Conforming to `StringInterpolationProtocol`

A type conforming to `ExpressibleByStringInterpolation` uses a helper type called `StringInterpolation` to perform its interpolation. Many types can use `DefaultStringInterpolation`, which implements `String`'s interpolation behavior. Types can also implement custom behavior by providing their own type conforming to `StringInterpolationProtocol`.

In addition to its formal requirements, `init(literalCapacity:interpolationCount:)` and `appendLiteral(_:)`, `StringInterpolationProtocol` has an additional, informal requirement, `appendInterpolation`. String interpolations using `\()` syntax are translated into calls to matching `appendInterpolation` methods.

`StringInterpolationProtocol` conformers must provide at least one `appendInterpolation` method which:

- Is an instance method, as opposed to a `static` or `class` method
- Does not specify a return type, explicitly returns `Void`, or is marked with the `@discardableResult` attribute
- Is at least as accessible as its containing type

There are no restrictions on an `appendInterpolation` method's argument list, generic parameters, availability, or error-throwing behavior.

If `appendInterpolation` is overloaded, the Swift compiler will choose an appropriate overload using the labels and argument types of each interpolation. When choosing an overload, any accessible `appendInterpolation` instance method may be used, even if it does not meet all of the requirements above. However, if a `StringInterpolationProtocol` conformer doesn't have any `appendInterpolation` methods which meet all of the requirements, an error will be reported at compile time.

To learn more about customizing string interpolation behavior, see the standard library documentation of the `ExpressibleByStringInterpolation` and `StringInterpolationProtocol` protocols.