Java Mapping for Modules

A Slice module maps to a Java package with the same name as the Slice module. The mapping preserves the nesting of the Slice definitions. For example:

Slice

```
// Definitions at global scope here...

module M1 {
    // Definitions for M1 here...
    module M2 {
        // Definitions for M2 here...
    };
};

// ...

module M1 {     // Reopen M1
    // More definitions for M1 here...
};
```

This definition maps to the corresponding Java definitions:

Java

```
package M1;
// Definitions for M1 here...

package M1.M2;
// Definitions for M2 here...

package M1;
// Definitions for M1 here...
```

Note that these definitions appear in the appropriate source files; source files for definitions in module `M1` are generated in directory `M1` underneath the top-level directory, and source files for definitions for module `M2` are generated in directory `M1/M2` underneath the top-level directory. You can set the top-level output directory using the `--output-dir` option with `slice2java`.

See Also

- Modules
- Using the Slice Compilers
- Java Mapping for Identifiers
- Java Mapping for Built-In Types
- Java Mapping for Enumerations
- Java Mapping for Structures
- Java Mapping for Sequences
- Java Mapping for Dictionaries
- Java Mapping for Constants
- Java Mapping for Exceptions