Task 2: SOFTWARE ARCHITECTURE

WALMART GLOBAL TECH

BY Samuel Waweru

@github : ***https://github.com/samkamau81***

Software Architecture can be undertaken using class diagrams. This are diagrams used to model software to describe classes and their relationships.

This can be done using UML (Unified Modelling Language) tools for Static Models.

Task: *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:

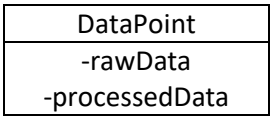
- *Dump mode: simply drops the data.*
- *Passthrough mode: inserts the data into the currently configured database.*
- *Validate mode: validates the data, then inserts it (both operations are carried out on the currently configured database).*

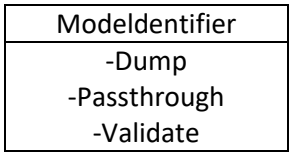
The processor must be able to swap between the following databases. Each database will require a different implementation to insert and validate dataPostgres. Redis. Elastic. The processor must implement a process method that accepts a DataPoint as a parameter.

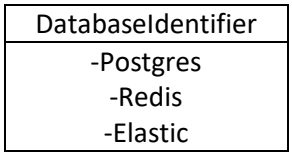
This method will have different behavior depending on the currently configured mode and database.

Here is a possible UML class diagram for the data processors in the pipeline:

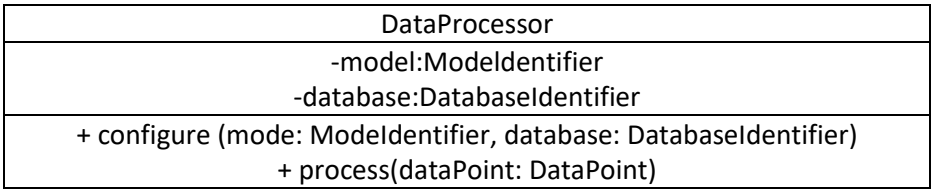
The Class Diagrams for the pipeline is;







The Class Diagram of the DataProcessor , that inherits Model Identifier, DatabaseIdentifier and dataPoint Classes.

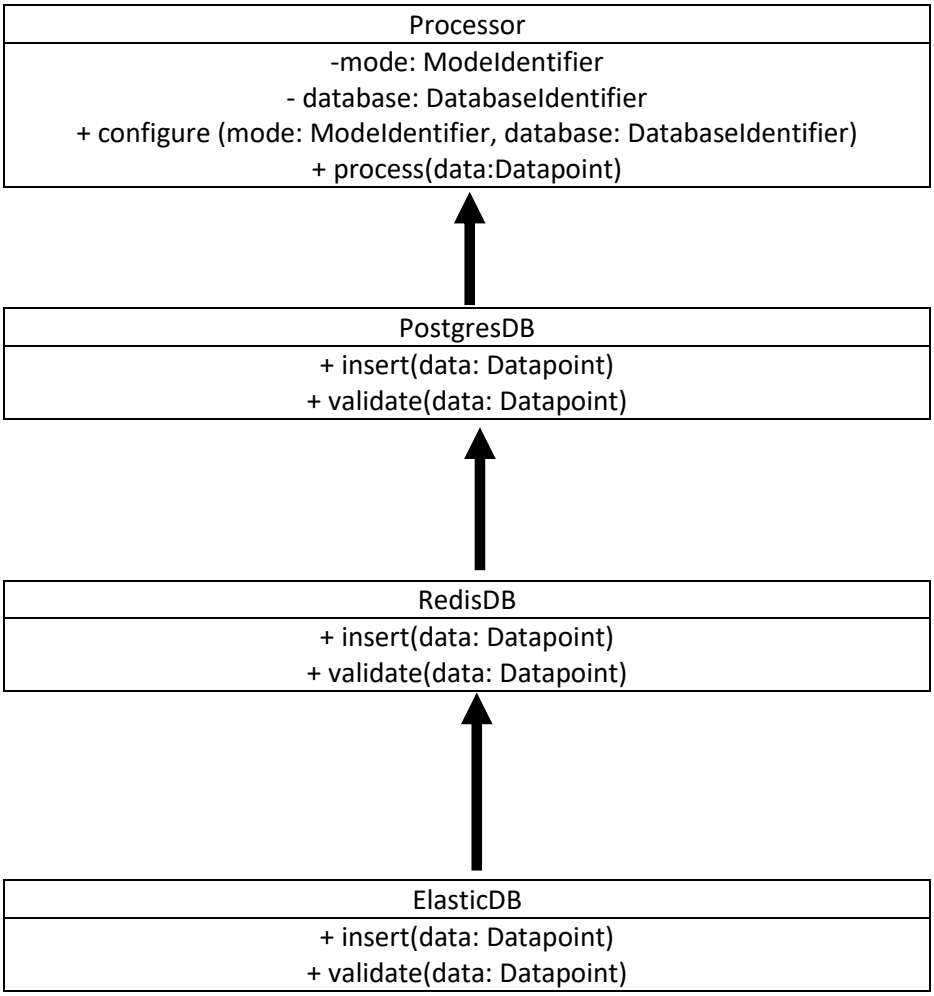


In this design, the DataProcessor class is responsible for processing data points. It has a mode attribute that represents the current operating mode, and a database attribute that represents the current database connection. The configure method allows the user to change the mode and/or the database connection. The process method processes a given DataPoint object, depending on the current mode and database configuration.

The ModeIdentifier and DatabaseIdentifier enums represent the possible values for the mode and database attributes, respectively. The DataPoint class represents both raw and processed data points, and can be used to pass data between methods.

This design allows the data processor to change between the required modes and databases, and to perform different operations depending on the current configuration.

Based on the given requirements, a possible UML class diagram for the data processors pipeline is shown below:



The Processor class represents the main data processor that is responsible for managing the operating mode and database connection. It has an attribute mode of type ModeIdentifier to store the current mode, and another attribute database of type DatabaseIdentifier to store the currently configured database. The Processor class also has a configure method that allows the user to change the operating mode and/or select the current database.

The Processor class also has a process method that takes a Datapoint as a parameter and processes it according to the currently configured mode and database. The behavior of this method will depend on the values of the mode and database attributes.

The PostgresDB, RedisDB, and ElasticDB classes represent the different databases that the processor can connect to. Each of these classes has an insert method and a validate method that take a Datapoint as a parameter and perform the corresponding operation on the data.

This design allows the processor to switch between the different operating modes and databases as needed, while also providing a simple interface for inserting and validating data in each of the different databases.