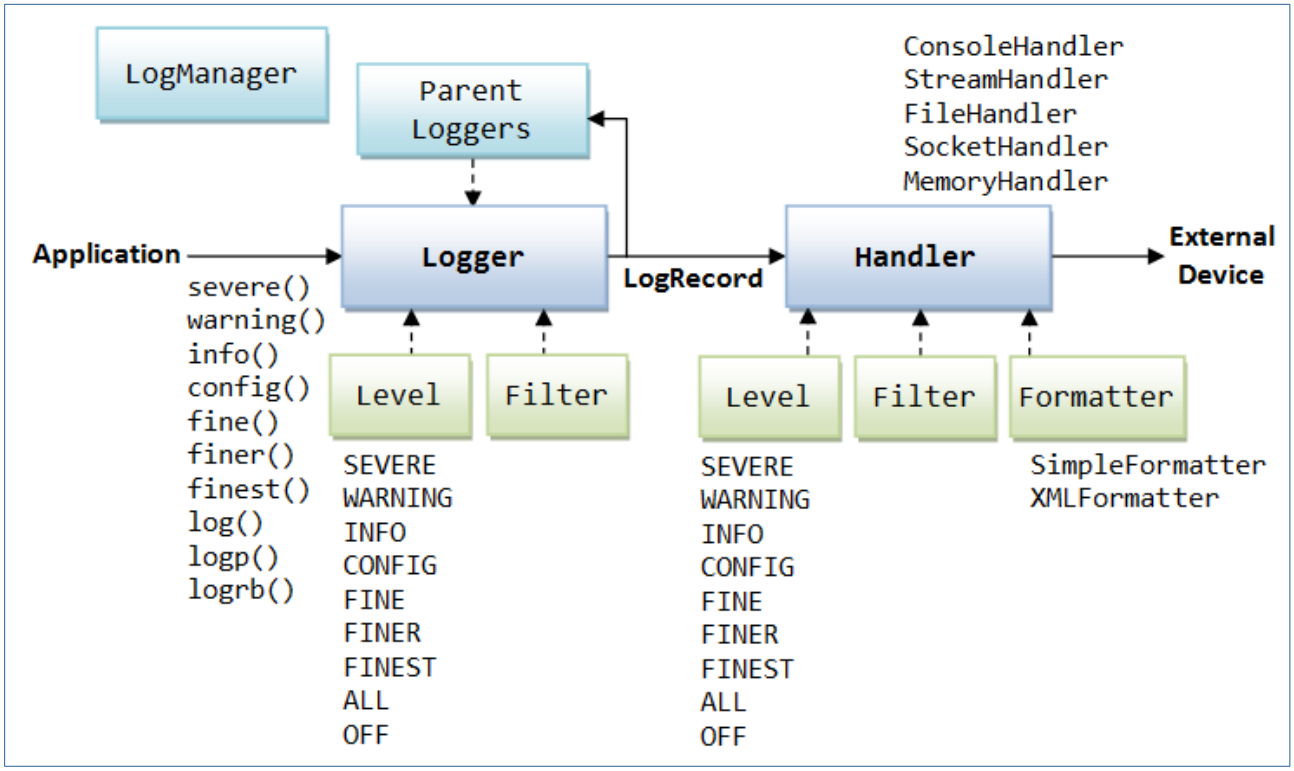

1. LogManager

- **Funktion**: Der `LogManager` verwaltet alle Logger in der Anwendung. Er ist für das Erstellen, Registrieren und Konfigurieren von Loggern verantwortlich. Jeder Logger wird vom `LogManager` verwaltet.
- **Beispiel**: Wenn ein Logger erstellt wird (z. B. `Logger.getLogger(MyClass.class.getName())`), wird dieser Logger vom `LogManager` verwaltet. Er bestimmt die Eigenschaften des Loggers wie Level, Handler und Filter.

2. Logger

- **Funktion**: Der `Logger` ist die zentrale Komponente, die für das Erzeugen von Protokollnachrichten zuständig ist. Jede Protokollierungsanforderung (`info()`, `warning()`, `severe()`, usw.) wird durch den Logger verarbeitet.
- **Levels**: Der Logger hat einen eigenen Log-Level. Dieser Level entscheidet, ob eine Nachricht mit einem bestimmten Schweregrad tatsächlich protokolliert wird.
    - Beispiel: Wenn der Logger-Level auf `WARNING` eingestellt ist, werden nur Nachrichten ab dem Level `WARNING` und höher (`SEVERE`) protokolliert, niedrigere Level wie `INFO` oder `CONFIG` werden ignoriert.
- **Filter**: Der Logger kann auch einen **Filter** haben. Ein Filter ist eine zusätzliche Komponente, die vor der Protokollierung prüft, ob eine Nachricht protokolliert werden soll, unabhängig vom Level. Er ermöglicht eine zusätzliche Feinkontrolle darüber, welche Nachrichten weiterverarbeitet werden.
    - **Beispiel**: Ein Filter könnte z. B. nur Nachrichten zulassen, die bestimmte Schlüsselwörter enthalten.

Parent Loggers: Logger sind hierarchisch organisiert. Wenn ein Logger keine eigenen Einstellungen (wie Level oder Handler) hat, erbt er diese von seinem Eltern-Logger. Dies ermöglicht eine flexible und zentrale Steuerung des Loggings in großen Anwendungen.

3. LogRecord

- **Funktion**: Der `LogRecord` ist ein Container, der alle Informationen über ein einzelnes Protokollierungsereignis enthält. Wenn eine Nachricht vom Logger erzeugt wird (z. B. `logger.info("Nachricht")`), wird ein `LogRecord` erstellt und an die Handler weitergeleitet.
- **Details**: Der `LogRecord` enthält Informationen wie den Log-Level, die Nachricht, den Logger-Namen, die Quellklasse, die Quellmethode, die Zeit des Ereignisses und optionale Ausnahmen (Exceptions).

4. Handler

- **Funktion**: Ein `Handler` ist verantwortlich dafür, wie und wohin die Log-Nachrichten gesendet werden. Jeder Logger kann mehrere Handler haben, die Nachrichten an verschiedene Ziele weiterleiten.
- **Beispiel**: Ein `ConsoleHandler` leitet Nachrichten an die Konsole weiter, ein `FileHandler` schreibt Nachrichten in eine Datei, und ein `SocketHandler` sendet Nachrichten über das Netzwerk.
- **Levels**: Genau wie der Logger hat auch jeder `Handler` einen eigenen Level. Dies bedeutet, dass ein Handler nur Nachrichten mit einem bestimmten Level verarbeitet oder weiterleitet.
    - **Beispiel**: Ein `FileHandler` könnte so konfiguriert sein, dass er nur `SEVERE`-Nachrichten in eine Datei schreibt, während ein `ConsoleHandler` alle Nachrichten anzeigt.
- **Filter**: Jeder `Handler` kann auch einen eigenen Filter haben, um zu steuern, welche Nachrichten weitergeleitet werden.
- **Formatter**: Ein `Formatter` ist dafür verantwortlich, wie die Protokollnachrichten formatiert werden, bevor sie ausgegeben werden. Beispielsweise wird der `SimpleFormatter` verwendet, um die Nachrichten in einfacher Textform auszugeben, während der `XMLFormatter` die Nachrichten im XML-Format protokolliert.

5. Filter

- **Funktion**: Ein Filter ist eine zusätzliche Komponente, die verwendet wird, um zu entscheiden, ob eine Nachricht tatsächlich protokolliert werden soll. Er funktioniert zusätzlich zum Level und bietet eine weitere Ebene der Kontrolle.
- **Anwendungsfall**: Ein Filter könnte zum Beispiel so konfiguriert sein, dass er nur Nachrichten mit einem bestimmten Inhalt oder Nachrichten, die von bestimmten Quellen kommen, protokolliert. Selbst wenn eine Nachricht den richtigen Level hat, kann ein Filter entscheiden, dass sie nicht protokolliert wird.

6. Externe Geräte (External Device)

- **Funktion**: Dies sind die tatsächlichen Ziele, an die die Log-Nachrichten gesendet werden. Dazu gehören z. B. die Konsole, eine Datei, eine Datenbank oder ein Netzwerkdienst. Der Handler steuert, welche externen Geräte angesprochen werden.

Warum haben Logger und Handler beide Levels?

- **Logger-Level**: Der Logger entscheidet, ob eine Nachricht auf Basis des Levels überhaupt verarbeitet wird. Wenn der Level des Loggers niedriger ist als der Level der Nachricht, wird die Nachricht ignoriert und nicht an die Handler weitergeleitet.
- **Handler-Level**: Selbst wenn eine Nachricht vom Logger akzeptiert wird, entscheidet der Handler noch einmal, ob er die Nachricht tatsächlich weiterleitet. Dies ermöglicht eine feinere Steuerung. Zum Beispiel könntest du den Logger so konfigurieren, dass er alle Nachrichten ab `INFO` verarbeitet, aber einen `FileHandler` nur `WARNING`-Nachrichten schreiben lassen und einen `ConsoleHandler` alle Nachrichten anzeigen lassen.

Fazit:

- **Logger**: Erzeugt und verwaltet die Protokollnachrichten. Er hat einen Level, der bestimmt, welche Nachrichten überhaupt in Betracht gezogen werden.
- **Handler**: Nimmt die Protokollnachrichten des Loggers und leitet sie an die entsprechenden Ausgabemedien (Konsole, Datei, etc.) weiter. Auch der Handler hat einen Level, der steuert, welche Nachrichten er akzeptiert.
- **Filter**: Bietet eine zusätzliche Möglichkeit, bestimmte Nachrichten auszuschließen, auch wenn sie den richtigen Level haben.
- **Formatter**: Bestimmt, in welchem Format die Nachrichten ausgegeben werden.