Task for candidates

Description

There is a set of source files in json format containing series of measurements for the given devices in the given moment in time. We need application to read the files and discover anomalies in the data. Anomalies detection should be implemented in java and configured via files. Anomalies definition will change in the future and it should be easy to modify source code to add new once. Source files are periodically generated by some external system and stored into well defined directory. Directory for the input files can be different for different deployments. Single file can be bigger then memory available for the application. Application will be executed by external scheduler so for the single execution application should process all available files and exit. In case of failure application should exit with status code different then 0.

Results should be printed out on standard output. Result should contain anomaly name, `deviceId` and `measuredValue` separated by colon for every detected anomaly.

Application should be implemented using spring-batch framework. Dependencies should be defined using maven or gradle. Source code should be placed in bitbucket.org or github.com.

Source file format:

```
[
  {
    "parentId":"DC1",
    "deviceId":"METER11",
    "measuredValue":100,
    "timestamp":"2012-04-23T10:00:00.000Z"
  },
  {
    "parentId":"DC1",
    "deviceId":"METER12",
    "measuredValue":100.1,
    "timestamp":"2012-04-23T10:15:00.000Z"
  },
  {
    "parentId":"DC2",
    "deviceId":"METER21",
    "measuredValue":100.2,
    "timestamp":"2012-04-23T10:00:00.000Z"
  },
  {
    "parentId":"DC2",
    "deviceId":"METER22",
    "measuredValue": 99.99,
    "timestamp":"2012-04-23T10:15:00.000Z"
  }
]
```

Solution should implement ValueToHigh anomaly - detecting measurements for devices assigned to given parent having `measuredValue` higher then configured limit. Example anomaly configuration looks like follows:

```
[
  {
    "parentPattern":"D.*1",
    "limit":100
  },
  {
    "parentPattern":"DC2",
    "limit": 100
  }
]
```

where:

- parentPattern - java RegExp expression. Given limit should be checked against every device having parentId matching the expression.
- limit - limit value to compare with `measuredValue`

For the provided configuration and data application should print:

```
ValueTooHigh,METER11,100.1
ValueTooHigh,METER21,100.2
```