AI ASSISTED CODING

Lab 19 – Code Translation: Converting between programming languages

Roll no: 2503A51L44

Name: Meer Burhan Ali Hashmi

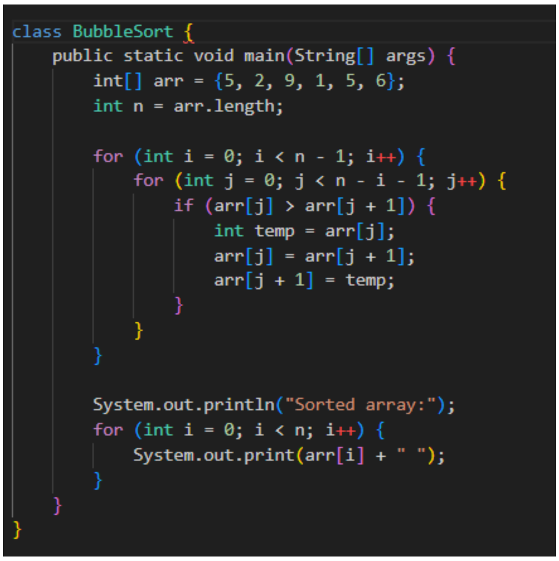
Batch:20

Lab Question 1: Sorting Algorithm Translation
You are part of a multinational development team. The backend is written in Java, but a new module requires a Python implementation of the same algorithm for integration with a data science pipeline.

Task 1: Use AI-assisted coding to translate a given Java bubble sort program into Python. Verify that the translated code works correctly.

Prompt: Translate the below java program into python code.

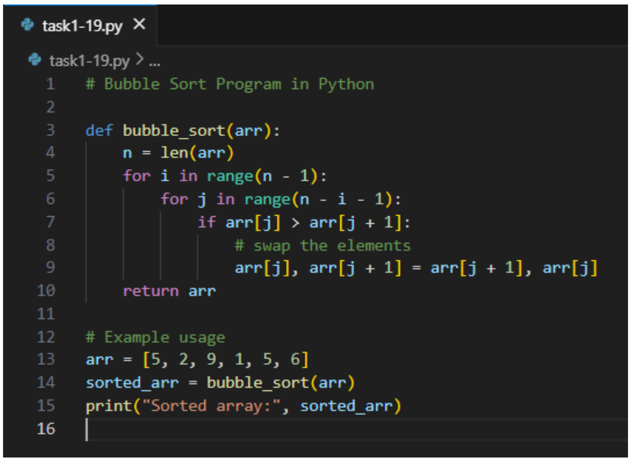
Original Java Code:



```java
class BubbleSort {
    public static void main(String[] args) {
        int[] arr = {5, 2, 9, 1, 5, 6};
        int n = arr.length;

        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }

        System.out.println("Sorted array:");
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

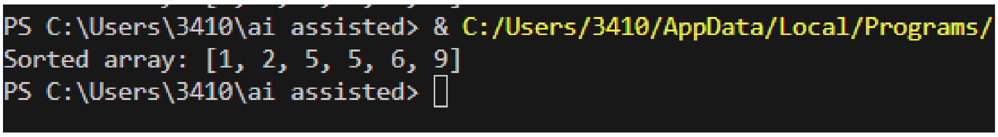
Code Generated:

(Translated Python Code)



```python
# Bubble Sort Program in Python

def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                # swap the elements
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

# Example usage
arr = [5, 2, 9, 1, 5, 6]
sorted_arr = bubble_sort(arr)
print("Sorted array:", sorted_arr)

```

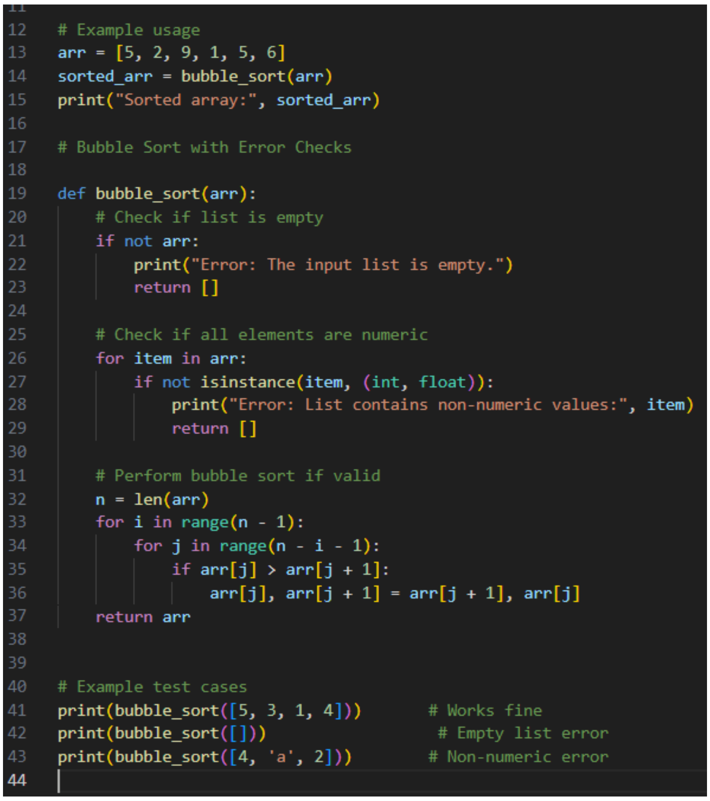
Output:



```
PS C:\Users\3410\ai assisted> & C:/Users/3410/AppData/Local/Programs/
Sorted array: [1, 2, 5, 5, 6, 9]
PS C:\Users\3410\ai assisted> 
```

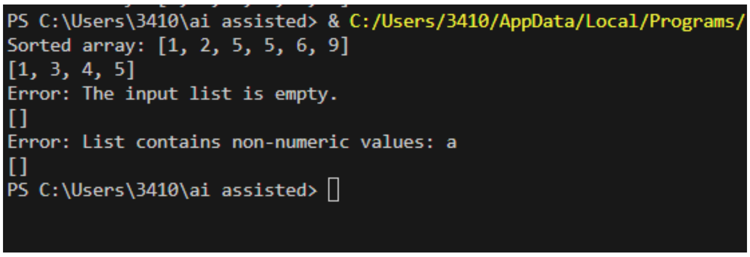
Task 2: Introduce errors in the Python version to check if the input list is empty or contains non-numeric values.

Code Generated:



```python
# Example usage
arr = [5, 2, 9, 1, 5, 6]
sorted_arr = bubble_sort(arr)
print("Sorted array:", sorted_arr)

# Bubble Sort with Error Checks

def bubble_sort(arr):
    # Check if list is empty
    if not arr:
        print("Error: The input list is empty.")
        return []

    # Check if all elements are numeric
    for item in arr:
        if not isinstance(item, (int, float)):
            print("Error: List contains non-numeric values:", item)
            return []

    # Perform bubble sort if valid
    n = len(arr)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr


# Example test cases
print(bubble_sort([5, 3, 1, 4]))        # Works fine
print(bubble_sort([]))                   # Empty list error
print(bubble_sort([4, 'a', 2]))         # Non-numeric error

```

Output:



```
PS C:\Users\3410\ai assisted> & C:/Users/3410/AppData/Local/Programs/
Sorted array: [1, 2, 5, 5, 6, 9]
[1, 3, 4, 5]
Error: The input list is empty.
[]
Error: List contains non-numeric values: a
[]
PS C:\Users\3410\ai assisted> 
```

Observation: The bubble sort algorithm was translated from Java to Python successfully, maintaining the same logic and output. Error-handling features were added in the Python version to check for empty or invalid input lists, ensuring the program runs smoothly and handles incorrect data safely.

Lab Question 2:File Handling Translation

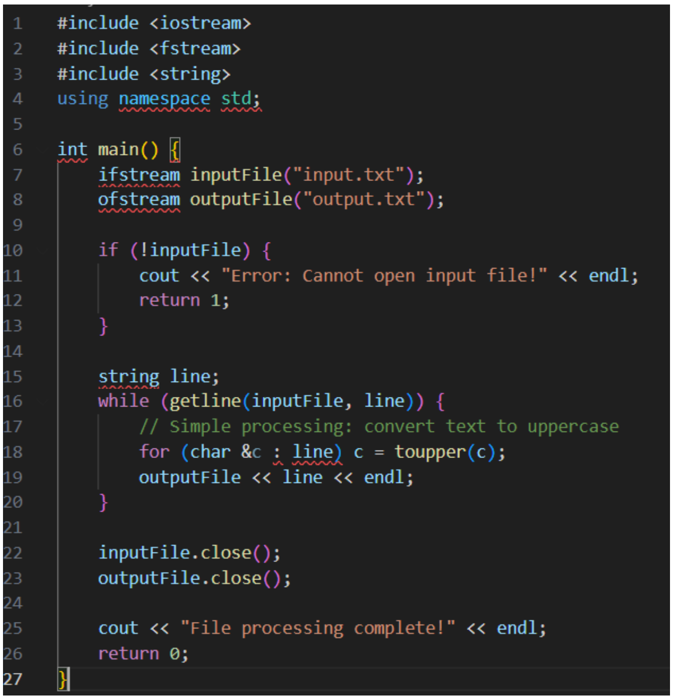
A company’s legacy codebases stores and processes files in C++, but the analytics team needs an equivalent program in JavaScript (Node.js) for integration with a web dashboard.

Task 1: Translate a given C++ file read-and-write program into JavaScript using AI assistance. Ensure the script reads a text file and writes processed output to a new file.

Prompt: Translate a **C++ file handling program** into **JavaScript (Node.js).**

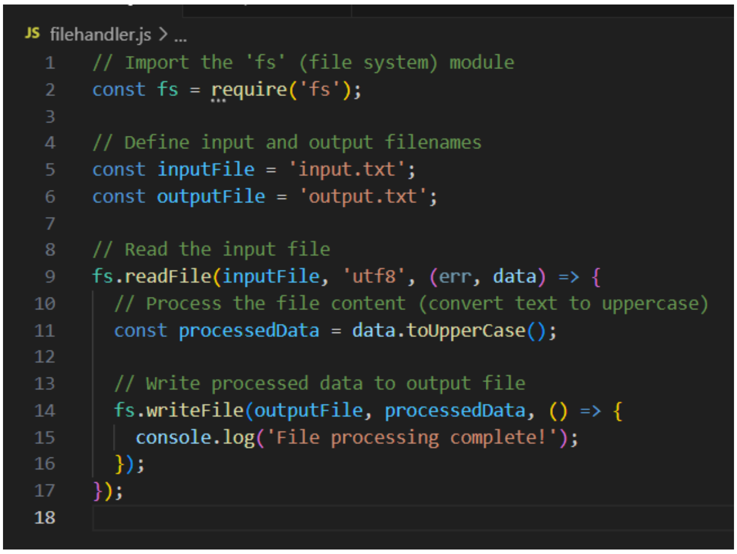
Code generated:

Original C++ program :



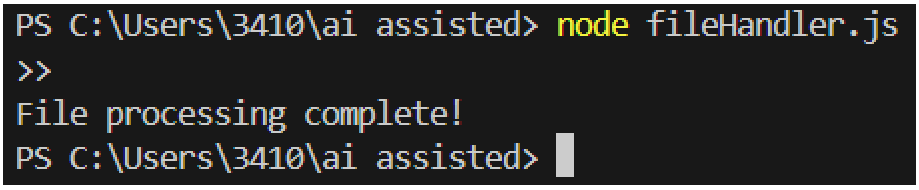
```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main() {
    ifstream inputFile("input.txt");
    ofstream outputFile("output.txt");

    if (!inputFile) {
        cout << "Error: Cannot open input file!" << endl;
        return 1;
    }

    string line;
    while (getline(inputFile, line)) {
        // Simple processing: convert text to uppercase
        for (char &c : line) c = toupper(c);
        outputFile << line << endl;
    }

    inputFile.close();
    outputFile.close();

    cout << "File processing complete!" << endl;
    return 0;
}
```

Equivalent JavaScript (Node.js) version :



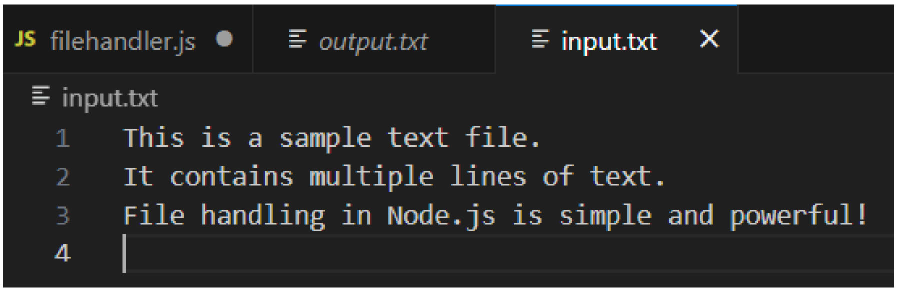
```javascript
// Import the 'fs' (file system) module
const fs = require('fs');

// Define input and output filenames
const inputFile = 'input.txt';
const outputFile = 'output.txt';

// Read the input file
fs.readFile(inputFile, 'utf8', (err, data) => {
  // Process the file content (convert text to uppercase)
  const processedData = data.toUpperCase();

  // Write processed data to output file
  fs.writeFile(outputFile, processedData, () => {
    console.log('File processing complete!');
  });
});
```

Output:



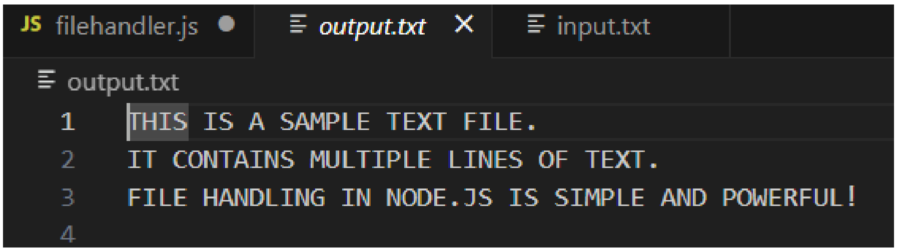
```
PS C:\Users\3410\ai assisted> node fileHandler.js
>>
File processing complete!
PS C:\Users\3410\ai assisted>
```

Input.txt file:



```
This is a sample text file.
It contains multiple lines of text.
File handling in Node.js is simple and powerful!

```

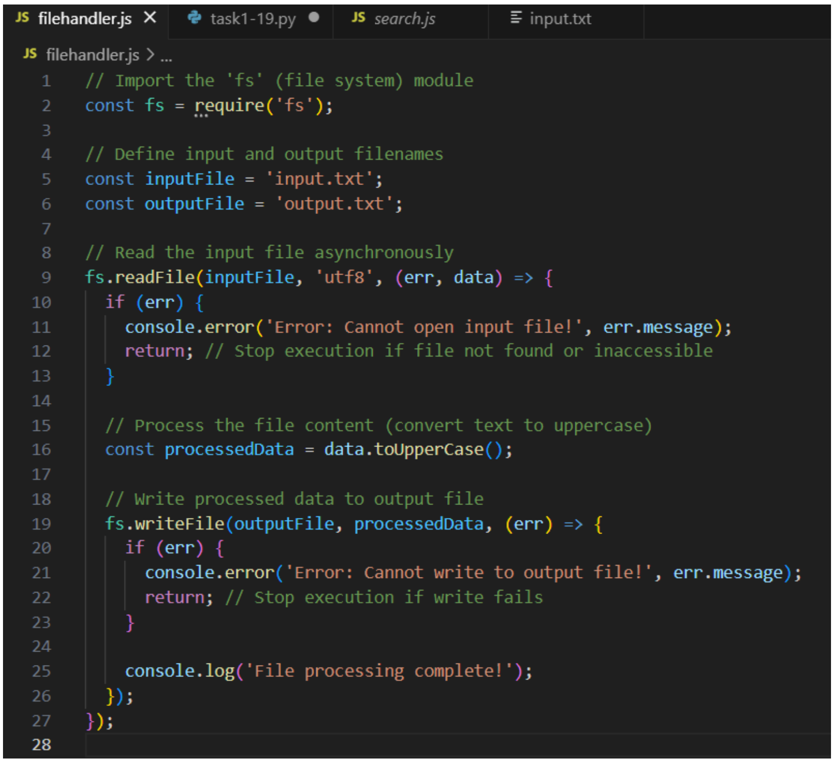
Created Output.txt file:



```
THIS IS A SAMPLE TEXT FILE.
IT CONTAINS MULTIPLE LINES OF TEXT.
FILE HANDLING IN NODE.JS IS SIMPLE AND POWERFUL!

```

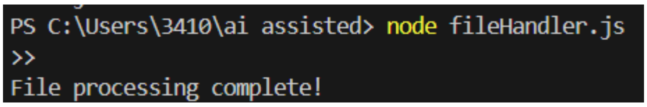
Task 2: Add error handling in the JavaScript version to gracefully handle missing files or permission errors.

Code Generated:



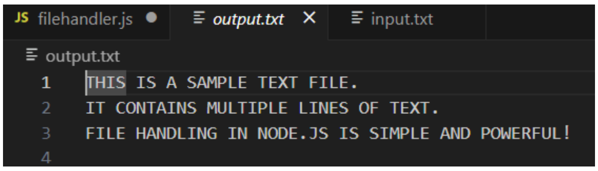
```javascript
// Import the 'fs' (file system) module
const fs = require('fs');

// Define input and output filenames
const inputFile = 'input.txt';
const outputFile = 'output.txt';

// Read the input file asynchronously
fs.readFile(inputFile, 'utf8', (err, data) => {
  if (err) {
    console.error('Error: Cannot open input file!', err.message);
    return; // Stop execution if file not found or inaccessible
  }

  // Process the file content (convert text to uppercase)
  const processedData = data.toUpperCase();

  // Write processed data to output file
  fs.writeFile(outputFile, processedData, (err) => {
    if (err) {
      console.error('Error: Cannot write to output file!', err.message);
      return; // Stop execution if write fails
    }

    console.log('File processing complete!');
  });
});

```

Normal case:



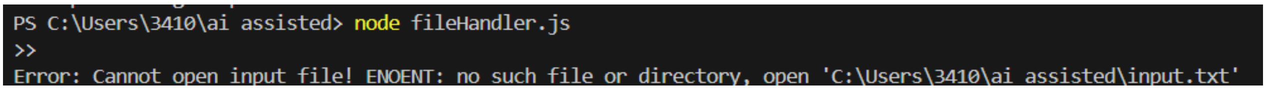
```
PS C:\Users\3410\ai assisted> node fileHandler.js
>>
File processing complete!
```

Creates output.txt



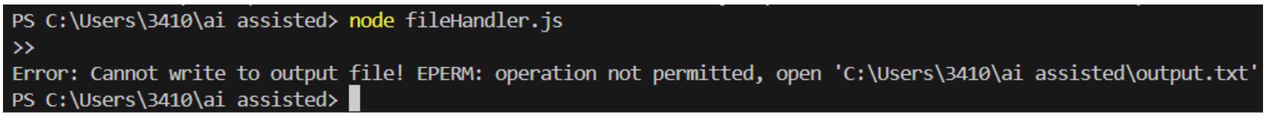
```
THIS IS A SAMPLE TEXT FILE.
IT CONTAINS MULTIPLE LINES OF TEXT.
FILE HANDLING IN NODE.JS IS SIMPLE AND POWERFUL!
```

(uppercase text):



```
PS C:\Users\3410\ai assisted> node fileHandler.js
>>
Error: Cannot open input file! ENOENT: no such file or directory, open 'C:\Users\3410\ai assisted\input.txt'
```

Missing input.txt:



```
PS C:\Users\3410\ai assisted> node fileHandler.js
>>
Error: Cannot write to output file! EPERM: operation not permitted, open 'C:\Users\3410\ai assisted\output.txt'
PS C:\Users\3410\ai assisted> 
```

No write permission:

Observation: The program reads data from an input file, processes it into uppercase, and writes the result to a new output file. It also includes error handling to manage missing files or permission issues gracefully without crashing.

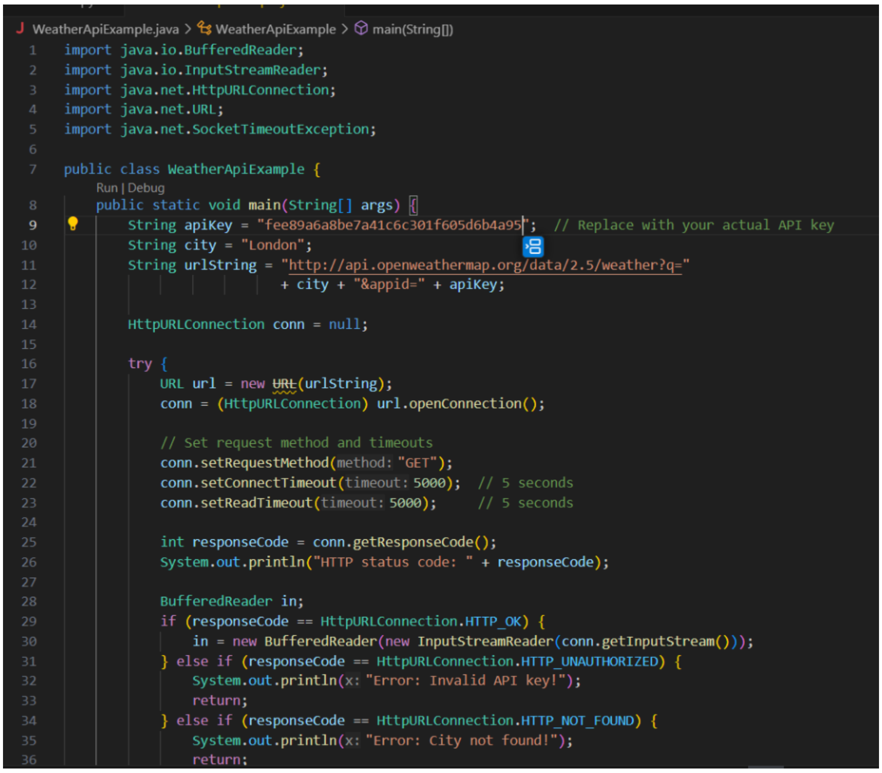
Lab Question 3: API Call Translation

Your team developed a prototype in Python to fetch weather data from an API,
but the production environment only supports Java.

Task 1: Translate the Python script (that makes an API call and prints the response) into Java using AI-assisted coding. Ensure equivalent functionality.

Prompt: Translate the below python script into java.

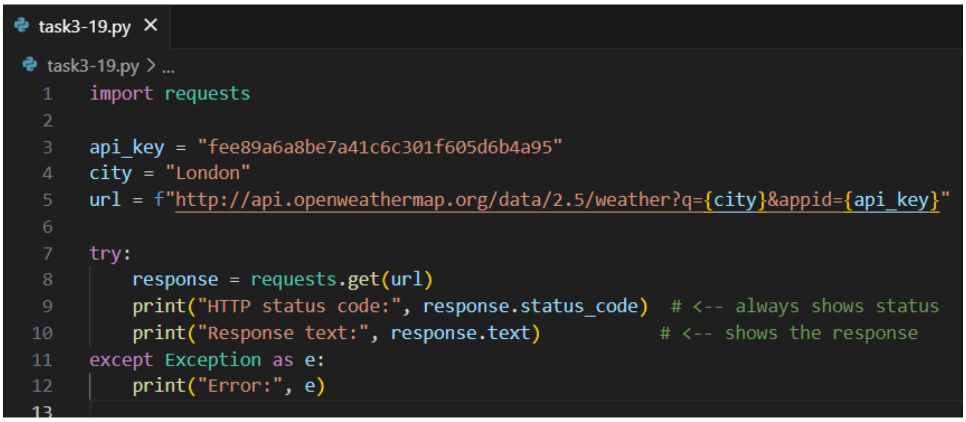
Original (Python) code:



WeatherApiExample.java > WeatherApiExample > main(String[])

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.SocketTimeoutException;

public class WeatherApiExample {
    Run | Debug
    public static void main(String[] args) {
        String apiKey = "fee89a6a8be7a41c6c301f605d6b4a95";  // Replace with your actual API key
        String city = "London";
        String urlString = "http://api.openweathermap.org/data/2.5/weather?q="
                           + city + "&appid=" + apiKey;

        HttpURLConnection conn = null;

        try {
            URL url = new URL(urlString);
            conn = (HttpURLConnection) url.openConnection();

            // Set request method and timeouts
            conn.setRequestMethod(method: "GET");
            conn.setConnectTimeout(timeout: 5000);  // 5 seconds
            conn.setReadTimeout(timeout: 5000);     // 5 seconds

            int responseCode = conn.getResponseCode();
            System.out.println("HTTP status code: " + responseCode);

            BufferedReader in;
            if (responseCode == HttpURLConnection.HTTP_OK) {
                in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
            } else if (responseCode == HttpURLConnection.HTTP_UNAUTHORIZED) {
                System.out.println(x: "Error: Invalid API key!");
                return;
            } else if (responseCode == HttpURLConnection.HTTP_NOT_FOUND) {
                System.out.println(x: "Error: City not found!");
                return;
```

Equivalent Java code:



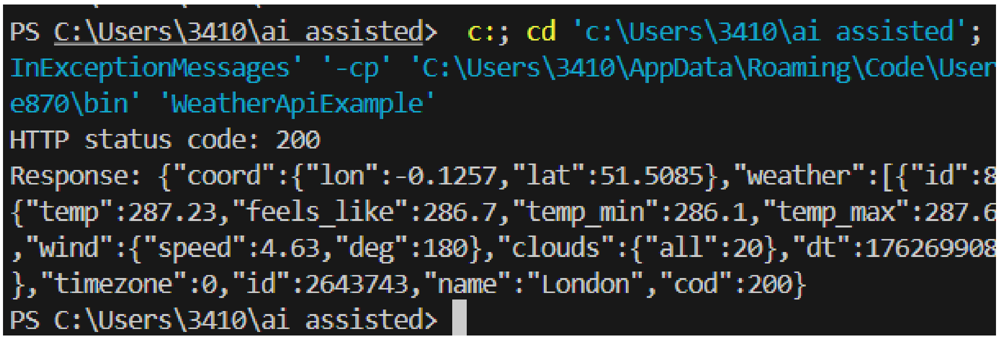
task3-19.py ×

task3-19.py > ...

```python
import requests

api_key = "fee89a6a8be7a41c6c301f605d6b4a95"
city = "London"
url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}"

try:
    response = requests.get(url)
    print("HTTP status code:", response.status_code)  # <-- always shows status
    print("Response text:", response.text)           # <-- shows the response
except Exception as e:
    print("Error:", e)
```

Output:

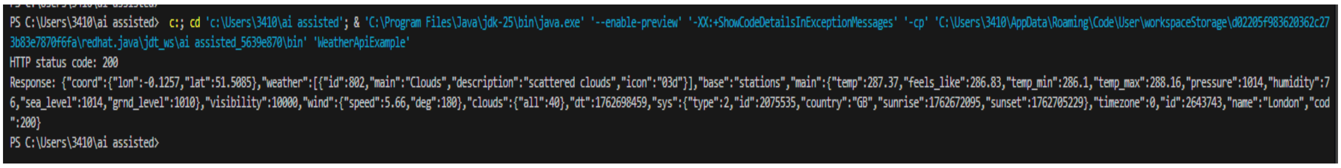
Task 2: Add proper error handling in the Java version for cases such as invalid API key, request timeout, no internet or connection.



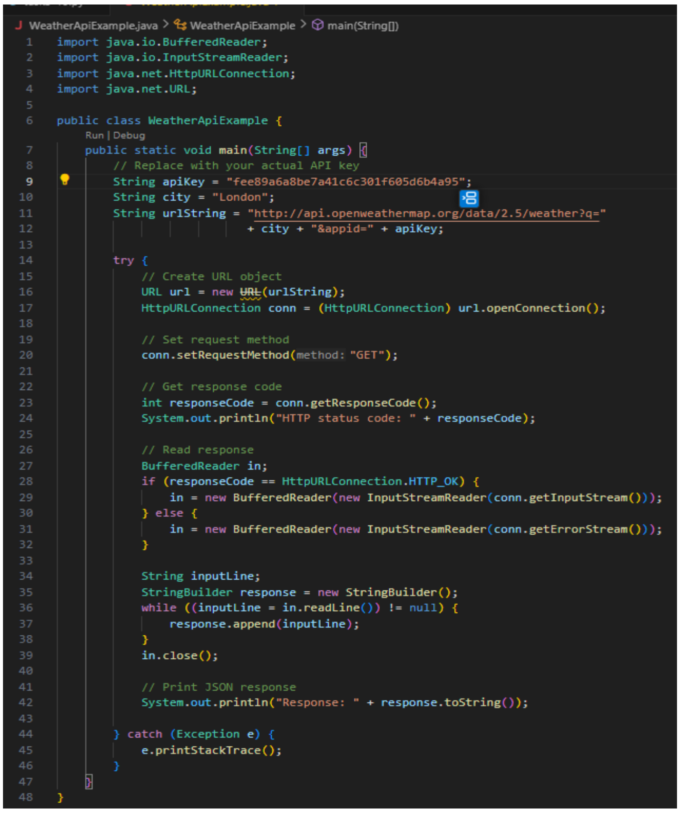
```
PS C:\Users\3410\ai assisted>  c:; cd 'c:\Users\3410\ai assisted';
InExceptionMessages' '-cp' 'C:\Users\3410\AppData\Roaming\Code\User
e870\bin' 'WeatherApiExample'
HTTP status code: 200
Response: {"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":8
{"temp":287.23,"feels_like":286.7,"temp_min":286.1,"temp_max":287.6
,"wind":{"speed":4.63,"deg":180},"clouds":{"all":20},"dt":176269908
},"timezone":0,"id":2643743,"name":"London","cod":200}
PS C:\Users\3410\ai assisted>
```

Code Generated:

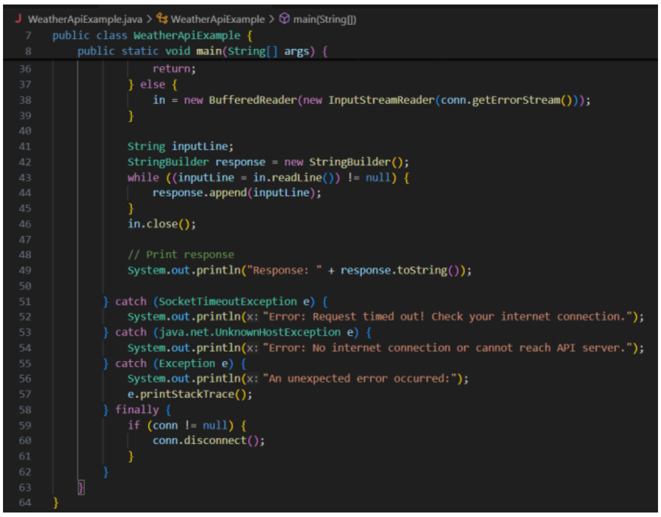
Successful API Call :



```
PS C:\Users\3410\ai assisted>  c:; cd 'c:\Users\3410\ai assisted'; & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\3410\AppData\Roaming\Code\User\workspaceStorage\d02205f983620362c27
3b83e7870f6fa\redhat.java\jdt_ws\ai assisted_5639e870\bin' 'WeatherApiExample'
HTTP status code: 200
Response: {"coord":{"lon":-0.1257,"lat":51.5085},"weather":[{"id":802,"main":"Clouds","description":"scattered clouds","icon":"03d"}],"base":"stations","main":{"temp":287.37,"feels_like":286.83,"temp_min":286.1,"temp_max":288.16,"pressure":1014,"humidity":7
6,"sea_level":1014,"grnd_level":1010},"visibility":10000,"wind":{"speed":5.66,"deg":180},"clouds":{"all":40},"dt":1762698459,"sys":{"type":2,"id":2075535,"country":"GB","sunrise":1762672095,"sunset":1762705229},"timezone":0,"id":2643743,"name":"London","cod
":200}
PS C:\Users\3410\ai assisted>
```



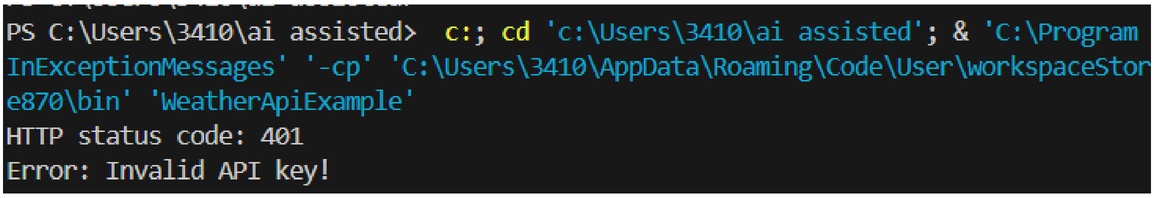
```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class WeatherApiExample {
    public static void main(String[] args) {
        // Replace with your actual API key
        String apiKey = "fee89a6a8be7a41c6c301f605d6b4a95";
        String city = "London";
        String urlString = "http://api.openweathermap.org/data/2.5/weather?q="
                         + city + "&appid=" + apiKey;

        try {
            // Create URL object
            URL url = new URL(urlString);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();

            // Set request method
            conn.setRequestMethod(method: "GET");

            // Get response code
            int responseCode = conn.getResponseCode();
            System.out.println("HTTP status code: " + responseCode);

            // Read response
            BufferedReader in;
            if (responseCode == HttpURLConnection.HTTP_OK) {
                in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
            } else {
                in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
            }

            String inputLine;
            StringBuilder response = new StringBuilder();
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            // Print JSON response
            System.out.println("Response: " + response.toString());

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```



```java
public class WeatherApiExample {
    public static void main(String[] args) {
                    return;
                } else {
                    in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
                }

                String inputLine;
                StringBuilder response = new StringBuilder();
                while ((inputLine = in.readLine()) != null) {
                    response.append(inputLine);
                }
                in.close();

                // Print response
                System.out.println("Response: " + response.toString());

            } catch (SocketTimeoutException e) {
                System.out.println(x: "Error: Request timed out! Check your internet connection.");
            } catch (java.net.UnknownHostException e) {
                System.out.println(x: "Error: No internet connection or cannot reach API server.");
            } catch (Exception e) {
                System.out.println(x: "An unexpected error occurred:");
                e.printStackTrace();
            } finally {
                if (conn != null) {
                    conn.disconnect();
                }
            }
        }
}
```

Invalid API Key:

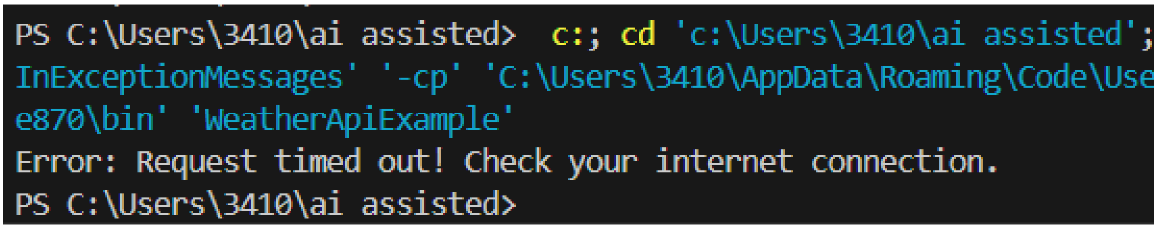
Request Timeout :



```
PS C:\Users\3410\ai assisted>  c:; cd 'c:\Users\3410\ai assisted'; & 'C:\Program
InExceptionMessages' '-cp' 'C:\Users\3410\AppData\Roaming\Code\User\workspaceStor
e870\bin' 'WeatherApiExample'
HTTP status code: 401
Error: Invalid API key!
```

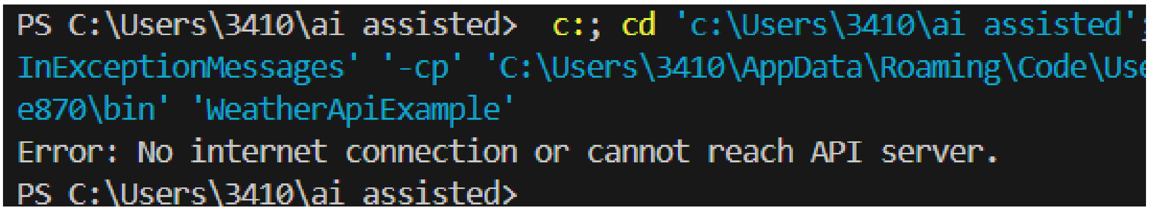
No

Internet / Cannot Reach API Server :



```
PS C:\Users\3410\ai assisted>  c:; cd 'c:\Users\3410\ai assisted';
InExceptionMessages' '-cp' 'C:\Users\3410\AppData\Roaming\Code\Use
e870\bin' 'WeatherApiExample'
Error: Request timed out! Check your internet connection.
PS C:\Users\3410\ai assisted>
```

Observation: The Java program successfully fetches weather data from the API, and with added error handling, it gracefully handles invalid API keys, wrong cities, timeouts, or no internet, giving clear and safe output.



```
PS C:\Users\3410\ai assisted>  c:; cd 'c:\Users\3410\ai assisted';
InExceptionMessages' '-cp' 'C:\Users\3410\AppData\Roaming\Code\Use
e870\bin' 'WeatherApiExample'
Error: No internet connection or cannot reach API server.
PS C:\Users\3410\ai assisted>
```