The return type is `int[ ]`, an array of `int` values. In the *wsimport*-generated interface `RandService`, the method `nextN` begins:

```
public List<Integer> nextN(...
```

The *wsimport* utility is within its rights to replace `int[ ]` with `List<Integer>`, as a `List` has a `toArray` method that returns an array; with automatic boxing/unboxing, the Java types `Integer` and `int` are interchangeable in the current context. The point is that the programmer typically needs to inspect at least the *wsimport*-generated interface, in this example `RandService`, in order to determine the argument and return types of every operation.

Companion Utilities: wsimport and wsgen

The *wsimport* utility eases the task of writing a Java client against a service that has a WSDL as the service contract. This utility has a client-side focus, although the utility can be helpful on the server side as well; a later example illustrates. The *wsgen* utility, which also ships with core Java 1.6 or greater, has a server-side focus. For example, *wsgen* can be used to generate a WSDL. The command:

```
% wsgen -cp . -wsdl rand.RandService
```

generates a WSDL file named *RandServiceService.wsdl*. However, this WSDL has a placeholder for the service endpoint rather than a usable URL:

```
...
<soap:address location="REPLACE_WITH_ACTUAL_URL"/>
...
```

When a service publisher such as `Endpoint`, Tomcat, Jetty, and the like generate the WSDL, the WSDL includes a usable URL.

The *wsgen* utility has another use. When the `RandService` is published with `Endpoint`, the publisher outputs information about dynamically generated classes, in this case `Next1`, `Next1Response`, `NetxN`, and `NextNResponse`. As noted earlier, these are JAX-B artifacts that the Java runtime uses to convert Java types into XML types and vice versa. The *wsgen* utility can be used to generate the JAX-B artifacts as files on the local system. For example, the command:

```
% wsgen -cp . rand.RandService
```

automatically creates a package/directory *rand/jaxws* and then populates this directory with *Next1.class*, *Next1Response.class*, *NextN.class*, and *NextNResponse.class*. Now if the `Endpoint` publisher is started after these files have been created, the publisher does not generate the JAX-B artifacts dynamically but instead uses the ones that *wsgen* already created.