Instructions on how to run the application.

1. Download the jar file provided.
2. Make sure java 17 is installed on the system (https://www.java.com/en/download/help/download_options.html)
3. On terminal - Navigate to the path where jar file is located.
4. Run command: java -jar UsageTranslator-1.0-SNAPSHOT.jar

Printed Expected Output:

```
sneha@Snehas-Laptop Desktop %
sneha@Snehas-Laptop Desktop % java -jar UsageTranslator-1.0-SNAPSHOT.jar
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
Successfully executed Insert Query for Chargeable: INSERT INTO chargeable (partnerID, product, partnerPurchasedPlanID, plan, `usage`) VALUES (?, ?, ?, ?, ?)
Successfully executed Insert Query for Domain: INSERT INTO domain (partnerPurchasedPlanID, domain) VALUES (?,?)
sneha@Snehas-Laptop Desktop %
```

Best Practices followed:

1. Application successfully validates and does sanity checks of the input data as per the mentioned constraints.

2. Modular approach: The classes are segregated to follow Single responsibility approach. This makes the code readable and understandable.

   Below are the responsibilities of the classes.

```
org.usageTranslator.model.Chargeable
        Model that holds all the properties of Chargeable entity.

org.usageTranslator.model.Domain
        Model that holds all the properties of Domain entity.


org.usageTranslator.dao.ChargeableDao
        Responsible for manipulating the Chargeable table in db

org.usageTranslator.dao.DomainDao
        Responsible for manipulating the Domain table in db

org.usageTranslator.service.ChargeableService
        Responsible for the validation of the input data, and business
    logic associated with the Chargeable table.

org.usageTranslator.service.DomainService
        Responsible for the validation of the input data, and business
    logic associated with the Domain table.

org.usageTranslator.util.InputCSVReaderUtil
        Responsible for the reading the input CSV data file.

org.usageTranslator.util.InputJSONReaderUtil
        Responsible for the reading the input JSON data file.


org.usageTranslator.validator.InputDataValidator
```

```
   Responsible for the validating the input data as per the
constraints in the requirements.
```

3. MySQL – Use of ‘PreparedStatement’ which protects against **SQL injection** attacks when interacting with a MySQL database using JDBC.
4. We iterate through each entry in the data file and store them in the data structures like Maps, Lists. Then each entry is processed and inserted in the database.

   Thus, we get the linear **time and space complexity** (O (n) where n = size of input data) which is directly proportional to the size of the input data (CSV and JSON data).

Next steps:

1. Storing **error messages** in the constant files.
2. Implementing the **Unit testing**.
3. Extending the functionality to implement **CRUD** operations of Chargeable and Domain table.
4. For performance, **Connection pooling** library like HikariCP can be configured. It facilitates reusability of pooled connections, that will enhance the efficiency.