````
## gRPC Distributed Statement Assignment 6

### Project Description

This assignment demonstrates a distributed system using gRPC. The project implements several microservices
that interact over gRPC, including services for echoing messages, delivering jokes, managing a flower garden,
tracking weight(with BMI calculation ), and ordering food. A registry service is also included, allowing nodes to
register themselves  and enabling clients to discover available services dynamically.

### How to Run the Program

The project supports different run configurations depending on whether you want to use the registry service or
run a standalone node:

- Without Registry:
   1)  Open a terminal and start the node:
  ```
  gradle runNode
   ```
  2) In another terminal, start the client:
  ```
  gradle runClient
  ```

- With Registry:
   1) Start the registry server in a terminal:
  ```
  gradle runRegistryServer
  ```

  2) In a second terminal, start the node with registry enabled:
  ```
  gradle runNode -PregOn=true

  ```
  3) Finally, in a third terminal, start the client with registry enabled:
  ```
  gradle runClient -PregOn=true
  gradle runClient2 -PserviceHost=localhost -PservicePort=8000 -PregistryHost=localhost -PgrpcPort=9002 -Pmessage="Hello from Client2"
  ```

### How to Work with the Program

- Interactive Client Interface:

The client application includes an interactive menu allowing you to choose from available services. Depending on your
selection, the program will prompt for further input (e.g., entering a message for the Echo Service, choosing a joke,
or placing a food order).

- Service Discovery and Registration:

When running with the registry, nodes automatically register their available gRPC
````

services. The client can query the registry to retrieve a list of registered services, facilitating dynamic service discovery and communication between distributed components.

- Development in an IDE:

Import the project as a Gradle project in your preferred IDE (e.g., IntelliJ or Eclipse). Follow the setup instructions provided in the project's configuration files to resolve any dependency or configuration issues.

Requirements Fulfilled

- gRPC Service Implementation:

The project implements multiple gRPC services (Echo, Joke, Flower, Weight Tracker, Food Ordering) that communicate over the network.

- Distributed Registry Service:

Nodes register themselves with a central registry that enables clients to discover available services dynamically.

- Interactive Client Application:

The client application includes an interactive menu that allows the user to select and interact with various services.

- Gradle Build Integration:
-

The project is built and run using Gradle, with tasks configured for both local testing and distributed deployment.

- Robust Error Handling:

Service methods include error handling to manage network issues and service unavailability, ensuring a smooth client experience.

Resources

- [GitHub](https://github.com/Bjablaso/ser321-spring25-A-Bjablaso/tree/main/Assignment6)
- [ScreenCast](https://youtu.be/9unvMbOVDKA)