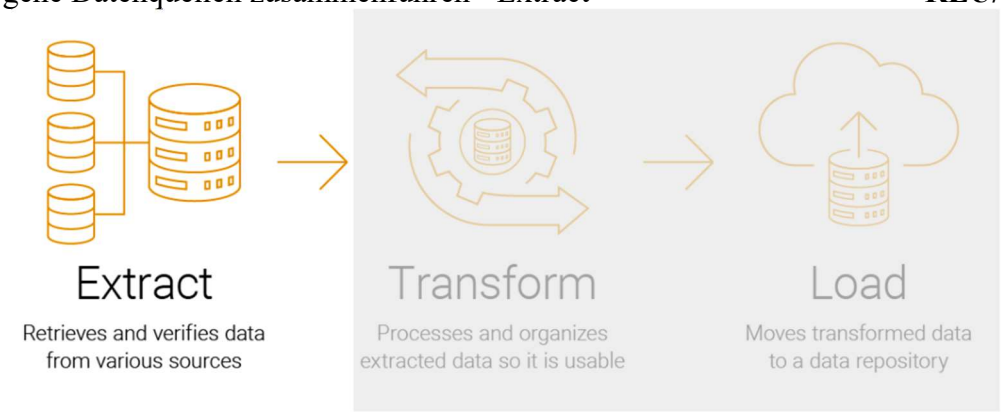


f

## Aufgabe 3 – Schritt E(xtract)

Um die Vereinsdaten zusammenzuführen, müssen diese im ersten Schritt „extrahiert“ werden, dieser Schritt **E** soll nun von Ihnen „per Hand“ implementiert werden.

Entwerfen Sie dazu zunächst ein Klassendiagramm und implementieren Sie danach ein Konsolenprogramm in C#, welches alle drei Datenformate einliesen und in die „Staging Area“ integrieren soll. Dabei können Sie davon ausgehen, dass der gesamte ETL Prozess als „Batch“ (=asynchron) ausgeführt wird, d.h. live bzw on-demand changes finden keine statt während des Prozesses.

**Beginnen Sie erst mit der Implementierung, wenn ihr Diagramm vorgestellt und ggf. überarbeitet wurde**.

## Anforderungen/Kriterien zur Bewertung:

Ihr Klassendiagramm…

- …unterteilt das Programm in mindestens fünf Klassen nach dem Konzept des Architekturmusters MVC (Model-View-Controller). Eine Klasse (oder jeweils eine Klasse pro Format) für das „Model,“ und eine weitere als “Controller” zum Einlesen und Bereitstellen jedes Datenformats. Schließlich eine Klasse für den „View,” um die eingelesenen Daten angemessen ausgeben lassen zu können und dem Benutzer ggf. Interaktion mit dem Programm zu ermöglichen.

- …berücksichtigt die Anforderungen des Programms (siehe unten)

Teilen Sie die Aufgaben zur Implementierung des Programms einigermaßen gleichmäßig untereinander auf und berücksichtigen Sie dabei folgende Randbedingungen:

- Die Gruppe definiert die Model-Klassen gemeinsam

- Jedes Gruppenmitglied implementiert mindestens eine der Controller-Klassen (im Kommentar der Klassen mit angeben)

## Anforderungen/Kriterien zur Bewertung des Programms

Ihr Programm…

- …kann die Datenformate JSON, XML und CSV einlesen. Dabei können Sie davon ausgehen, dass sich das Datenformat und deren Struktur (=Spalten) nicht ändert.

- [optional]… kann automatisch (ohne Nutzereingabe) auch mit Änderungen der Daten und / oder zusätzlichen Eingabedaten umgehen, d.h. wenn zusätzliche Spalten in den Daten vorliegen. Zusätzliche Typen müssen Sie nicht betrachten, alle vorhandenen reichen aus, um alle Möglichkeiten abzudecken.

- …prüft die Korrektheit der Eingabedaten (also z.B. korrekter Typ und / oder im Wertebereich der jeweiligen Daten)

- …prüft die Konsistenz der Eingabedaten (also z.B. dass gleiche Daten(sätze) auch gleiche Informationen enthalten, d.h. widerspruchsfrei sind)

- …ermöglicht grundsätzlich unterschiedliche Vorgehensweisen bei inkorrekten / inkonsistenten Daten bei einem „Einlesevorgang“ anhand einer vordefinierten Konfiguration (z.B. alle inkorrekten Daten werden verworfen)

- [optional] …ermöglicht einem Nutzer interaktiv auf inkorrekte /inkonsistente Daten zu reagieren, indem Ihr Programm verschiedene Vorgehensweise vorschlägt (z.B. mögliche Anpassungen der Daten / Reparaturen).

- …protokolliert den gesamten Prozess in einer sinnvollen und nachvollziehbaren Form (z.B. in Form einer Log-Datei / Konsolenausgaben)

- [optional]…gibt Feedback an den Nutzer hinsichtlich der Qualität der Daten anhand definierter Messwerte / Metriken.

- …implementiert eine „Staging Area“ in Form von POCO (plain old class objects), um die extrahierten Daten zu verwalten und für den Schritt „T“ verfügbar zu machen. Insbesondere die unterschiedlichen Datenformate/Daten **müssen ohne Änderung** integriert werden, dies spielt erst in Schritt „T“ eine Rolle.

- …ist vollständig kommentiert, insbesondere Begründungen bei Entscheidungen und/oder Umsetzungen von z.B. einzelnen Teile der Überprüfung der Daten oder der Datenstruktur in der Staging Area.