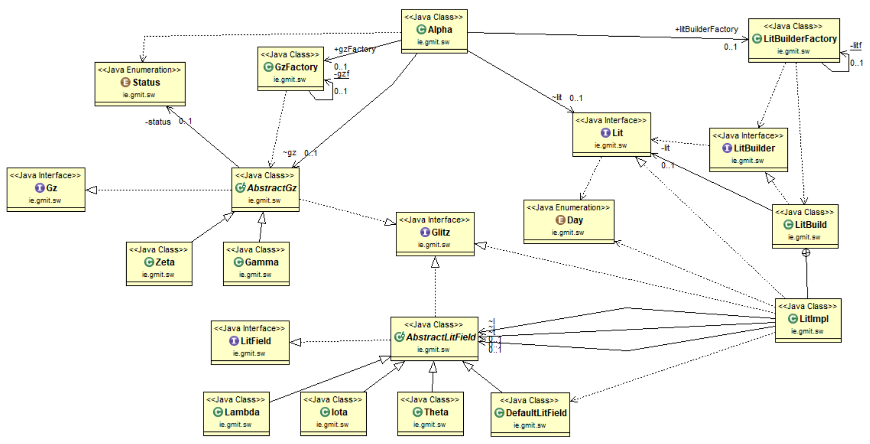

The classes Gamma, Lambda, Iota, Theta and Zetta share a common attribute: about, which is derived from the Glitz interface.

The classes Gamma and Zeta extend AbstractGz class, which implements both the Gz and Glitz interface. A factory build here is fit for purpose. This design pattern allows for abstraction as the user is interested in a Status input and not the class which returns. The GzFactory class returns an AbstractGz, which can be Gamma or Zetta. The classes Gamma and Zeta share a theme, they are built with a Status input. The AbstractGz class contains a private status field, with an accessor method. This field is set on construction.

The classes Lambda, Iota and Theta extend the AbstractLitField, which implements both the LitField and the Glitz interface. These 3 classes are described as being "orchestrated together to create complex types", this requires a builder design pattern as this allows these classes to be combined for a single result. The LitBuilderFactory returns a LitBuilder interface, which is implemented by the LitBuild class. The LitBuild returns the inner-class, LitImpl with its build method. This inner-class implements the Lit and Glitz interfaces and contains 3 AbstractLitField's, these are set to Lambda, Iota, Theta when built by the user. The classes Lambda, Iota and Theta share a theme, they can be combined together. To achieve this I used a binary count to 8, with each of the three classes having a value. The sum of these values returns a day.

The Alpha class contains GzFactory which creates the AbstractGz object dependant on the Status input, and a LitBuilderFactory, which builds a Lit object.