Go Code:

The code defines an HTTP server that serves JSON data related to wafer information. It listens to incoming requests and responds accordingly. The server exposes a single endpoint, "/wafers", which returns JSON data related to a wafer. The returned wafer data has four fields: Etime, Stime, Diameter, and Numchip.

The following packages are imported:

encoding/json: It's used to marshal and unmarshal JSON data.

fmt: It provides formatted I/O operations similar to C's printf and scanf.

math/rand: It's used to generate random numbers.

net/http: It's used to create an HTTP server.

time: It provides time-related functions.

Wafer Struct

The Wafer struct defines the structure of a wafer. It has four fields:

Etime: It's a string field that represents the end time of the wafer.

Stime: It's a string field that represents the start time of the wafer.

Diameter: It's an integer field that represents the diameter of the wafer.

Numchip: It's a float field that represents the number of chips on the wafer.

Global Variables

The global variable is defined as an integer, and w2 is defined as a Wafer instance.

Main Function

The main function is the entry point for the program. It starts by setting the value of global to 0.Next, it sets up an HTTP server by calling the http.HandleFunc function twice. The first call sets up a handler function for the "/wafers" endpoint, and the second call sets up a handler function for the root endpoint.

After setting up the server, it starts listening for incoming requests on port 4445.

Handler Functions

Index Function

The index function is a handler function for the root endpoint. It sets the content type of the response to JSON.

HandleUsers Function

The handleUsers function is a handler function for the "/wafers" endpoint. It checks the HTTP method of the request and calls either the get function or the errorHandler function accordingly.

Get Function

The get function is called when an HTTP GET request is received on the "/wafers" endpoint. It marshals the w2 wafer instance to JSON and writes the resulting JSON to the response.

Next, it generates a random sleep time between 0 and 9 seconds and updates the w2 instance's fields: Numchip, Etime, Stime, and Diameter. It then writes the updated w2 instance to the response.

Randate Function

The randate function generates a random time between January 1st, 2023, and January 1st, 2024.

ErrorHandler Function

The errorHandler function is called when an error occurs. It sets the HTTP status code of the response, sets the content type of the response to JSON, and writes the error message to the response.

//

Java Code:

This code is a Java program that receives JSON objects from a URL and saves them to a MySQL database. The program creates a GUI using the Swing library, which displays the received JSON objects in a text area and provides a button to save the data to the database. The program uses a timer to continuously check for new JSON objects on the specified URL.

GUI:

The program creates a JFrame object. The text area is created using the JTextArea class, and is set to be non-editable. The JScrollPane class is used to add a scroll bar to the text area. The JLabel class is used to add a label to the top of the GUI that reads "Receiving JSON objects". The JButton class is used to create a button at the bottom of the GUI that reads "Save to Database".

Timer:

The program uses a timer to continuously check for new JSON objects on the specified URL. The timer is created using the Timer class from the javax.swing package. It is set to trigger every 10 seconds and executes an ActionListener which reads the JSON objects from the specified URL and appends them to the text area.

Saving to Database:

The program saves the received JSON objects to a MySQL database using the JDBC driver. When the "Save to Database" button is pressed, the JSON objects in the text area are retrieved and parsed using the JSONObject class from the org.json package. The parsed data is then inserted into the MySQL database using a prepared statement.

Database Connection:

The program establishes a connection to the MySQL database using the DriverManager class from the java.sql package. The connection details, including the database URL, username, and password, are specified in the code. The program loads the JDBC driver using the Class.forName() method and creates a connection object using the DriverManager.getConnection() method.

Problems:

There was an issue with connecting to the database and the problem was the MySQL connector due to its version.