

# NODE STATUS REPORTING

## Problem Definition

Given an input file that depicts the messages received by a monitoring system from n nodes, print the status of the nodes.

## Sample Input

```
1508405807242 1508405807141 vader HELLO
1508405807340 1508405807350 luke HELLO
1508405807378 1508405807387 luke LOST vader
1508405807467 1508405807479 luke FOUND r2d2
1508405807468 1508405807480 luke LOST leia
1508405807512 1508405807400 vader LOST luke
1508405807560 1508405807504 vader HELLO
```

## Sample output

```
vader ALIVE 1508405807560 vader HELLO
luke ALIVE 1508405807468 luke LOST leia
r2d2 ALIVE 1508405807467 luke FOUND r2d2
leia DEAD 1508405807468 luke LOST leia
```

## Constraints

1. The input file is specified as a command line argument

## Assumptions

1. The valid format for a line is  
<received\_timestamp\_at\_node> <emitted\_timestamp\_at\_node> <node\_name>  
<status>
2. The valid list of status is HELLO, LOST, FOUND
3. If no file is specified, or incorrect file specified user defined exception is thrown
4. Valid node status are ALIVE, DEAD, UNKNOWN
5. If conflicting status is emitted at the same time, the node status is set to UNKNOWN

## Development Environment

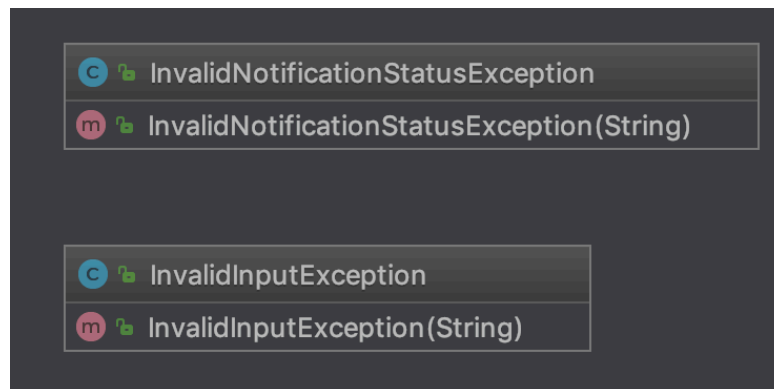
1. Java8, Gradle 4.3, Junit 4

## How to run

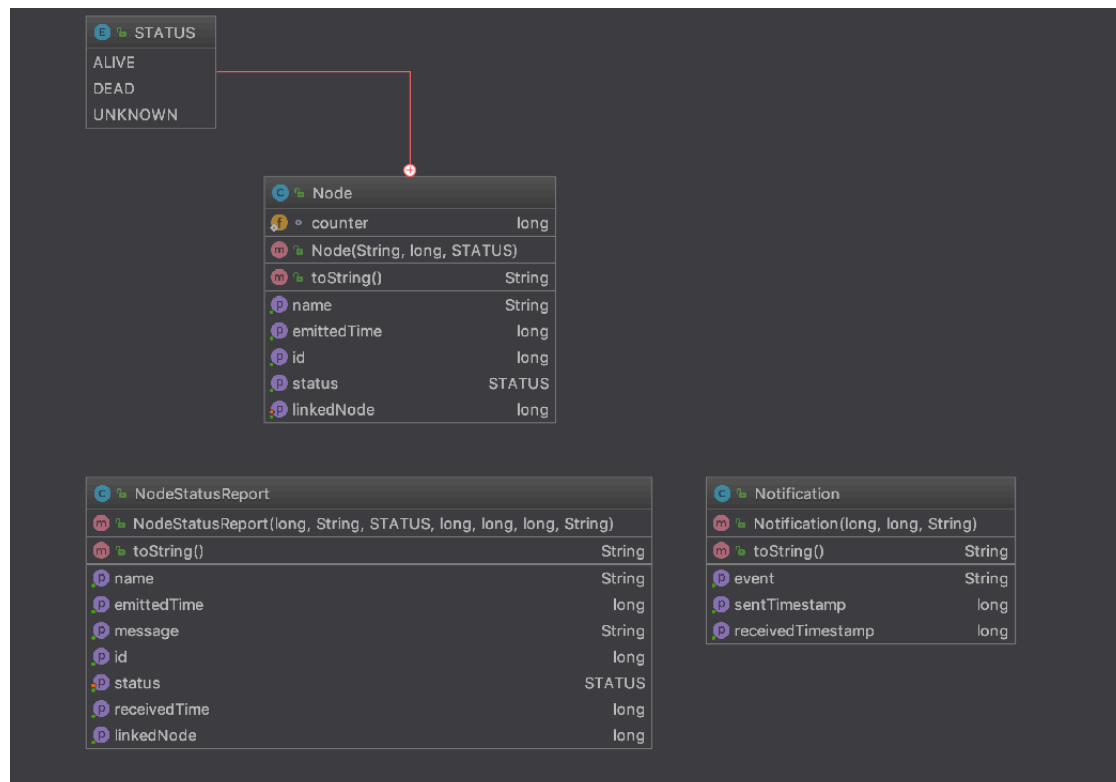
1. Import project from github (master branch) – <https://github.com/ClozerUK/Node-Status-Reporting>
2. Import as a gradle project
3. Set command line argument to match an input file
4. Run Main.java

## Class Diagrams









### Exceptions

















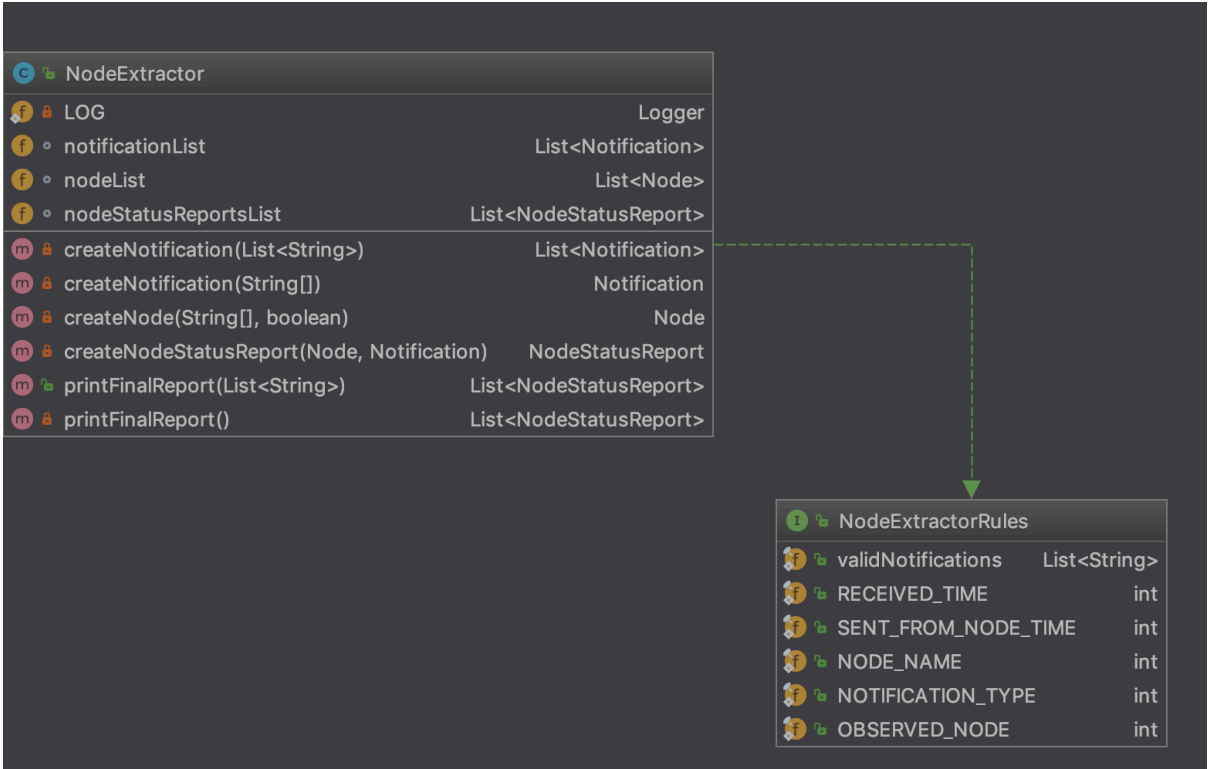
### Model



Service

		InputFileReader	
		LOG	Logger
		readFile(String[])	List<String>
		validFormat(String)	boolean

		NodeExtractorRules	
		validNotifications	List<String>
		RECEIVED_TIME	int
		SENT_FROM_NODE_TIME	int
		NODE_NAME	int
		NOTIFICATION_TYPE	int
		OBSERVED_NODE	int



## Implementation Description

1. READ AND VALIDATE FILE
  - a. Read the input file
  - b. Validate the file -> check for valid file name and format
  - c. Create a list of all messages
2. CREATE NOTIFICATION
  - a. Read the line and create a notification.
  - b. A notification has a received time, sent time, message
3. CREATE NODE
  - a. A node has an auto generated id (implemented as a simple id and not GUID)
  - b. Every node in addition to id has a name, emitted time, status, linked node
  - c. Status is determined as ALIVE if emitted message is HELLO
  - d. If the node has observed another node, then create the observed node
  - e. Set the parent node's status as ALIVE and the observed node as per the message's status
  - f. A node is ALIVE if the parent node has a message 'FOUND'
  - g. A node is DEAD if the parent node has a message 'LOST'
  - h. If the monitoring system receives conflicting status, set all related nodes as UNKNOWN

Eg:

```
1508405807510 1508405807505 vader LOST r2d2
1508405807511 1508405807505 r2d2 FOUND leia
```

In the above example, vader, r2d2, leia are all UNKNOWN as vader implies that r2d2 is dead at 1508405807505 but r2d2 emits at the same time that it has FOUND leia. A node cannot be active and dead at the same time, hence all the nodes related to this could not be known as well. This is an assumption on how UNKNOWN status works.

4. CREATE NOTIFICATION STATUS REPORT
  - a. Once all the nodes are formed, form the notification status report based on the required format
  - b. Print the status