

AI ASSISTED CODING

Lab 19 – Code Translation: Converting between programming languages

Name: G Hasini

Roll no: 2503A51L41

Batch: 24BTCAICSB20

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:

```
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)

```
task1-19.py X  
task1-19.py > ...  
1  # Bubble Sort Program in Python  
2  
3  def bubble_sort(arr):  
4      n = len(arr)  
5      for i in range(n - 1):  
6          for j in range(n - i - 1):  
7              if arr[j] > arr[j + 1]:  
8                  # swap the elements  
9                  arr[j], arr[j + 1] = arr[j + 1], arr[j]  
10     return arr  
11  
12  # Example usage  
13 arr = [5, 2, 9, 1, 5, 6]  
14 sorted_arr = bubble_sort(arr)  
15 print("Sorted array:", sorted_arr)  
16 |
```

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:

```
11  # Example usage
12 arr = [5, 2, 9, 1, 5, 6]
13 sorted_arr = bubble_sort(arr)
14 print("Sorted array:", sorted_arr)
15
16
17 # Bubble Sort with Error Checks
18
19 def bubble_sort(arr):
20     # Check if list is empty
21     if not arr:
22         print("Error: The input list is empty.")
23         return []
24
25     # Check if all elements are numeric
26     for item in arr:
27         if not isinstance(item, (int, float)):
28             print("Error: List contains non-numeric values:", item)
29             return []
30
31     # Perform bubble sort if valid
32     n = len(arr)
33     for i in range(n - 1):
34         for j in range(n - i - 1):
35             if arr[j] > arr[j + 1]:
36                 arr[j], arr[j + 1] = arr[j + 1], arr[j]
37
38     return arr
39
40 # Example test cases
41 print(bubble_sort([5, 3, 1, 4]))      # Works fine
42 print(bubble_sort([]))                # Empty list error
43 print(bubble_sort([4, 'a', 2]))       # Non-numeric error
44 |
```

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 :

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 using namespace std;
5
6 int main() {
7     ifstream inputFile("input.txt");
8     ofstream outputFile("output.txt");
9
10    if (!inputFile) {
11        cout << "Error: Cannot open input file!" << endl;
12        return 1;
13    }
14
15    string line;
16    while (getline(inputFile, line)) {
17        // Simple processing: convert text to uppercase
18        for (char &c : line) c = toupper(c);
19        outputFile << line << endl;
20    }
21
22    inputFile.close();
23    outputFile.close();
24
25    cout << "File processing complete!" << endl;
26    return 0;
27 }
```

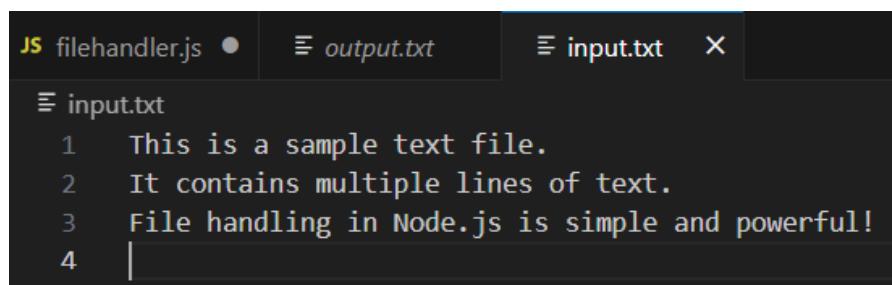
Equivalent JavaScript (Node.js) version :

```
JS filehandler.js > ...
1 // Import the 'fs' (file system) module
2 const fs = require('fs');
3
4 // Define input and output filenames
5 const inputFile = 'input.txt';
6 const outputFile = 'output.txt';
7
8 // Read the input file
9 fs.readFile(inputFile, 'utf8', (err, data) => {
10    // Process the file content (convert text to uppercase)
11    const processedData = data.toUpperCase();
12
13    // Write processed data to output file
14    fs.writeFile(outputFile, processedData, () => {
15      console.log('File processing complete!');
16    });
17 });
18
```

Output:

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

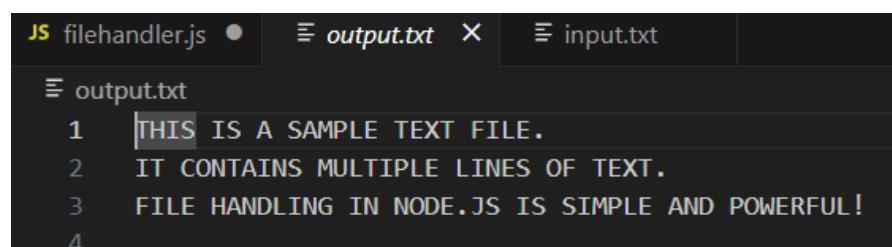
Input.txt file:



The screenshot shows a terminal window with three tabs at the top: 'filehandler.js' (selected), 'output.txt', and 'input.txt'. The 'input.txt' tab is active, displaying the following text:

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

Created Output.txt file:

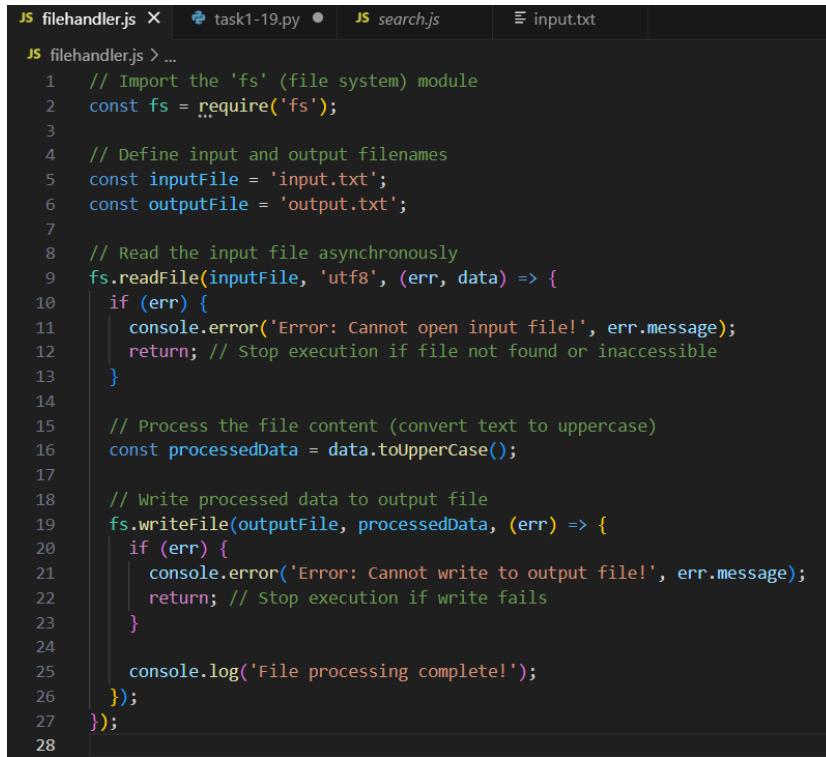


The screenshot shows a terminal window with three tabs at the top: 'filehandler.js' (selected), 'output.txt', and 'input.txt'. The 'output.txt' tab is active, displaying the following text, which is the uppercase version of the input file:

```
1 THIS IS A SAMPLE TEXT FILE.
2 IT CONTAINS MULTIPLE LINES OF TEXT.
3 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:

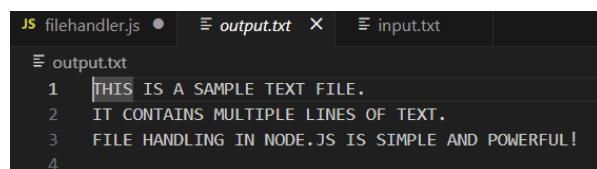


```
JS filehandler.js X task1-19.py ● JS search.js input.txt
JS filehandler.js > ...
1 // Import the 'fs' (file system) module
2 const fs = require('fs');
3
4 // Define input and output filenames
5 const inputFile = 'input.txt';
6 const outputFile = 'output.txt';
7
8 // Read the input file asynchronously
9 fs.readFile(inputFile, 'utf8', (err, data) => {
10   if (err) {
11     console.error('Error: Cannot open input file!', err.message);
12     return; // Stop execution if file not found or inaccessible
13   }
14
15   // Process the file content (convert text to uppercase)
16   const processedData = data.toUpperCase();
17
18   // Write processed data to output file
19   fs.writeFile(outputFile, processedData, (err) => {
20     if (err) {
21       console.error('Error: Cannot write to output file!', err.message);
22       return; // Stop execution if write fails
23     }
24
25     console.log('File processing complete!');
26   });
27 });
28 }
```

Normal case:

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

Creates output.txt (uppercase text):



```
JS filehandler.js ● output.txt X input.txt
JS output.txt
1 THIS IS A SAMPLE TEXT FILE.
2 IT CONTAINS MULTIPLE LINES OF TEXT.
3 FILE HANDLING IN NODE.JS IS SIMPLE AND POWERFUL!
4
```

Missing input.txt:

```
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'
```

No write permission:

```
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>
```

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:

```
task3-19.py x
task3-19.py > ...
1 import requests
2
3 api_key = "fee89a6a8be7a41c6c301f605d6b4a95"
4 city = "London"
5 url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}"
6
7 try:
8     response = requests.get(url)
9     print("HTTP status code:", response.status_code) # <-- always shows status
10    print("Response text:", response.text)           # <-- shows the response
11 except Exception as e:
12     print("Error:", e)
13
```

Equivalent Java code:

```
WeatherApiExample.java > WeatherApiExample > main(String[])
1 import java.io.BufferedReader;
2 import java.io.InputStreamReader;
3 import java.net.HttpURLConnection;
4 import java.net.URL;
5
6 public class WeatherApiExample {
7     Run|Debug
8     public static void main(String[] args) {
9         // Replace with your actual API key
10        String apiKey = "fee89a6a8be7a41c6c301f605d6b4a95";
11        String city = "London";
12        String urlString = "http://api.openweathermap.org/data/2.5/weather?q="
13                           + city + "&appid=" + apiKey;
14
15        try {
16            // Create URL object
17            URL url = new URL(urlString);
18            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
19
20            // Set request method
21            conn.setRequestMethod(method: "GET");
22
23            // Get response code
24            int responseCode = conn.getResponseCode();
25            System.out.println("HTTP status code: " + responseCode);
26
27            // Read response
28            BufferedReader in;
29            if (responseCode == HttpURLConnection.HTTP_OK) {
30                in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
31            } else {
32                in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
33            }
34
35            String inputLine;
36            StringBuilder response = new StringBuilder();
37            while ((inputLine = in.readLine()) != null) {
38                response.append(inputLine);
39            }
40            in.close();
41
42            // Print JSON response
43            System.out.println("Response: " + response.toString());
44
45        } catch (Exception e) {
46            e.printStackTrace();
47        }
48    }
}
```

Output:

```
PS C:\Users\3410\ai assisted> 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\d02205f98362036c273e83e70f6fa\weather.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":76,"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>
```

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

Code Generated:

```
J WeatherApiExample.java > WeatherApiExample > main(String[])
1 import java.io.BufferedReader;
2 import java.io.InputStreamReader;
3 import java.net.HttpURLConnection;
4 import java.net.URL;
5 import java.net.SocketTimeoutException;
6
7 public class WeatherApiExample {
8     Run|Debug
9     public static void main(String[] args) {
10         String apiKey = "fee89aaabbe7a41c6c301f605d6b4a95"; // Replace with your actual API key
11         String city = "London";
12         String urlString = "http://api.openweathermap.org/data/2.5/weather?q="
13             + city + "&appid=" + apiKey;
14
15         HttpURLConnection conn = null;
16
17         try {
18             URL url = new URL(urlString);
19             conn = (HttpURLConnection) url.openConnection();
20
21             // Set request method and timeouts
22             conn.setRequestMethod(method: "GET");
23             conn.setConnectTimeout(timeout: 5000); // 5 seconds
24             conn.setReadTimeout(timeout: 5000); // 5 seconds
25
26             int responseCode = conn.getResponseCode();
27             System.out.println("HTTP status code: " + responseCode);
28
29             BufferedReader in;
30             if (responseCode == HttpURLConnection.HTTP_OK) {
31                 in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
32             } else if (responseCode == HttpURLConnection.HTTP_UNAUTHORIZED) {
33                 System.out.println(x: "Error: Invalid API key!");
34                 return;
35             } else if (responseCode == HttpURLConnection.HTTP_NOT_FOUND) {
36                 System.out.println(x: "Error: City not found!");
37                 return;
38             }
39
40             String inputLine;
41             StringBuilder response = new StringBuilder();
42             while ((inputLine = in.readLine()) != null) {
43                 response.append(inputLine);
44             }
45             in.close();
46
47             // Print response
48             System.out.println("Response: " + response.toString());
49
50         } catch (SocketTimeoutException e) {
51             System.out.println(x: "Error: Request timed out! Check your internet connection.");
52         } catch (java.net.UnknownHostException e) {
53             System.out.println(x: "Error: No internet connection or cannot reach API server.");
54         } catch (Exception e) {
55             System.out.println(x: "An unexpected error occurred:");
56             e.printStackTrace();
57         } finally {
58             if (conn != null) {
59                 conn.disconnect();
60             }
61         }
62     }
63 }
64 }
```

```
J WeatherApiExample.java > WeatherApiExample > main(String[])
7 public class WeatherApiExample {
8     public static void main(String[] args) {
36         return;
37     } else {
38         in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
39     }
40
41     String inputLine;
42     StringBuilder response = new StringBuilder();
43     while ((inputLine = in.readLine()) != null) {
44         response.append(inputLine);
45     }
46     in.close();
47
48     // Print response
49     System.out.println("Response: " + response.toString());
50
51 } catch (SocketTimeoutException e) {
52     System.out.println(x: "Error: Request timed out! Check your internet connection.");
53 } catch (java.net.UnknownHostException e) {
54     System.out.println(x: "Error: No internet connection or cannot reach API server.");
55 } catch (Exception e) {
56     System.out.println(x: "An unexpected error occurred:");
57     e.printStackTrace();
58 } finally {
59     if (conn != null) {
60         conn.disconnect();
61     }
62 }
63 }
64 }
```

Successful API Call :

```
PS C:\Users\3410\ai assisted> c:; cd 'c:\Users\3410\ai assisted'; InExceptionMessages' '-cp' 'C:\Users\3410\AppData\Roaming\Code\User\870\bin' 'WeatherApiExample'
HTTP status code: 200
Response: {"coord":{"lon":-0.1257,"lat":51.5085}, "weather": [{"id":800,"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>
```

Invalid API Key:

```
PS C:\Users\3410\ai assisted> c:; cd 'c:\Users\3410\ai assisted'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java' 'InExceptionMessages' '-cp' 'C:\Users\3410\AppData\Roaming\Code\User\870\bin' 'WeatherApiExample'
HTTP status code: 401
Error: Invalid API key!
```

Request Timeout :

```
PS C:\Users\3410\ai assisted> c:; cd 'c:\Users\3410\ai assisted'; InExceptionMessages' '-cp' 'C:\Users\3410\AppData\Roaming\Code\User\870\bin' 'WeatherApiExample'
Error: Request timed out! Check your internet connection.
PS C:\Users\3410\ai assisted>
```

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\User\870\bin' 'WeatherApiExample'
Error: No internet connection or cannot reach API server.
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.