Project Approach and Prioritization

Objective:

To implement and fix a congestion tax calculator for Gothenburg, ensure its functionality, and make it adaptable for use in other cities with different tax rules.

Time Allocation:

Given the time constraint of 4 hours, I prioritized the following:

1. **Core Functionality:**
    a. **Primary Focus**: Ensured that the congestion tax calculation logic worked correctly for the specified scope (Gothenburg, 2013).
    b. Implemented the tax calculation method, including handling:
        i. Time-based tax rates for different periods of the day.
        ii. Daily tax limit of 60 SEK.
        iii. Weekends, public holidays, days before public holidays, and July exemptions.
        iv. Vehicle types that are tax-exempt (e.g., emergency vehicles, buses, motorcycles).
    c. **Rationale**: These are the core requirements of the application, and ensuring correctness here was the most critical component.
2. **Entry Point for the Application:**
    a. Added an HTTP entry point using a REST controller to make the application callable with vehicle information and timestamps via API.
    b. **Rationale**: The application needed an interface through which the calculation can be triggered. This also makes it extensible for future use cases.
3. **Handling Exemptions and Year Limitation:**
    a. Focused on restricting calculations to the year 2013 and incorporated checks for tax-exempt vehicles.
    b. **Rationale**: The year-based scope was a hard requirement, and handling tax exemptions is fundamental to ensure accurate results.
4. **Externalizing Configuration:**
    a. Moved city-specific parameters (like time-based tax rates and exemptions) to an external JSON configuration file.

    b. **Rationale**: This allows future scalability to other cities, as requested by the bonus scenario, without hardcoding city-specific rules into the codebase.

5. **Testing:**
    a. Wrote unit tests to validate core functionality (e.g., correct tax calculation, vehicle exemptions, date-based logic).
    b. **Rationale**: Testing ensures that the core logic works as expected and provides a baseline for future feature additions or modifications.

What Was Left Out (Due to Time Constraints):

1. **Detailed Exception Handling:**
    a. **Reason**: I did basic error handling for now but would enhance it with more specific exceptions and logging for production readiness.
2. **Further Optimization for Multiple Cities:**
    a. While I moved configuration parameters (e.g., tax rules) to a JSON file, I did not fully implement a system that could easily handle multiple city rules in parallel (for example, a database solution or a configuration management system).
    b. **Future Work**: Implement a more dynamic system for cities, with separate configuration files or database entries for each, and a service that dynamically picks the correct configuration at runtime.
3. **Comprehensive Integration Tests:**
    a. Focused primarily on unit testing due to time constraints. With more time, I would add integration tests to validate the entire flow, especially with the REST API.
    b. **Future Work**: Integration tests to verify the interaction between the controller, service, and external configuration.
4. **Deployment and Dockerization:**
    a. Due to the time limit, I did not focus on deploying the application or containerizing it with Docker.
    b. **Future Work**: Containerization with Docker, followed by setting up CI/CD pipelines for automated deployment.

Additional Work for Future Consideration:

1. **Performance Optimization:**
   a. With more time, I would focus on performance optimization, especially regarding high-traffic environments where many vehicles are taxed simultaneously.
2. **Handling Edge Cases:**
   a. Incorporate edge cases like handling invalid timestamps or unexpected vehicle types more robustly.
3. **UI Interface:**
   a. With more time, a simple front-end interface for non-technical users could be added to make interacting with the tax calculation service easier.
4. **Multi-Tenant Support:**
   a. Implement support for multi-tenancy to easily manage tax rules for different cities dynamically, possibly by allowing editors to update the configuration through a user interface.

Conclusion:

In the allotted 4 hours, I focused on the core functionality and building a scalable foundation for future extension (e.g., using external configuration for city-specific rules). Given more time, I would have improved error handling, integration testing, API documentation, and added multi-city dynamic rule support.