PROJECT
MET CS 665

SENSOR DATA PROCESSING SYSTEM

Doyle B. Johnson
doybjohn@bu.edu

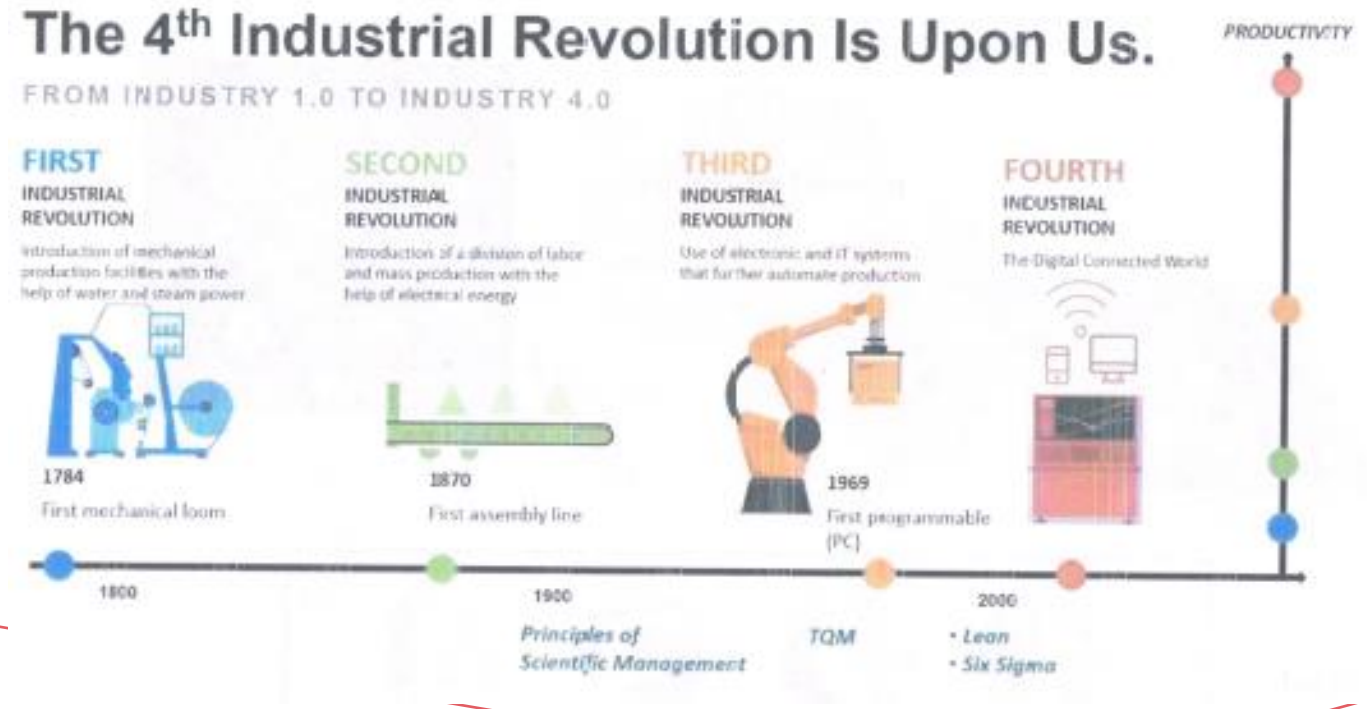
BOSTON UNIVERSITY

TABLE OF CONTENTS

BOSTON UNIVERSITY

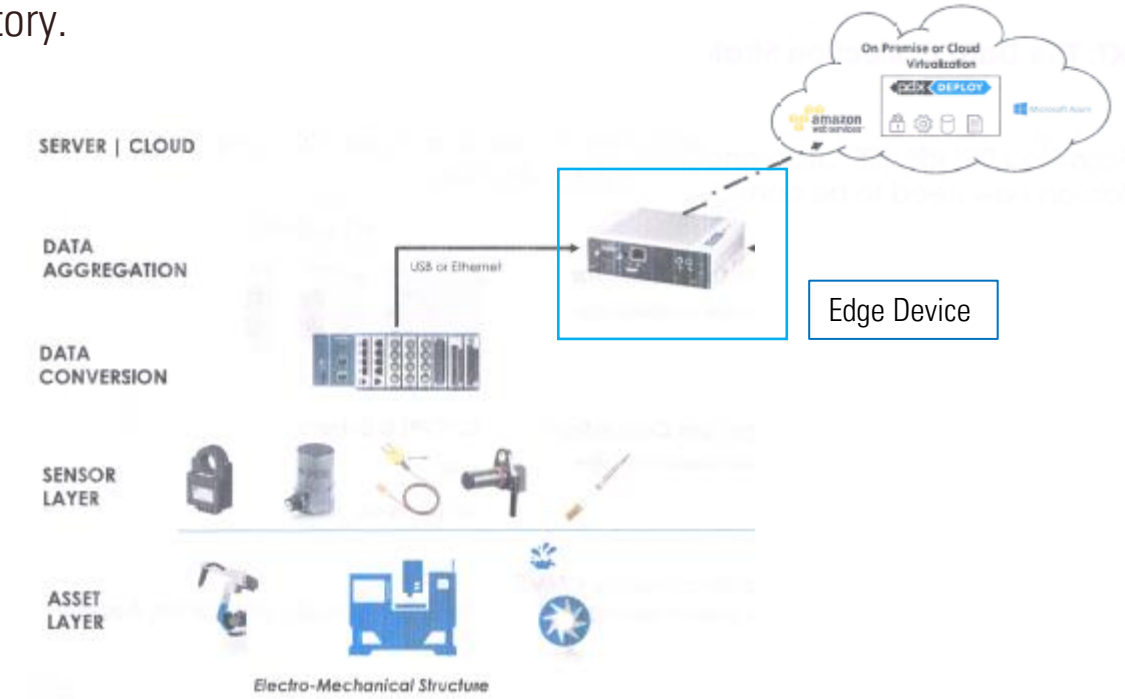
Project Introduction

- We are currently in the "Fourth Industrial Revolution." This stage of industrial transformation is characterized by what is being called "Smart Factories." That is, sensors and data processing components are being incorporated into the factory machines and factory operations to provide near real-time data on the current state (health, quality) of the factory environment. Even more significantly, AI algorithms are being used to analyze the data in near real-time to predict when equipment may fail, or when product quality may go out of specification.



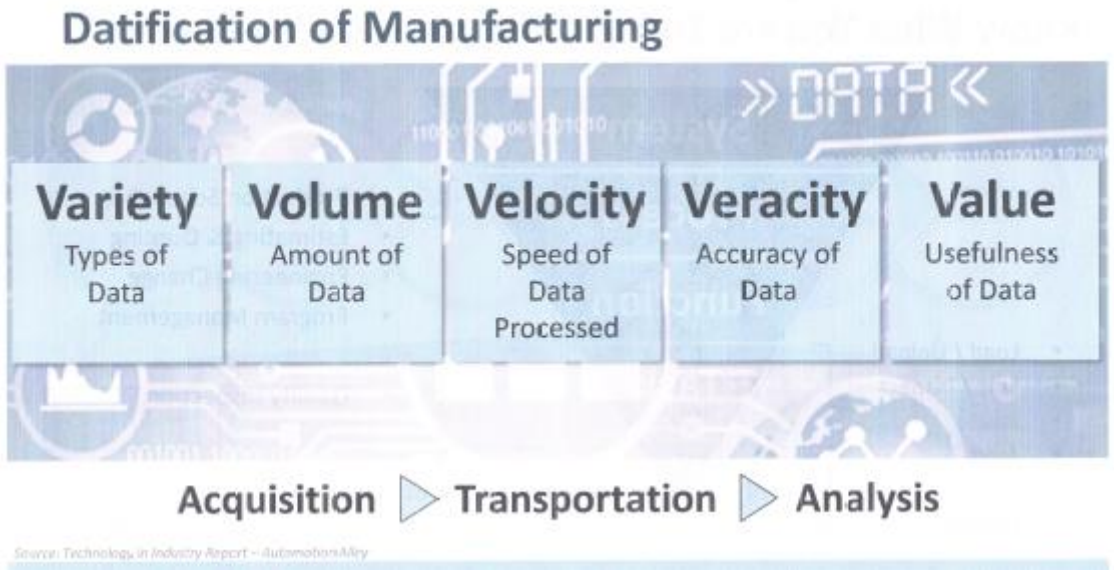
Business Use Case

- As described, modern Smart Factories produce an incredible amount of raw sensor and system data. This data needs to be processed, analyzed and stored in an efficient manner.
- With so much data coming from the factory floor, ideally the data is pre-processed with an "edge" device before sending the data to a data repository.

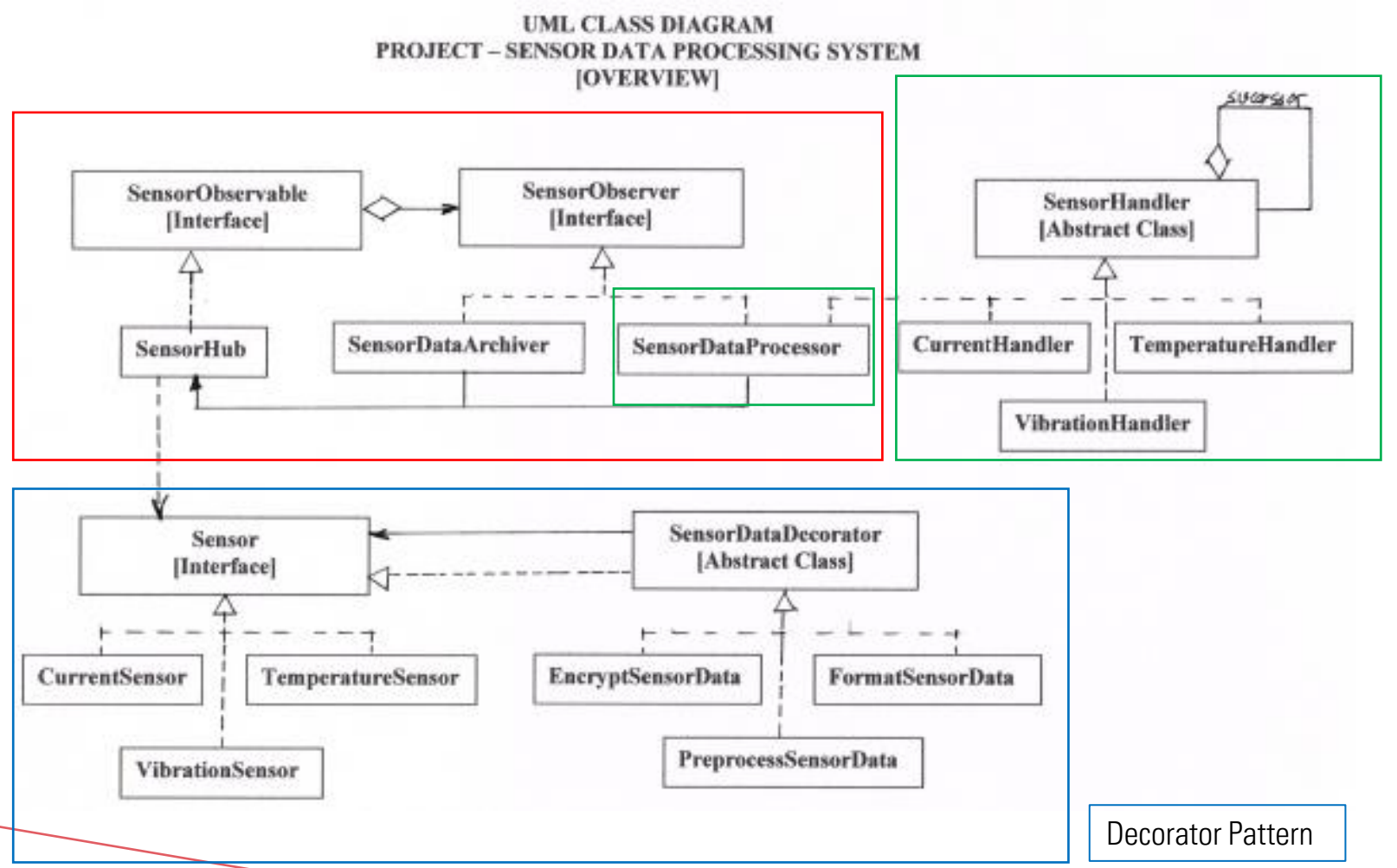


Business Use Case

- Additionally, some of the data needs to be immediately and continually processed with sophisticated AI algorithms to determine machine health and product quality. The data also needs to be stored in a data archive for future analysis and algorithm improvement.
- In this project, I implement a sensor data processing system that forwards the sensor data from a Sensor Hub to the appropriate recipients (Observer Pattern). In addition, some data pre-processing can be applied to the data (Decorator Pattern). Finally, one of the data recipients can process the data in near real-time with AI algorithms. However, each type of sensor data requires different processing, so a specific data handler processes each different type of data (Chain of Responsibility Pattern).



Design Overview



Observer Pattern

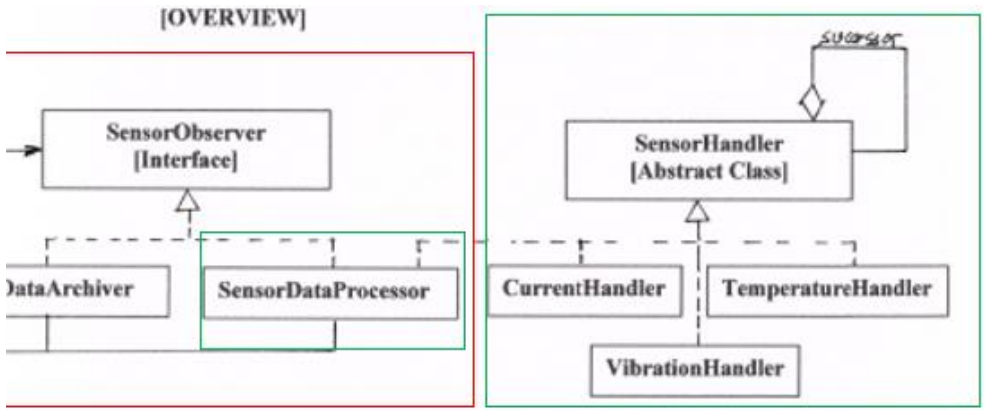
Chain of Responsibility Pattern

Decorator Pattern

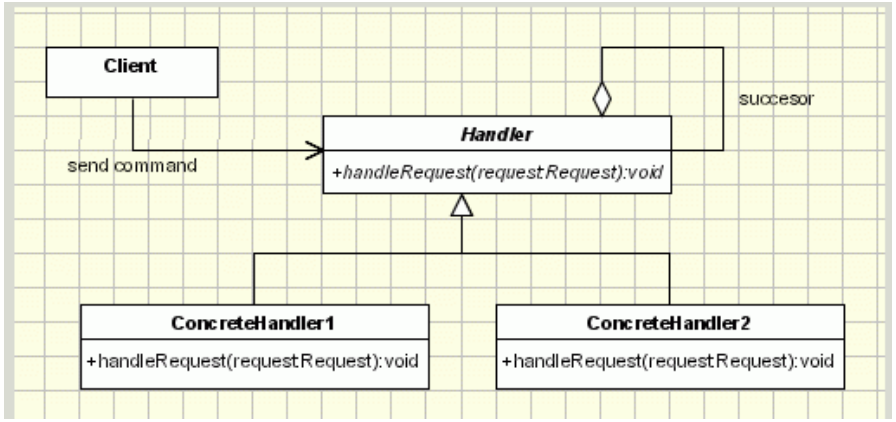
Discussion of Design Patterns

- In the present implementation, the Sensor Hub acts as an "edge device" to receive and process local sensor data. The data can be formatted, pre-processed, encrypted, etc. using a Decorator Pattern to apply the requisite processing. The Sensor Hub is also an "Obseverable" in an Obsever Pattern. Remote processing modules that wish to receive the sensor data can register with the Sensor Hub.

- In this project, there are two Observers: SensorDataArchiver and SensorDataProcessor. The SensorDataArchiver merely receives and archives all the sensor data for later analysis. The SensorDataProcessor, however, also implements a Chain of Responsibility Pattern in order to process each type of sensor data separately. The SensorDataProcessor launches the processing chain after receiving the sensor data from the Sesnor Hub.



Chain of Responsibility Pattern

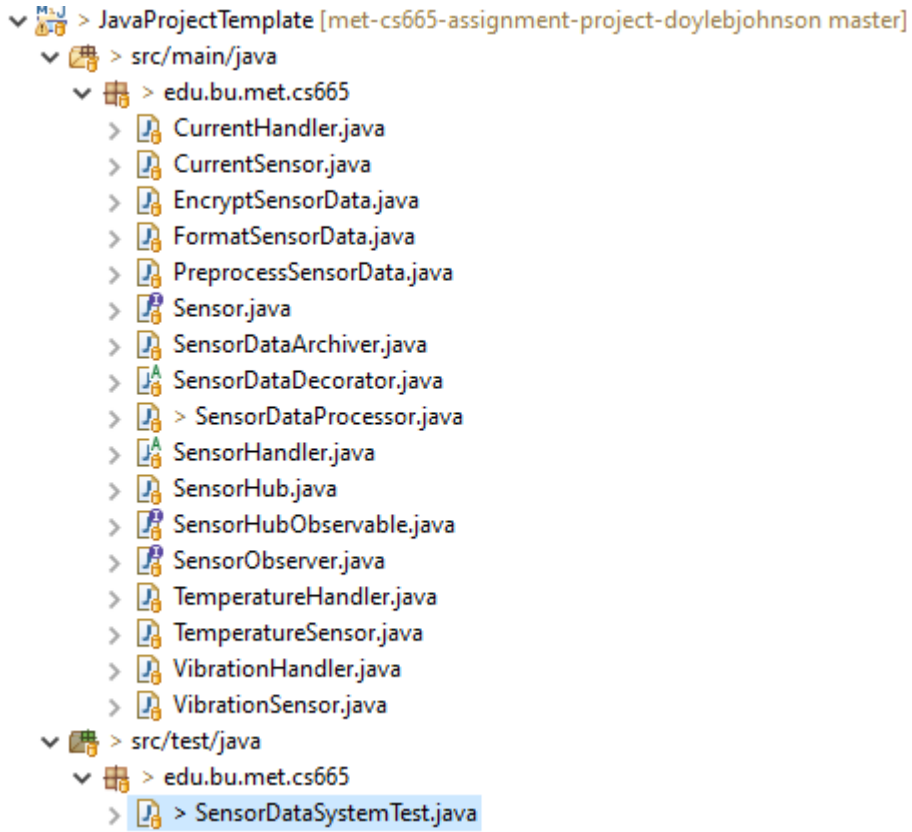


- The generic UML for the Chain of Responsibility Pattern is shown above. A client sends a data or request object. The "Handler" interface defines a "handleRequest" method and can have a method to set the order of the processing chain.
- Each concrete handler has a "nextHandler" attribute that is set so each handler knows where to pass the request to next. When a request is received, the first handler in the chain analyzes the request to determine whether it can handle the request or not. If it can, it processes the request. If not, the request is then passed to the next handler in the chain.
- In some scenarios, it is possible that no appropriate handler is found. In these cases, the request can simply be dropped, or handled by a "catch-all" handler at the end.

Implementation

BOSTON UNIVERSITY

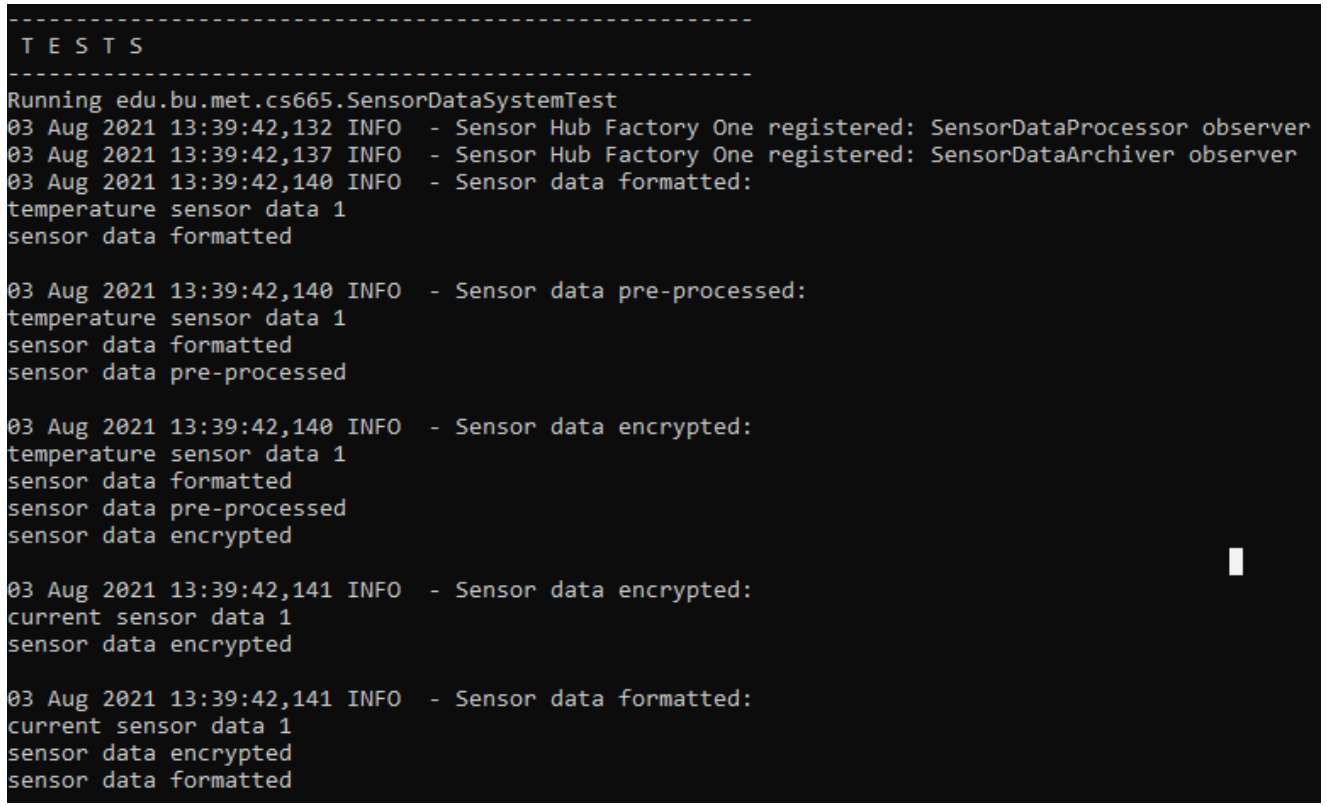
- I will now discuss the code implementation and demonstrate the operation of the program.



Test Results

BOSTON UNIVERSITY

- The log4j output illustrates the operation of the system. Initially, the SensorHub is instantiated, the observers are registered, and the sensor data is formatted as specified in the unit tests:



Test Results

BOSTON UNIVERSITY

- Next, the observers are notified and receive the sensor data. The SensorDataArchiver merely archives the data, but the SensorDataProcessor begins the Chain of Responsibility Processing:

```
03 Aug 2021 13:39:42,141 INFO  - Sensor Data Processor has received new sensor data.
03 Aug 2021 13:39:42,141 INFO  - Beginning processing of sensor data through chain of responsibility.
03 Aug 2021 13:39:42,141 INFO  - Processing vibration sensor data.
03 Aug 2021 13:39:42,141 INFO  - Sensor observer: SensorDataProcessor observer has been notified by Sensor Hub Factory One
03 Aug 2021 13:39:42,141 INFO  - Sensor Data Archiver has received new sensor data.  The sensor data will now be archived.
03 Aug 2021 13:39:42,141 INFO  - Sensor observer: SensorDataArchiver observer has been notified by Sensor Hub Factory One
03 Aug 2021 13:39:42,142 INFO  - Sensor Data Processor has received new sensor data.
03 Aug 2021 13:39:42,142 INFO  - Beginning processing of sensor data through chain of responsibility.
03 Aug 2021 13:39:42,142 INFO  - Not vibration sensor data.
03 Aug 2021 13:39:42,142 INFO  - Processing current sensor data.
03 Aug 2021 13:39:42,142 INFO  - Sensor observer: SensorDataProcessor observer has been notified by Sensor Hub Factory One
03 Aug 2021 13:39:42,142 INFO  - Sensor Data Archiver has received new sensor data.  The sensor data will now be archived.
03 Aug 2021 13:39:42,142 INFO  - Sensor observer: SensorDataArchiver observer has been notified by Sensor Hub Factory One
03 Aug 2021 13:39:42,142 INFO  - Sensor Data Processor has received new sensor data.
03 Aug 2021 13:39:42,142 INFO  - Beginning processing of sensor data through chain of responsibility.
03 Aug 2021 13:39:42,143 INFO  - Not vibration sensor data.
03 Aug 2021 13:39:42,143 INFO  - Not current sensor data.
03 Aug 2021 13:39:42,143 INFO  - Processing temperature sensor data.
03 Aug 2021 13:39:42,143 INFO  - Sensor observer: SensorDataProcessor observer has been notified by Sensor Hub Factory One
03 Aug 2021 13:39:42,143 INFO  - Sensor Data Archiver has received new sensor data.  The sensor data will now be archived.
03 Aug 2021 13:39:42,143 INFO  - Sensor observer: SensorDataArchiver observer has been notified by Sensor Hub Factory One
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.372 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

Conclusion

BOSTON UNIVERSITY

- The Chain of Responsibility Pattern is a useful design pattern for situations where a client request may require different types of processing.
- A particular advantage is that the sender of the request does not need to know anything about which receiver can process the request.
- The Chain of Responsibility Pattern can conveniently be combined with the Observer Pattern by having one object be both an Observer and a Handler.
- The processing chain is easily extended, since each handler only needs to know the name of the next handler. In addition, there is no need to add additional code to a "central" handler every time a new processing module is added.

Bibliography

BOSTON UNIVERSITY

- Mo Abuali, Lecture Notes, Industry 4.0 Domains of Knowledge, Univ. Of Cincinnati, 2020.
- Jay Lee, Industrial AI: Applications with Sustainable Performance, Shanghai Jiao Tong University Press, 2020.
- Chain of Responsibility Pattern, https://www.oodesign.com/chain-of-responsibility-pattern.html.
- Chain of Responsibility Pattern, http://www.avajava.com/tutorials/lessons/chain-of-responsibility-pattern.html.
- Chain of Responsibility Pattern, https://www.tutorialspoint.com/design_pattern/chain_of_responsibility_pattern.htm.