Java and Java Automation Developer’s Test Assignemnt

Assignment
The goal of the assignment is to create a tool, that provides statistic aggregation. The tool is listening to input from other servers and will aggregate and store the information for later usage. Tool will use custom plain text protocol to get the data. Restful API will be provided to access the data. More details below.

Restrictions and Help
You will be provided a sample project, that has an example service „hello world“ implemented. Usage of external libraries is restricted. Only existing libraries/dependencies can be used. Everything else should be implemented in the project.
To run provided package do following:

1. Install maven from http://maven.apache.org/. Add maven to path etc.
2. Unzip pt-summer-2016.zip.
3. Step into pt-summer-2016 folder and execute „mvn clean install“.
4. To run, execute „mvn jetty:run“. This starts jetty webserver and your application in it.
5. To access application go to url: http://localhost:9090/sample/helloWorld/YOUR_INPUT_HERE

Input Protocol
To listen the information separate port is opened, where plain text protocol is used. Format is „`<metric path> <metric value> <timestamp>`“. Metric path is a string. For example: „`sitename.host.javaserver.brand.activity.login.reponse`“. Metric value is a number and metric timestamp is timestamp in milliseconds. Example of full message: „`p local.random.diceroll 321 1461140237`“. Every metric value is sent in a separate connection.
Example, how to send message like that in linux command line:

```
        PORT=2003
        SERVER=localhost
        echo "local.random.diceroll 4 `date +%s`" | nc -q0 ${SERVER} ${PORT}
```

There are also other ways like telnet or write your own custom program.

Data aggregation
Application is getting raw data. In any second application can receive thousands of values. Application has to aggregate values per second in memory. Following values need to be calculated:

- 1SecondAvg – 1 second average.
- 1SecondMax – 1 second maximum.
- 1MinuteAvg – 1 minute average.
- 1MinuteMax – 1 minute maximum.

Calculated values are stored into files. Every aggregated metric has its own separate file. For example „`local.random.diceroll`“ will be written into following files:

- `local/random/diceroll.1SecondAvg`
- `local/random/diceroll.1SecondMax`
- `local/random/diceroll.1MinuteAvg`
- `local/random/diceroll.1MinuteMax`

File is formatted in text format. Every line contains two values – timestamp and value. For example:

```
1461141000 123
1461142000 134
1461143000 101
```

Data Query API

This is a restful API to get aggregated data. Request is json and response is json. Stored files will be used to collect the information, that needs to be returned.

- /aggr/query
    - Request parameters:
        - pattern=<string>, mandatory – Regular expression of metric name.
        - start=<timestamp>, mandatory –
        - end=<timestamp>, mandatory –
    - Response parameters:
        - List of metric with parameters:
            - name=<string>, mandatory – metric name like `local.random.diceroll.1SecondAvg`
            - datapoints=list of arrays, array consists of two elements:
                - First position contains value.
                - Second position contains timestamp.

Examples:
Request:
http://localhost:9090/aggr/query?pattern=local.random.diceroll.1SecondAvg&start=1461162542693&end=1461162682693
Request:
http://localhost:9090/aggr/query?pattern=local.random.*.1SecondAvg&start=1461162542693&end=1461162682693
Result:
[{"name":"local.random.diceroll.1SecondMax","datapoints":[[1461163460175,123],[1461163461175,145],[1461163462175,101]]},{"name":"local.random.diceroll.1SecondAvg","datapoints":[[1461163460175,12],[1461163461175,20],[1461163462175,13]]}]
Result in pretty format:

```
[
    {
        "datapoints": [
            [
                1461163460175,
                123
            ],
            [
                1461163461175,
                145
            ],
            [
                1461163462175,
                101
            ]
        ],
        "name": "local.random.diceroll.1SecondMax"
    },
    {
        "datapoints": [
            [
                1461163460175,
                12
            ],
            [
                1461163461175,
                20
            ],
            [
                1461163462175,
                13
            ]
        ],
        "name": "local.random.diceroll.1SecondAvg"
    }
]
```