Testing gRPC

The tests in the Hello World example illustrates use of the **JUnit (https://junit.org/)** framework. The test coverage is not complete. It only shows how to get started with testing gRPC services. You could add to it as an exercise to increase your own knowledge.

Let's look at the test class definition in the `GreeterTest.java` source file:

```
copypackage com.example.helloworld;

import akka.actor.testkit.typed.javadsl.ActorTestKi
import akka.actor.testkit.typed.javadsl.TestKitJuni
import akka.actor.typed.ActorSystem;
import akka.actor.typed.javadsl.Behaviors;
import akka.grpc.GrpcClientSettings;
import akka.http.javadsl.ServerBinding;

import com.typesafe.config.Config;
import com.typesafe.config.ConfigFactory;

import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.ClassRule;
import org.junit.Test;

import java.util.concurrent.CompletionStage;
import java.util.concurrent.TimeUnit;

import static org.junit.Assert.assertEquals;

public class GreeterTest {

  // important to enable HTTP/2 in server ActorSyst
  private static final Config config = ConfigFactor
          .parseString("akka.http.server.preview.en
          .withFallback(ConfigFactory.defaultApplic

  @ClassRule
  public static final TestKitJunitResource testKit

  private static ActorSystem<?> serverSystem = test
  private static ActorSystem<?> clientSystem;
  private static GreeterServiceClient client;

  @BeforeClass
  public static void setup() throws Exception {
    CompletionStage<ServerBinding> bound = new Gree
    // make sure server is bound before using clien
    bound.toCompletableFuture().get(5, TimeUnit.SEC

    clientSystem = ActorSystem.create(Behaviors.emp
    // the host and TLS certificate config are pick
    client = GreeterServiceClient.create(
        GrpcClientSettings.fromConfig("helloworld.G
        clientSystem
      );
```

```
  }

  @AfterClass
  public static void teardown() {
    ActorTestKit.shutdown(clientSystem);
    client = null;
  }

  @Test
  public void greeterServiceRepliesToSingleRequest(
    HelloReply reply = client.sayHello(HelloRequest
        .toCompletableFuture()
        .get(5, TimeUnit.SECONDS);
    HelloReply expected = HelloReply.newBuilder().s
    assertEquals(expected, reply);
  }

}
```

Note how we create two `ActorSystem`s, one for the server and another for the client. The test is then using the client to verify that it retrieves the expected responses from the server.

Unit testing

The above test example is a full integration test using real client and server, including communication via HTTP/2. For some testing of the service implementation it might be more appropriate to write unit tests without interaction via the gRPC client. Since the service interface and implementation doesn't require any gRPC infrastructure it can be tested without binding it to a HTTP server.

```
copypackage com.example.helloworld;

import akka.actor.testkit.typed.javadsl.ActorTestKi
import akka.actor.testkit.typed.javadsl.TestKitJuni
import akka.actor.typed.ActorSystem;

import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.ClassRule;
import org.junit.Test;

import java.util.concurrent.TimeUnit;

import static org.junit.Assert.assertEquals;

public class GreeterServiceImplTest {

  @ClassRule
  public static final TestKitJunitResource testKit

  private static ActorSystem<?> system = testKit.sy
  private static GreeterService service;

  @BeforeClass
  public static void setup() {
    service = new GreeterServiceImpl(system);
  }

  @Test
  public void greeterServiceRepliesToSingleRequest(
    HelloReply reply = service.sayHello(HelloReques
        .toCompletableFuture()
        .get(5, TimeUnit.SECONDS);
    HelloReply expected = HelloReply.newBuilder().s
    assertEquals(expected, reply);
  }

}
```

Add streaming tests

As an exercise to increase your understanding you could add tests for the **streaming call (streaming.html)**, both as integration test and unit test style.

The Akka documentation of **Testing streams (https://doc.akka.io/docs/akka/current/stream/stream-testkit.html)** might be useful.