17.17 Additional Notes on Java SE 8 Interfaces

Java SE 8 Interfaces Allow Inheritance of Method Implementations

8

Functional interfaces *must* contain only one `abstract` method, but may also contain `default` methods and `static` methods that are fully implemented in the interface declarations. For example, the `Function` interface—which is used extensively in functional programming—has methods `apply` (`abstract`), `compose` (`default`), `andThen` (`default`) and `identity` (`static`).

When a class implements an interface with `default` methods and does *not* override them, the class inherits the `default` methods' implementations. An interface's designer can now evolve an interface by adding new `default` and `static` methods without breaking existing code that implements the interface. For example, interface `Comparator` (Section 16.7.1) now contains many `default` and `static` methods, but older classes that implement this interface will still compile and operate properly in Java SE 8.

Recall that one class can implement many interfaces. If a class implements two or more unrelated interfaces that provide a `default` method with the same signature, the implementing class *must* override that method; otherwise, a compilation error occurs.

Java SE 8: `@FunctionalInterface` Annotation

8

You can create your own functional interfaces by ensuring that each contains only one `abstract` method and zero or more `default` and/or `static` methods. Though not required, you can declare that an interface is a functional interface by preceding it with the `@FunctionalInterface` **annotation**. The compiler will then ensure that the interface contains only one `abstract` method; otherwise, it will generate a compilation error.