Formatted Logger

- Erstellen Sie eine Klasse `Logging`.
- Die Klasse `Logging` soll eine Methode `AddLogMessage(string msg)` haben. Diese Methode soll die Log-Message intern speichern. Es können also laufend Log-Nachrichten hinzugefügt werden.
- Fügen Sie zu der Klasse `Logging` eine Methode `PrintMessages(Func<string, string> formattingFunc)` hinzu. Diese Methode soll alle gespeicherten Log-Messages ausgeben. Vor der Ausgabe soll jede einzelne Nachricht mit dem übergebenen Funktionsobjekt angepasst werden.
- Bevor die Log-Nachrichten ausgegeben werden, sollen diese optional mit einer Formatierung versehen werden. Implementieren sie folgende Formatierungs-Methoden:
    - `FormatAddDateTime`: Stellt dem Message-String Datum & Zeit voran
    - `FormatUpperCase`: Wandelt alle Buchstaben im Message-String in Großbuchstaben um.
    - `FormatLengthLimit`: Länge der einzelnen Messages auf 30 Zeichen beschränkt. Schneidet alle Zeichen nach dem 30. Zeichen ab.
- Testen sie die Funktionsweise in einem passenden Hauptprogramm.

Erweiterung:

- Fügen Sie der Klasse `Logging` eine Methode `PrintMessagesFiltered(Predicate<string> filterFunc)` hinzu. Diese Methode soll eine „Filter-Funktion“ als Parameter entgegennehmen. Die Methode `PrintMessagesFiltered(…)` soll alle Log-Messages ausgeben, für die das übergebene Predicate den Wert true liefert.
- Implementieren sie zwei Filter-Funktionen: `FilterError` und `FilterWarning`. `FilterError` soll alle Messages auswählen, die den Teilstring "Error" enthalten. `FilterWarning` soll alle Messages auswählen, die den Teilstring "Warning" enthalten.
- Testen Sie die Methode `PrintMessagesFiltered` im Hauptprogramm