Task 2: Software Architecture

After much deliberation amongst your team, you have been trusted to design part of a new data processing pipeline. You will be responsible for coming up with the architecture for a dynamically reconfigurable data processor. The processor will have several different modes, each of which treats incoming data in a different way. Some modes will pass the result along to a database, and others will additionally validate the data against the database. To further complicate matters, there are several databases in play, and the processor needs the ability to toggle between them on the fly. The system needs to be extendable - your team has every intention of adding additional modes and databases in the future - so take extra care to ensure the system isn't rigid. Adhere to good design principles, use design patterns where they make sense, and remember your SOLID principles. Your project lead will use this to assess your ability to design clean code, so be sure to show off what you know. Success here could mean a promotion, don't squander the chance!

Here is your task

Your task is to draft a UML class diagram describing the data processors for a pipeline. The component responsible for reading in input data is being designed by another engineer, so you only need to worry about what happens to the data when it reaches your processor. You may assume three classes already exist:

Datapoint: this class represents both raw and processed data points. Any time data moves between methods you may use this class as an abstraction.

ModeIdentifier: an enum used to identify a processor mode.
DatabaseIdentifier: an enum used to identify a database connection.

Here are the requirements for your design:

- The processor must implement a configure method that accepts a ModeIdentifier and a DatabaseIdentifier as parameters.
    - This method is called to change the operating mode of the processor, and/or select the current database.
- The processor must be able to change between the following modes:
    - Dump mode: simply drops the data.
    - Passthrough mode: inserts the data into the currently configured database.
    - Validate mode: validates the data, then inserts it (both operations are carried out on the currently configured database).

- The processor must be able to swap between the following databases. Each database will require a different implementation to insert and validate data
    - Postgres.
    - Redis.
    - Elastic.
- The processor must implement a process method that accepts a DataPoint as a parameter.
    - This method will have different behavior depending on the currently configured mode and database.

There is no need to get into implementation specifics, keep things abstract and high level. For example, you need only specify connect, insert, and validate methods for each database, there is no need to specify how those methods go about performing their verbs. The point of this task is to think about how code is structured.