**Example Generic Interface with Implementation**

**This is an example to work with generic interface and its implementation, working with abstractions, not with real types of files/entities, ignores specific things for these entities, pass concrete classes into to another service, which worked with concrete entities to save to DB. This example incapsulates the common logic to get files, processing, parsing and then pass it for saving service.**

*There is a part of the real project, which processing 2 types of files from external source, which firstly saved to Azure storage (this part is not displayed here) and then parsing, processing and saving to local DB if it’s necessary after all actions (this part is not displayed here)*

* So, these 2 types of files, which have similar structure, but a little different. So, there is an abstract class named `AbstractDeviceDto` which consists all common fields. Classes `RegisteredDeviceDto` and `RevokedDeviceDto` are descendances of this abstract class.
* We have to observe, which new records are arrived since last updation, which were modified, deleted or possible duplicates (this part is not displayed here). Therefore we have also classes `CompareAbstractDevicesResultDto` and `CompareResultItemModifiedDto`.
* Interface `CompareResultItemModifiedDto` is generic because implementation should be the same for registered and revoked devices. Only one method `Workflow` incapsulates logic to get data, parsing, processing and saving to DB and should be executed as atomic operation. So, client code calls only this method in the result.
* Implementation of this method gets all unprocessed files (for this method no matter, which type of device is passed because it works with abstractions), then selects the oldest (in the real project there are some additional actions here, therefore we get a list firstly, not only the oldest file), then parse this file and pass to another service for saving to DB.