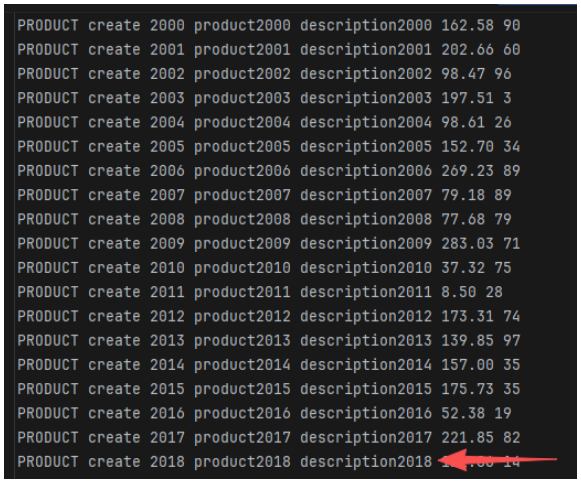
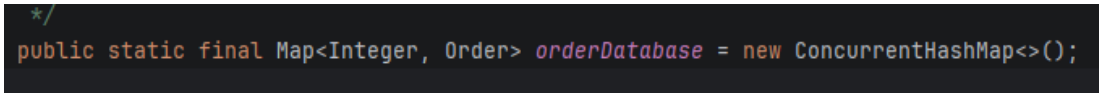
1: Generative AI and Modifications

- **Usage 1**: In this submission, Generative AI was used solely to create test data. We did not use Generative AI to write any code. We used it to brainstorm scenarios for the workload.txt file.
- **Issues and Modifications:**
    - **Issue 1**: We found that the AI lacked context on our specific API requirements. Specifically, AI consistently omitted the mandatory 'description' field for product-related commands. My group had to manually refactor these test cases to match our implementation's expected input format.
    - 

    - **Issue 2:** In addition, the AI struggled with the specific syntax of our update protocols. For example, when generating UserService update commands, the AI failed to include the required colon delimiter (:) between keys and values (e.g. email:test@test.com). Because our simple parsing logic relies on this specific delimiter to distinguish fields, we had to manually check and correct formatting errors across the workload file to ensure the parser could understand the input.

2: Technical Debt and Scalability:

- **Concurrency and Thread Safety**:
    - **A1:** Although our services use a FixedThreadPoll and a ConcurrentHashMap, we have taken shortcuts in handling complex transaction integrity. In A1, we assumed sequential workloads and didn't need to serve multiple requests simultaneously. However, our current logic when checking product stock before placing an order is not atomic. In a high-concurrency environment, this would cause problems.
    - **A2:** For A2, this will require a significant rewrite to implement atomic operations to ensure the Concurrency and Thread Safety

- **Data Persistence:**
  - **A1:** For Assignment 1, we used an in-memory storage strategy with a ConcurrentHashMap to act as a mock database. While this successfully facilitates inter-service communication and passes all current test cases, it lacks data persistence.
  - **A2:** For Assignment 2, we recognize that this in-memory fake database must be replaced with a persistent storage solution, such as a relational database. The relational database can help to ensure that the system state survives interruptions and restarts.
  - 

3: Codes that require few modifications:

- The data processing part of the UserHandler, ProductHandler, and OrderHandler can be reused. The only modification would be to read data from the database instead of from the User, Product, and Order classes.
- ConfigReader: Most of the code in ConfigReader can be reused. Since this class is used to read the configuration, if the configuration file format for A2 does not change much, most of the code can be reused.
- ISCS and ISCS handler: ISCS acts as a middle layer, connecting OrderService with ProductService/UserService. Since A2 is still using the microservice architecture, I don't think we need to rewrite the ISCS part.

4: The reason why I did not use AI to generate any code

- **Alignment with Course Materials:** The lecture examples provided a comprehensive foundation for RESTful APIs and microservice architecture. Since these resources were specifically tailored to the requirements of Assignments, we found it more efficient and reliable to adapt the provided course code than to prompt an AI for generic solutions that might deviate from the intended learning outcomes.
- **Technical Reliability and Precision:** Our experience with AI-generated test cases demonstrated that the model often lacks the specific context of our system's protocol. The AI consistently produced incorrect test data, such as omitting 'description' fields and colon delimiters, which required tedious manual correction. We concluded that relying on AI for core logic would likely introduce subtle bugs that would take longer to debug than writing the code manually.
- **Commitment to learning objectives:** We view this assignment as a critical opportunity to practice the skills that we learned in lectures. Relying on AI to solve engineering challenges would bypass the problem-solving process, which defeats the purpose of the exercise. By manually implementing the handlers and data models, we ensured a deep understanding of the system. The experience was essential for the increased complexity expected in Assignment 2.