Design Decisions

The system architecture follows a Model-View-Controller (MVC) pattern, ensuring a clear separation of concerns between data management, user interface, and application logic. The Repository design pattern is utilized for managing data, allowing for easy access and manipulation of insurance data stored in JSON files. This approach simplifies data handling and enhances maintainability by avoiding complex database management.

Data Storage Format

The data is stored in JSON files, providing a lightweight and flexible format for managing the insurance data. The use of JSON facilitates easy reading and writing operations, allowing the application to perform CRUD operations seamlessly without the overhead of a traditional database management system.

Task 5: Implement Claims Management

1. Implemented the Repository design pattern to store and manage all insurance data, including claims.
2. Developed servlets for brokers to:
    - Submit new claims.
    - Update claims.
    - Approve/reject claims.
    - View the status of claims (approved, pending, rejected).

Task 6: Implement Reporting Feature

1. Developed a reporting servlet that summarizes:
    - The number of customers managed by each broker.
    - The total number of policies and claims processed.
    - The approval and rejection rates of claims.
2. Ensured that the reporting feature reads data from the repository and generates an aggregated summary in a user-friendly format.

Task 7: Thread-Safe Resource Management

1. Modified all servlets to ensure that shared resources are accessed in a thread-safe manner.
2. Employed synchronized blocks or concurrent data structures to prevent race conditions during simultaneous access.
3. Simulated concurrent access using multiple browser windows and testing tools such as Apache JMeter.

Task 8: Servlet Configuration Using Deployment Descriptor

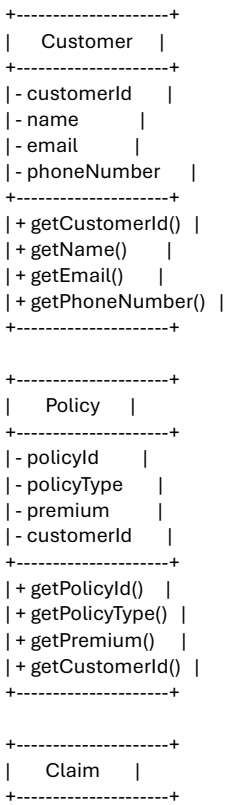
1. Mapped all servlets using the web.xml deployment descriptor only, avoiding annotations for servlet mapping.
2. Ensured that all servlets are properly defined and mapped in the web.xml file, including initialization parameters where necessary.

Collaboration Details

The group collaborated effectively using GitHub to manage the project repository. Tasks were divided among team members, allowing for a balanced workload. Regular meetings were held to discuss progress and address any challenges. GitHub was utilized for version control, enabling team members to track changes, review code, and merge contributions seamlessly.

Documentation and Diagrams

The report includes UML diagrams that illustrate the system's architecture and class relationships, as well as sequence diagrams for key processes such as claims management and reporting. Screenshots are provided to showcase the system in action, highlighting the user interface for customer, policy, and claim management.



Conclusion

The development of the Insurance Broker Management System has provided a comprehensive understanding of Java Servlets, the Repository design pattern, and thread-safe programming practices. The project has enhanced collaboration skills through effective use of GitHub and has fostered an appreciation for proper documentation and design in software development.