Dagger 2 Conventions Demonstrated in The Course

1. Components are interfaces annotated with @Component annotation

Components can be both composition roots and injectors.

Annotation also specifies which modules are used by the Component.

2. Modules are classes annotated with @Module annotation

Modules contain the logic that composes the provided services.

3. Provider methods in Modules annotated with @Provides annotation

Provider methods either instantiate new services or return references to "bootstrapping dependencies" (dependencies that were injected into Module at construction time).

4. Provided services can be used as method arguments in other provider methods

Dagger analyzes declarations of all provider methods and, after ensuring that all dependencies satisfied, wires them together.

5. Components cache services provided from methods annotated with @Singleton "scope" annotation

This effectively ensures that only one instance of "scoped" service will be returned by the same Component.

Note that scoped services are required only in ApplicationComponent because all other Components must not be used more than once per lifecycle.

6. Component's void methods with single argument allow injection into clients of type of an argument

Though the name of injector methods can be anything, it is recommended to name them "inject".

7. Client's non-private, non-final fields, annotated with @Inject annotation will be injected by Component's injectors

The fields must be either public or package-private because Dagger doesn't use reflection.

8. Inter-dependencies between Components specified as part of @Component annotation

Component B that depends on Component A has implicit access to all services exposed by A:

- can be used in injections by B
- can be used by provider methods inside Modules of B

9. If Component A is scoped, then all Components that depend on it must be scoped too

Honestly, I don't know why this convention exists.

10. Subcomponents specified by @Subcomponent annotation

As opposed to dependent Components, Subcomponents don't need to be scoped.

11. Parent Component must expose factory method that will return a new instance of Subcomponent

The arguments of this method are the modules used by Subcomponent.

Subcomponents have implicit access to all services provided by parent Component.

- can be used in injections by Subcomponent
- can be used by provider methods inside Modules of Subcomponent

12. Components can use more than one module

All Modules of a single Component share the same objects graph.

Dagger can implicitly instantiate Modules with no-arguments constructors.

13. Dagger can automatically discover services that have public constructor annotated with @Inject annotation

Dagger makes sure that all constructor's dependencies can be satisfied.