Error-Resilient Python to Java Transpiler
User Manual

A. Introduction:

The **Error-Resilient Python to Java Transpiler** is designed to automate this process while intelligently identifying and correcting common errors. This tool analyzes Python code line by line, detects mistakes, and applies predefined correction strategies to ensure accurate conversion into Java. It can handle missing semicolons, incorrect function names, structural differences, and more. Additionally, it generates a detailed **error log**, documenting every correction with explanations, providing transparency in the conversion process.

By utilizing this transpiler, developers can quickly obtain Java code that adheres to correct syntax, significantly reducing debugging time and improving code reliability. Whether you're a beginner transitioning between languages or an experienced developer streamlining workflow, this tool simplifies Python-to-Java conversion while preserving code integrity.

B. Features:

- **Automatic Syntax Error Recovery**: Fixes missing colons, incorrect indentation, and misplaced parentheses.
- **Accurate Code Generation**: Ensures Java output maintains the logic of the original Python code.
- **Error Logging**: Records each detected issue and its correction in error_log.txt.
- **Dark-Themed UI**: A visually appealing, easy-to-read interface.

C. System Requirements:

- **Operating System**: Windows, Linux, macOS
- **Java Version**: JDK 11 or higher

D. Installation Guide:

Follow the steps below to install and run the **Error-Resilient Python to Java Transpiler**:

Step 1: Clone the Repository

Open your terminal or command prompt and run the following command to clone the repository from GitHub:

git clone https://github.com/AkshatPandey-2004/Error-Resilient-Transpiler.git

Step 2: Navigate to the Project Folder

Once the cloning process is complete, navigate into the project directory:

cd Error-Resilient-Transpiler

Step 3: Open the Project in a Code Editor

You can open the folder in your preferred code editor, such as VS Code:

code .

Step 4: Compile the Java Files

Before running the transpiler, compile all Java files using the following command:

javac *.java

Step 5: Run the Compiler UI

After successful compilation, start the transpiler's user interface by running:

java CompilerUI

Now, you can start converting Python code into Java using the **Error-Resilient Transpiler**!

E. User Interface Overview

The **Error-Resilient Python to Java Transpiler** user interface consists of three primary panels:

1. Python Code Panel (Input Section)

- This panel is located on the **left side** of the UI.
- It contains a **text area** where users can write or paste Python code that needs to be converted.
- The text area has a dark-themed background with white-colored text for easy readability.
- A **label ("Input Python Code")** is displayed at the top of this panel.
- Users can type their Python scripts here before running the conversion process.

2. Java Code Panel (Output Section)

- This panel is positioned on the **right side**, adjacent to the Python Code Panel.
- It displays the **converted Java code** after processing the input Python code.
- The text area is styled similarly to the Python Code Panel but uses **green text** to differentiate the output.
- The panel is **non-editable**, ensuring users can view the transpiler's generated Java code without modification.
- A **label ("Converted Java Code")** is shown at the top of this panel.

3. Error Log Panel

- This panel is located on the **far right** of the interface.
- It displays a **log of detected errors and corrections** made during the Python-to-Java conversion.
- The text is highlighted in **red** to make it easy for users to identify the mistakes that were fixed.
- A **label ("Error Log")** is displayed at the top of this panel.
- The error log helps developers understand what modifications were made to ensure correct Java syntax.

Buttons in the UI

Below the header, there are **three main buttons** to control the transpiler:

1) **Convert Button**
   a) Triggers the transpiler to process the Python code and generate Java code.
   b) Updates the **Java Code Panel** with the converted output.
   c) Logs any syntax fixes in the **Error Log Panel**.
2) **Manual Button**
   a) Opens the **User Manual** providing instructions on using the transpiler.
   b) Helps users understand the tool's functionality and features.
3) **Code Button**
   a) Redirects users to the **GitHub repository** of the project.
   b) Allows developers to access the source code, contribute, or report issues.

F. Usage Instructions

Step 1: Enter Python Code

Type or paste Python code into the **Input Python Code Area**.

Step 2: Convert Code

Click the **Convert** button to start the translation process.

Step 3: Review Java Output

The **Converted Java Code Area** displays the translated Java code.

Step 4: Check Error Log (If Needed)

If errors were detected and fixed, they will appear in the **Error Log Panel** and error_log.txt.

Step 5: Copy or Save Java Code

Copy the generated Java code and use it in your project.

G. Sample Test Code

Below are examples demonstrating how the **Error-Resilient Python to Java Transpiler** works.

a) Standard Test Case (Correct Python Code)

This example shows how correctly written Python code is converted into Java without any modifications.

Input (Python Code):

```
a=10
b=9
if a<b:
  print(b)
else:
  print(a)
```

Output (Converted Java Code):

```
import java.util.*;
public class output {
   public static void main(String[] args) {
      int a = 10;
      int b = 9;
      if (a < b) {
            System.out.println(b);
      } else {
            System.out.println(a);
      }}}
```

b) Error-Resilient Test Case (Python Code with Errors)

This example demonstrates how the transpiler fixes common errors like **typos, missing colons, and incorrect indentation**.

Input (Python Code with Errors):

```
def hello():
   a=10
   b="hello"
   c 9.8
   if a>20
     print("bigger"
```

Output (Fixed and Converted Java Code):

```
import java.util.*;

public class output {

   public static void main(String[] args) {

   public static void hello() {

            int a = 10;

            String b = "hello";

            double c = 9.8;

            if (a > 20) {

               System.out.println("bigger");

      } }}}
```

Error Log Generated:

```
------ Error Entry ------

Timestamp: Wed Apr XX XX:XX:XX IST 20XX

Syntax Error: Expected SYMBOL, but found NUMBER (9.8)

-------------------------

------ Error Entry ------

Timestamp: Wed Apr XX XX:XX:XX IST 20XX

Syntax Error: Expected SYMBOL, but found KEYWORD (print)

-------------------------
```

H. Limitations

While the **Error-Resilient Python to Java Transpiler** efficiently converts Python code to Java and corrects common syntax errors, there are certain limitations:

1. Indentation Issues

Python relies on **indentation** to define blocks of code, whereas Java uses **curly braces** {}. The transpiler attempts to infer the correct structure, but in cases where indentation is inconsistent, it may produce incorrect Java code.

2. Decremented Loop Conversion

In Python, loops can run in reverse using range(start, stop, step), where step is negative. Java requires explicit decrementing logic, and the transpiler may not always correctly infer complex step values.

3. Does Not Support Python Libraries

The transpiler is designed to convert **core Python syntax** to Java but **does not support Python libraries** such as NumPy, Pandas, Matplotlib, or custom module imports.

**** There might be more limitations that arise in edge cases or complex Python scripts. Users should review the generated Java code and make necessary modifications.**

I. Contact the Developer

If you have any questions, suggestions, or encounter issues with the **Error-Resilient Python to Java Transpiler**, feel free to reach out to the developer.

Email: akshatpandey26.2004@gmail.com

GitHub Repository: https://github.com/AkshatPandey-2004/Error-Resilient-Transpiler

LinkedIn: Akshat Pandey