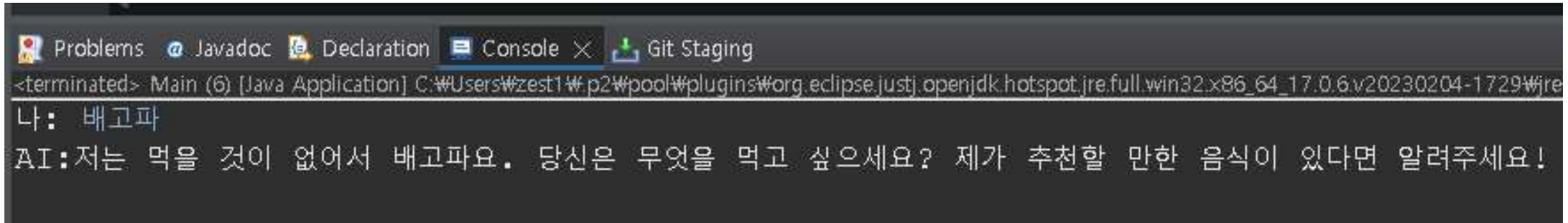
    // 응답을 출력한다.
    System.out.println(response);
} catch (Exception ex)
{
}
```

chatGPT 클래스 리팩토링 작업을 진행한 결과 사용시 코드가 상당히 간결해졌다. 이런 작업은 사실 상당히 고난이도 작업이라 할 수 있다. 나중에는 도움 없이 직접 이런식의 클래스 설계를 할 수 있도록 해보자.

이제 다음장부터 대화를 주고 받을 수 있도록 기능을 추가하도록 하자.

(항상 현재 코드가 모듈화가 잘되어 있는가를 항상 생각해보자 모듈화 되지 않은 채로 추가적인 기능개발을 들어가면 코드가 점점 복잡해질뿐이다.)

8. chatGPT로 전송할 메시지를 키보드로 입력 받아 보자.



The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for Problems, Javadoc, Declaration, Console, and Git Staging. The console output shows a Java application running on a Windows machine. The user input is "나 : 배고파" (I : I'm hungry). The AI response is "AI : 저는 먹을 것이 없어서 배고파요 . 당신은 무엇을 먹고 싶으세요 ? 제가 추천할 만한 음식이 있다면 알려주세요 !" (AI : I'm hungry because I have nothing to eat. What do you want to eat? If there is any food I can recommend, please let me know!).

```
<terminated> Main (6) [Java Application] C:\Users\zest1\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre
나 : 배고파
AI : 저는 먹을 것이 없어서 배고파요 . 당신은 무엇을 먹고 싶으세요 ? 제가 추천할 만한 음식이 있다면 알려주세요 !
```

9.지금까지는 한번 말을 건네면 한번 대답 오는게 끝 이였다. 대화가 계속 이어 질 수 있게 “대화종료” 라고 입력 하기 전까지 무한반복 하도록 하자.

- chatGPT는 http를 사용하여 통신한다. Http는 비연결성 통신이기 때문에 스트림을 한번 사용하면 재사용 할 수 없다. 따라서 한번 대화를 주고 받고 그 다음에는 또 다시 `connect()` 메서드를 사용해야 한다.

```
Console × Problems Debug Shell
<terminated> Main (6) [Java Application] C:\Users\Wzest1\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\bin\jav
나: 안녕
AI:하세요! 챗봇 AssistBot입니다. 무엇을 도와드릴까요? :)
나: 맘스터치 메뉴 추천해줘
AI:당근치즈버거, 화이트칼릭버거, 프라이드치킨, 맘스터치버거, 불고기버거 등이 맛있는 추천 메뉴입니다. 맘스터치의 유
나: 대화종료
```


10. 앞서 배운 것처럼 **Http** 통신은 비연결성이기에 **chatGPT**는 대화이력을 기억하고 있지 않다. 따라서 9번까지 해서 대화를 시도하면 대화 내용이 어색할 것이다.

그러므로 매번 새로운 채팅을 칠 때마다 이전에 주고 받은 대화 내용을 **json**형태로 모두 전송해줘야 한다. (끝말잇기를 시도해보면 된다)

- **chatGPT**의 응답을 저장해두고 다음 번 **send**시 대화 내역에 포함하여 전송해보자.
- 아래는 대화 내역을 전송하는 예시이다. **Messages** 키 안에 배열 형태로 대화를 추가해서 전송해줘야 한다.
- **JSONObject**, **JSONArray** 클래스를 활용해 보자.

객체화 하여 멤버변수로 가지고
있으면서 **messages** 에 추가로
계속 넣는다면?

```
# Note: you need to be using OpenAI Python v0.27.0 for the code below to work
import openai

openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Who won the world series in 2020?"},
        {"role": "assistant", "content": "The Los Angeles Dodgers won the World Series in 2020."},
        {"role": "user", "content": "Where was it played?"}
    ]
)
```

```
private JSONObject sendMsg = new JSONObject();
private HttpURLConnection con;

public ChatGPT(String key)
{
    this.key = key;

    sendMsg.put("model", "gpt-3.5-turbo");
    sendMsg.put("messages", new JSONArray());
}
```

```
Console X Problems Debug 5
Main (6) [Java Application] C:\Users\#zest1\#
나: 끝말잇기 하자
AI: 자전거
나: 거머리
AI: 리어카
나: 카센터
AI: 터널
나:
```


11. chatGPT 클래스의 `connect()` 메서드를 보면 연결을 위한 URL이 직접 적혀있다.
이렇게 되면 chatGPT 클래스를 사용하는 입장에서 URL 주소가 변경되면 사용 할 수가 없게 된다.
따라서 `connect()` 메서드의 매개변수로 받도록 하자.
- `connect()`의 시그니처가 변경되면서 기존의 코드들이 에러가 발생 할 것이다.
이를 해결할 방법은 무엇인가? chatGPT 클래스를 지금까지 사용해준 사용자들에게 모두 수정해달라고 부탁할것인가?

```
public HttpURLConnection connect()
{
    try
    {
        //URL주소를 객체화하는 클래스
        URL url = new URL("https://api.openai.com/v1/chat/completion

        //객체화한 url을 통해 http통신을 위한 객체를 생성한다. 메서드 방식은 POST 이다
        con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("POST");
        //http 통신시 데이터 형태는 json이라고 선언
        con.setRequestProperty("Content-Type", "application/json; utf-8"
        //발급받은 키값을 넣어준다.
```

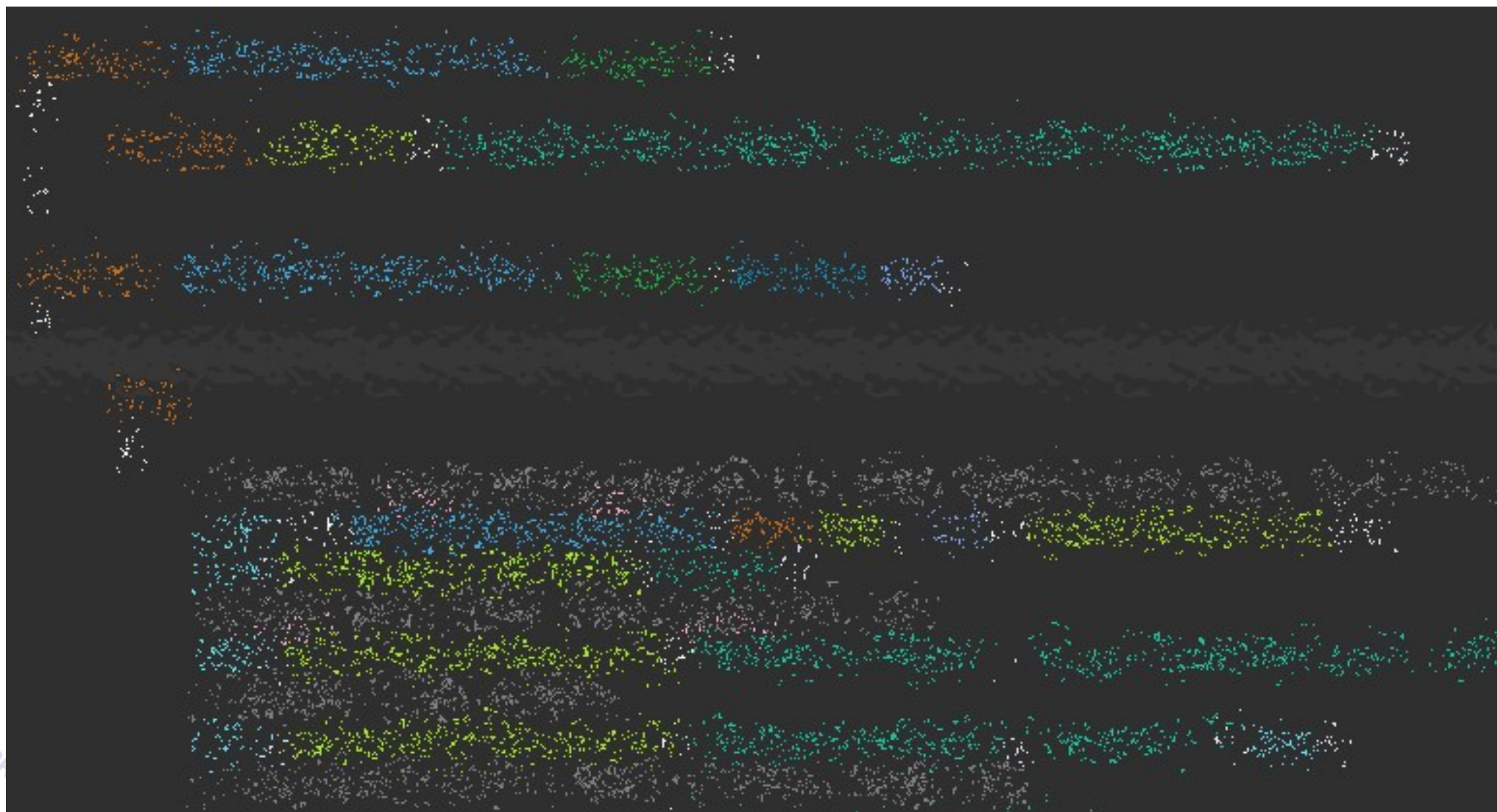
```
String key = "sk-sz1nhpR9mq3EHzioxtfT3BlbkFJfxtEw2Afwe3EBsBUXqKc";
ChatGPT gpt = new ChatGPT(key);
try
{
    while(true)
    {
        gpt.connect("https://api.openai.com/v1/chat/completions");

        Scanner scan = new Scanner(System.in);

        System.out.print("나: ");

        String msg = scan.nextLine();

        if(msg.equals("대화종료"))
```



12. chatGPT클래스를 사용하는 코드를 보면 키 값과 URL이 코드에 하드코딩 되어 있다.

만약 이것들이 변하면 코드를 다시 컴파일 해야 하기 때문에 불편하다.

외부 txt 파일에서 키 값과 url 값을 가져와서 사용하는 방식이면 재컴파일 없이 txt 파일만 수정하면 된다.

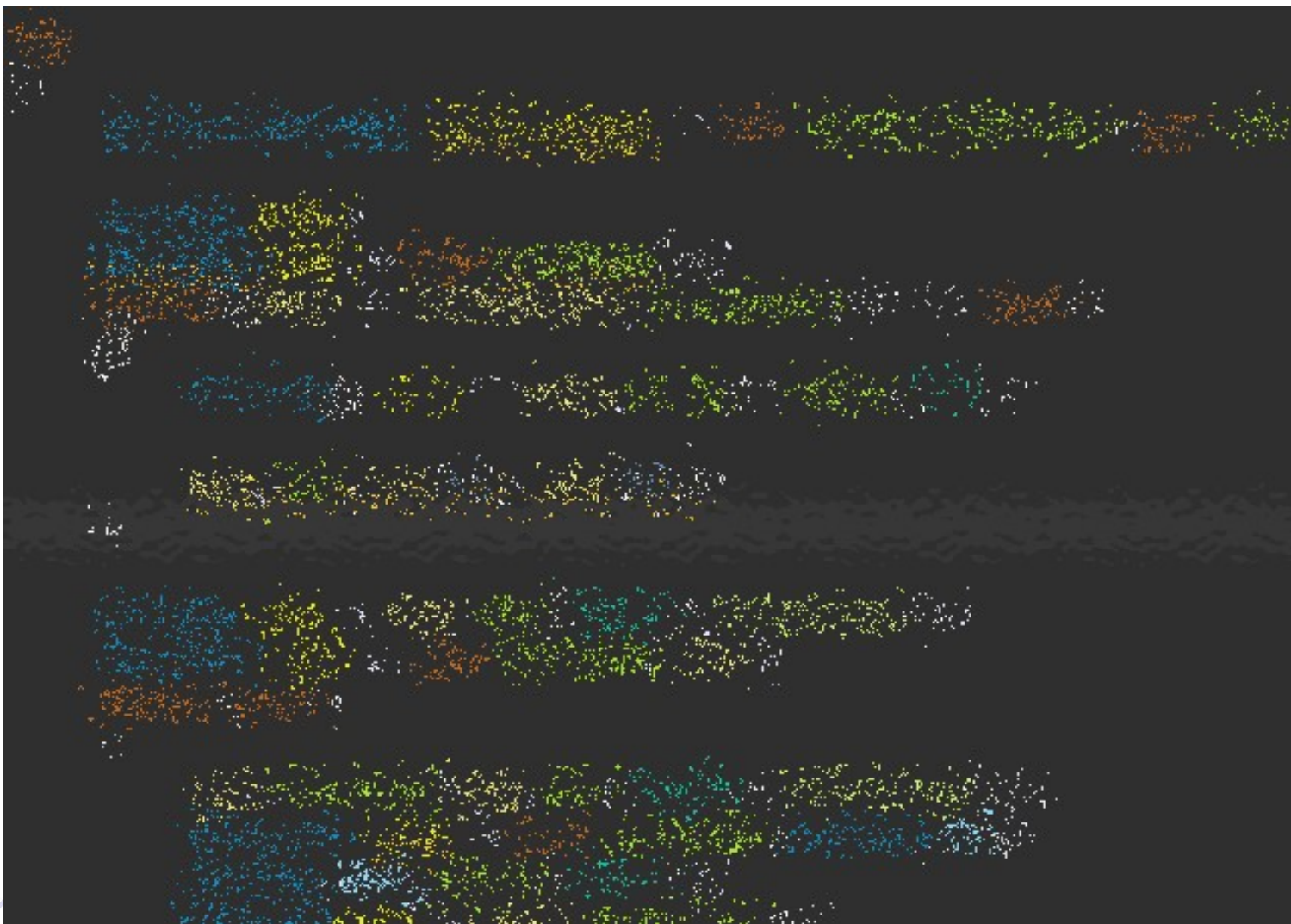
- FileReader 을 이용해서 키값과 url을 가져오자. (BufferedReader 의 Readline()을 활용하면 한줄씩 읽어오기에 편하다.)
- Map 자료구조를 이용하면 편하다. JsonObject 를 활용해도 좋다.

```
String key = "sk-sz1nhpR9mq3EHzioxtfT3BlbkFJfxtEw2Afwe3EBsBUXqKc";
ChatGPT gpt = new ChatGPT(key);
try
{
    while(true)
    {
        gpt.connect("https://api.openai.com/v1/chat/completions");
        Scanner scan = new Scanner(System.in);
    }
}
```

외부 txt파일로 분리

```
1 key=sk-sz1nhpR9mq3EHzioxtfT3BlbkFJfxtEw2Afwe3EBsBUXqKc
2 url=https://api.openai.com/v1/chat/completions
```





Park Ju Byeong

Park Ju Byeong



THANK YOU



강사 박주병