Metrics

APIs:

1. Create a Metric

This API allows the user to create multiple metrics in 1 call by passing the names as list in the body.

Usage:

Http request type: POST
http://localhost:8080/metrics/create
Header: Content-Type = application/json
Body:

```
{
        "names":["metricName1","metricName2"]
}
```

2. Post Values to a Metric

This API allows the user to send the value of each metric in the message body. Response returned contains list of all the metric names for which values have been added.

Usage:

Http request type: POST
http://localhost:8080/metrics/values

Header: Content-Type = application/json
Body:

```
[
{
        "name":"metricName1",
        "value": 10.5
},
{
        "name":"metricName2",
        "value": 7.816
}
]
```

3. Retrieve Statistics

This API allows the user to retrieve the min, max, mean and median values for a particular metric by passing the metric name as a parameter

Usage:

Http request type: GET
http://localhost:8080/metrics/stats?name=metricName1

Time and Space complexity analysis:

1. Create a Metric :
   Time complexity for each metric name:
   The names are added to a map which takes O(1) time.
2. Post Values to a Metric:
   Time complexity for each metric value:
   The metric values are stored in a list and each new value is appended to the end of the list and also added to a heap : O(log n) time where n is the number of metric values added so far.
3. Retrieve Statistics:
   Time complexity for each metric:
   The metric values are precomputed and values can be retrieved immediately from the hash map: O(1) time.

At any point, for each metric, the space needed is O(n)
Where n is the number of metrics added so far since the metric values stored in an Arraylist and is added to either min heap or max heap.

Assumptions:

1. If user tries to create an already existing metric, no action is taken and existing metrics along with it's values won't change.
2. If the user tries to add values to a non existing metric, the values get ignored and no new metric is created.
3. Special characters such as %,/,\.etc are not going to be used for the metric name.('.', '-','_' are all usable)

How to run the application:

The application can be run as a standalone on the machine:

- $cd <location of the project directory>
- $./gradlew clean build
- $cd build/libs
- $java -jar *.jar

The application can also be run as on docker:

- $cd <location of the project directory>
- $./gradlew build buildDocker
- $docker run -p 8080:8080 flexengage/metrics:latest