Programación de Objetos Distribuidos
Soluciones Trabajo Práctico
gRPC - Communication Patterns

Ejercicio 1

1.1

Los números de cada uno de los campos del mensaje deben ser valores positivos y no pueden repetirse. Sólo los valores del enum empiezan en 0.

Por lo tanto no se puede declarar el campo `query` con 0.

Sí se pueden declarar los campos en desorden (por ejemplo declarando primero un campo con valor dos y una línea después un campo con valor uno).

El ejemplo del enunciado puede corregirse así:

```
message SearchRequest {
 string query = 1;
 uint32 page_number = 2;
 uint32 result_per_page = 3;
}
```

1.2

Los métodos de gRPC deben recibir un mensaje y retornar un mensaje. Por lo tanto no se pueden declarar los métodos Add y `Substract` con dos parámetros, es necesario un mensaje que encapsule esos dos parámetros `double`.

```
message OperationRequest {
 double x = 1;
 double y = 2;
}
```

Tampoco se puede declarar el método `Random` que no recibe un mensaje de request. Para estos casos se puede declarar un mensaje vacío:

```
message EmptyMessage {
}
```

La definición completa del servicio es la siguiente:

```
message OperationRequest {
 double x = 1;
```

```
 double y = 2;
}

message EmptyMessage {
}

message DoubleResponse {
 double value = 1;
}

service CalculatorService {
 rpc Add(OperationRequest) returns (DoubleResponse);
 rpc Substract(OperationRequest) returns (DoubleResponse);
 rpc Random(EmptyMessage) returns (DoubleResponse);
}
```

En el paquete `google.protobuf` ya se encuentran definidos algunos mensajes útiles como el mensaje vacío y varios mensajes "*wrappers*" de los tipos primitivos. De esta forma no se repiten las declaraciones de mensajes entre varios servicios. En lugar de declarar un mensaje `DoubleResponse` con un campo primitivo `double` basta con usar `google.protobuf.DoubleValue`. Para el mensaje vacío también existe `google.protobuf.Empty`.

La definición completa del servicio ahora queda de la siguiente forma:

```
import "google/protobuf/wrappers.proto";
import "google/protobuf/empty.proto";

message OperationRequest {
 double x = 1;
 double y = 2;
}

service CalculatorService {
 rpc Add(OperationRequest) returns (google.protobuf.DoubleValue);
 rpc Substract(OperationRequest) returns (google.protobuf.DoubleValue);
 rpc Random(google.protobuf.Empty) returns (google.protobuf.DoubleValue);
}
```

1.3

Protobuf no soporta el concepto de valores de tipo enum vacíos, es decir, en caso de consultar un campo de tipo enum siempre retornará un valor válido incluso si nunca se lo seteó.

Para el ejemplo del enunciado, si desde el Servant se instancia un `FlightStatusResponse` y nunca se setea el campo `flightStatus`, cuando el cliente lo consulte obtendrá un valor por defecto que corresponde al ordinal 0 es decir la constante `ON_TIME` y no corresponde.

Es por eso que una buena práctica es declarar una constante extra correspondiente al primer ordinal para poder distinguir estos casos. En el ejemplo basta con declarar una constante `FLIGHT_STATUS_UNSPECIFIED`.

El mensaje `FlightStatusRequest` sólo contiene un campo de tipo String y puede reemplazarse por el mensaje `google.protobuf.StringValue`. Los enums no son mensajes por lo que es necesario declarar un `message` que tenga un campo de tipo enum, incluso si sólo se necesita ese campo.

```
service FlightService {
 rpc FlightStatus(google.protobuf.StringValue) returns (FlightStatusResponse);
}

enum FlightStatus {
 FLIGHT_STATUS_UNSPECIFIED = 0;
 ON_TIME = 1;
 DELAYED = 2;
 CANCELLED = 3;
}

message FlightStatusResponse {
 FlightStatus flight_status = 1;
}
```

Ejercicio 2

Servicio

```
syntax = "proto3";

package ping;

import "google/protobuf/wrappers.proto";
import "google/protobuf/empty.proto";

service PingService {
 rpc Ping(google.protobuf.Empty) returns (google.protobuf.StringValue);
 rpc Time(google.protobuf.Empty) returns (google.protobuf.UInt32Value);
 rpc Echo(google.protobuf.StringValue) returns (google.protobuf.StringValue);
 rpc Hello(google.protobuf.StringValue) returns (google.protobuf.StringValue);
 rpc Fortune(google.protobuf.Empty) returns (google.protobuf.StringValue);
}
```

Servant

```
public class Servant extends PingServiceGrpc.PingServiceImplBase {

   private final List<String> fortuneQuotes;

   public Servant(List<String> fortuneQuotes) {
       this.fortuneQuotes = fortuneQuotes;
```

```
    }

    @Override
    public void ping(Empty request, StreamObserver<StringValue>
responseObserver) {
        responseObserver.onNext(StringValue.of("pong"));
        responseObserver.onCompleted();
    }

    @Override
    public void time(Empty request, StreamObserver<UInt32Value>
responseObserver) {
        responseObserver.onNext(UInt32Value.of(LocalDateTime.now().getHour()));
        responseObserver.onCompleted();
    }

    @Override
    public void echo(StringValue request, StreamObserver<StringValue>
responseObserver) {
        responseObserver.onNext(request);
        responseObserver.onCompleted();
    }

    @Override
    public void hello(StringValue request, StreamObserver<StringValue>
responseObserver) {
        responseObserver.onNext(StringValue.of("Hello
%s".formatted(request.getValue())));
        responseObserver.onCompleted();
    }

    @Override
    public void fortune(Empty request, StreamObserver<StringValue>
responseObserver) {
        String quote = fortuneQuotes.get(new SecureRandom().ints(1, 0,
fortuneQuotes.size())
                .findAny()
                .getAsInt());
        responseObserver.onNext(StringValue.of(quote));
        responseObserver.onCompleted();
    }

}
```

Servidor

```
public class Server {

    public static void main(String[] args) throws IOException,
InterruptedException {
        List<String> fortuneQuotes = List.of("Today it's up to you to create
```

```
the peacefulness you long for.",
                "A friend asks only for your time not your money.",
                "If you refuse to accept anything but the best, you very often
get it.",
                "A smile is your passport into the hearts of others.",
                "A good way to keep healthy is to eat more Chinese food.");
        Servant servant = new Servant(fortuneQuotes);

        int port = 50051;
        io.grpc.Server server = ServerBuilder.forPort(port)
                .addService(servant)
                .build();
        server.start();
        System.out.println("Server started, listening on " + port);
        server.awaitTermination();
        Runtime.getRuntime().addShutdownHook(new Thread(() -> {
            System.out.println("Shutting down gRPC server since JVM is shutting
down");
            server.shutdown();
            System.out.println("Server shut down");
        }));
    }

}
```

Cliente

```
public class Client {

    public static void main(String[] args) {
        ManagedChannel channel = ManagedChannelBuilder.forAddress("localhost",
50051)
                .usePlaintext()
                .build();

        PingServiceGrpc.PingServiceBlockingStub blockingStub =
PingServiceGrpc.newBlockingStub(channel);

        System.out.println(blockingStub.ping(Empty.getDefaultInstance()));
        System.out.println(blockingStub.time(Empty.getDefaultInstance()));
        System.out.println(blockingStub.echo(StringValue.of("Hello World!")));
        System.out.println(blockingStub.hello(StringValue.of("world")));
        System.out.println(blockingStub.fortune(Empty.getDefaultInstance()));
    }

}
```