Certainly! Let's provide two code examples to illustrate Java web services, one for SOAP and one for REST.

Example 1: Creating a SOAP Web Service with JAX-WS

```java
import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public class Calculator {

    @WebMethod
    public int add(int a, int b) {
        return a + b;
    }
}
```

In this example, we create a simple SOAP web service using JAX-WS. The `Calculator` class is annotated with `@WebService`, indicating that it's a web service. The `add` method is annotated with `@WebMethod`, making it accessible as a web service operation. Clients can call this service by sending SOAP messages to the appropriate endpoint.

Example 2: Creating a RESTful Web Service with JAX-RS

```java
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;

@Path("/books")
public class BookService {

    @GET
    @Path("/{id}")
    public String getBook(@PathParam("id") int id) {
        // Retrieve book information from a data source and return it as JSON or XML.
        return "Book #" + id + ": Introduction to Java";
    }
}
```

In this example, we create a RESTful web service using JAX-RS. The `BookService` class is annotated with `@Path("/books")`, specifying the base URI for this resource. The `getBook` method is annotated with `@GET` and `@Path("/{id}")`, indicating that it responds to HTTP GET requests at a URL like `/books/{id}`. It retrieves book information based on the `id` provided in the URL and returns it as a response, typically in JSON or XML format.

These examples demonstrate the basic structure and annotations used to create SOAP and REST web services in Java. Actual implementations may involve more complex logic, data access, and security considerations, but these examples provide a starting point for understanding how to build web services in Java.