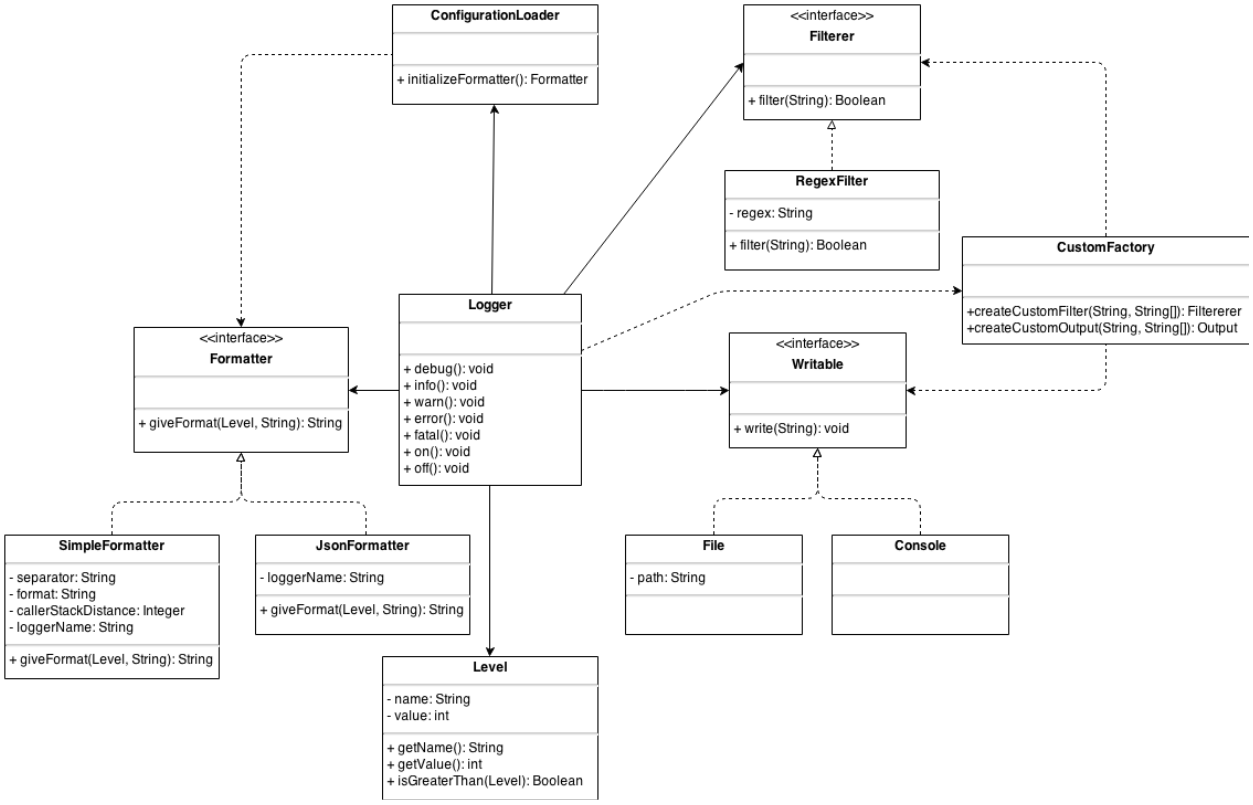
Logger



Se observan cinco interacciones principales con el Logger:

- Formatter: Genera el mensaje de log correspondiente al formato dado por el usuario
- Level: Representa el nivel en el cual se van a loguear los mensajes
- Writable: Interfaz genérica representando todos los medios posibles para la salida
- Configuration: Se encarga de cargar la configuración asociada
- Filterer: Filtra aquellos mensajes que cumplen determinados requisitos.

Writable

Se quería buscar la forma de que una entidad externa, en este caso el logger, posea una colección ilimitada de medios donde pueda escribir la salida. Sin embargo, la intención era que no se diferencie el tipo de medio, sino que hacia afuera sólo se desea escribir determinado mensaje sin importar las limitaciones técnicas de cada dispositivo.

Por esta razón se modeló como una interfaz llamada “Writable” donde cada nuevo dispositivo de salida debe implementar de la forma correspondiente el método “write” que solo recibe el mensaje en cuestión a escribir.

Uno de los problemas que se presentaron fue que dependiendo del caso, la acción de escribir puede generar una excepción, pero también es posible que esto no ocurra nunca.

Pensando en la extensibilidad a cualquier tipo de medio de salida, no era correcto forzar a que la interfaz arroje excepciones de tipos específicos. Se encapsularon entonces todos los tipos de excepciones bajo una abstracta llamada “WriteException” que es la forma correcta de informar un error bajo esta interfaz, sin importar el tipo real de excepción particular a cada medio.

Formatter

Encargado de generar el mensaje a loguear a partir de un mensaje a loguear recibido, un nivel de logueo.

Para la primera entrega, solo se pedían formatos simples de texto. Supusimos que podría suceder que más adelante nos pidan otros formatos posibles como json, xml, etc. Si bien en un principio se pensó en una clase abstracta, se tuvo que descargar esta posibilidad puesto que se obtenía un diseño muy acoplado en el que cualquier formatter hubiera tenido un separador y un formato. Fue por ello que decidimos que lo mejor era una interfaz con un método giveFormat, el cual cualquier clase que implementara esta interfaz debería definir para darle el formato al log a su manera.

Para la segunda entrega, se agregó el Formato Json. El diseño hecho anteriormente resultó adecuado, por lo que la integración del JsonFormatter resultó sencilla. Simplemente haciendo que el JsonFormatter implementara Formatter logramos que el Logger pudiera usarlo de la misma forma que lo hacía con el SimpleFormatter.

Level

Se definió una clase llamada Level con el objetivo de diferenciar los distintos niveles de logueo. Mediante su uso el Logger puede definir en qué nivel se encuentra de acuerdo a la configuración inicial, así como también determinar si un mensaje debe ser logueado o no. Debido a que en un determinado nivel de logueo se deben loguear los mensajes de niveles inferiores, a cada nivel se le asignó un valor numérico con el objetivo de facilitar la comparación.

En el caso de que se necesite crear un nuevo nivel, éste se agrega al enum y se crea el método correspondiente para loguear en ese nivel.

Logger collection

Para soportar el uso de múltiples loggers se definió una nueva estructura llamada LoggerCollection que contiene a todos los loggers agregados.

Esta estructura fue declarada de tipo singleton para que no haya confusion mediante la distribucion de los loggers en distintas estructuras contenedoras. De esta forma forzamos a que la colección sea única y contenga a todos los objetos loggers que se desee.

Filterer

Encargado de indicar que mensajes deben ser filtrados, de acuerdo a un determinado requisito. Se pensó en la Interfaz Filterer para que el logger pueda abstraerse de quien es que filtra el log y utilizar de la misma forma cualquier Filtro. En esta entrega se pidió un RegexFilter, el cual indica si un mensaje debe ser filtrado o no de acuerdo a la regular expression que se le setea. Además se admiten CustomFilters, los cuales deberían implementar está interfaz, logrando de esta forma que el Logger los trate como cualquier otro filtro.

CustomFactory

Es una Factory que se encarga de crear los CustomFilter y los OutputFilter. Con ella se logró reutilizar la funcionalidad necesaria para crear un objeto Custom para luego poder usarlo en el logger como cualquier otro Filterer o Writable respectivamente.