Description: OnlineShop.com has hired you as a backend lead to work on the new website called OnlineShopping.com, which is similar to <https://www.flipkart.com/>. The frontend development has been outsourced, while the backend development will be done in-house.

In the current sprint, which is working towards the MVP version of the product, you are asked to create APIs for customers to view products in different categories. In addition to this admins will be able to add, or remove products, and update product prices and/or inventory at any time.

Use Case:

1. As a customer, I want to view products in a category and have them arranged on the page based on inventory availability or price that I select.
2. As an admin, I want to be able to add, and remove product, or update product prices and/or inventory at any time.

Business Criteria:

1. The category page should not display products if their inventory is unavailable or limited.
2. If there are no products available with sufficient inventory, an error message should be displayed on the category page.
3. If a non-existent category is requested, an error message is to be sent back

Technical Criteria:

1. The sample json provided for the application has price and inventory populated in the product listing. Make sure that price and inventory are two separate subdomain.
2. Any service to service communication should use mechanism to avoid cascading failures. Make sure three retries are performed and if data is not available, NAN is populated as value.
3. The application should be demonstrated using Swagger in a containerized environment, with a DB running in another containerized environment.
4. The code should be of production quality, with proper unit test coverage (at least 80%).
5. JUnit 5 and Mockito should be used for test cases, avoiding integration test cases as much as possible.
6. Modularity is required for easy maintenance, and logging/tracing for operation support.
7. Proper exception handling with standard HTTP status codes and Swagger documentation is expected.
8. API best practices should be followed
9. Lombok should be used to reduce boilerplate code. Libraries like ModelMapper can be used for copying object data.
10. The system should be demonstrated with an API gateway and service discovery.
11. Relevant data models and entity relationships should be designed and added to the DB. Any DB type (SQL or NoSQL), including in-memory databases, can be used for the current implementation. The chosen DB type should be justified. Sample data related to two categories are provided.
12. While updating the category, validation should be performed for price format, at least one attribute availability, and mandatory fields of name, brand, and description.
13. The response format should follow a common response format for success and errors as well with including trace-id in all the responses.

Sample Category data is available in the same folder. This can be modified, if needed