DATA ACCESS LAYER SYSTEM UML CLASS DIAGRAM

The provided UML Class Diagram illustrates the architecture of a data processing system designed to collect, parse, fuse and adapt data from various sources. The core components include specialized data listeners, a parser, a fusion model, and an adaptable output strategy. The design ensures modularity, scalability, and maintainability, allowing easy integration of new data sources and output methods. The following sections provide an overview of each class and the rationale behind their design choices. This system has structured around four main components: data listeners, data fusion, data parser and output strategy.

Data Collection

The “DataListener” abstract class serves as the foundation for different types of data listeners, including “ConsoleListener”, “FileDataListener”, “WebSocketDataListener”, and “TCPDataListener”. This abstraction allows the system to handle different kind of data sources interchangeably, promoting scalability and scalability.

Data Parsing

The “DataParser” class is responsible for converting raw data collected by listeners into structured “PatientData”. This centralizes the parsing logic, making it easier to manage and modify as needed.

Data Fusion and Analysis

The “DataFusion” class aggregates and analyzes data from multiple sources. It connects different data components and provides methods to fuse and analyze the collected data such as connect(), fuse(), and analyze(), ensuring comprehensive data processing.

Data Output

The “DataSourceAdapter” class adapts the fused data for output, utilizing an interface named as “OutputStrategy”. This design allows for interchangeable and flexible output methods, supporting different formats or destinations.

Conclusion

The UML Class diagram for data access layer system features modularity, scalability, and maintainability. It effectively collects, parses, fuses and adapts data from various sources. This system easily integrates new data sources and output methods, providing robustness and flexibility, which makes it an efficient solution for diverse data processing needs.