POLITECNICO
MILANO 1863

Politecnico di Milano

COMPUTER SCIENCE AND ENGINEERING

Software Engineering 2

Code Inspection

PowerEnJoy

Authors:
Francesco Fabiani
Jagadesh Manivannan
Niccolò Pozzolini

Professors:
Elisabetta Di Nitto
Luca Mottola

Version 1.0 **February 5th 2017**

Contents

1 Description of the code **2**
1.1 Revision history 2
1.2 Assigned class: TechDataServices 2

2 Results of inspection **4**
2.1 Notation 4
2.2 Report 4
2.2.1 Checklist 4
2.2.2 General 7

Chapter 1

Description of the code

1.1 Revision history

Version	Date	Authors	Summary
1.0	05/02/2017	Fabiani, Manivannan, Pozzolini	Initial release

Table 1.1: Changelog of this document

1.2 Assigned class: TechDataServices

The TechDataServices.java is the class given for us for code inspection. Before executing our code inspection, a basic understanding of the code was required. From our study of the code, the TechDataServices.java class performs the following operations in the below mentioned sequence:

- Retrieves some RoutingTasks selected by Name or MachineGroup ordered by Name.
- Checks if there are no two routing tasks with the same SeqId are valid at the same period.
- Gets the techDataCalendar for a routingTask, if there is a routingTask associated with no MachineGroup the DEFAULT TechDataCalendar is returned.
- Finds the fisrt day in the TechDataCalendarWeek where capacity != 0, beginning at dayStart (dayStart included).

- Requests the remain capacity available for dateFrom in a TechDataCalendar. If the dateFrom is not in an available TechDataCalendar period,it returns zero value.
- Moves a day in TechDataCalendar, by producing the Timestamp for the beginning of the next day available and its associated capacity. If the dateFrom is not in an available TechDataCalendar period,it returns the next day available.
- Moves forward in a TechDataCalendar, starting from the dateFrom and moves forward only on available period. If the dateFrom is not an available TechDataCalendar period, the startDate is the begining of the next available day.
- Finds the last day in the TechDataCalendarWeek where capacity != 0, ending at dayEnd (dayEnd included).
- Requests the remain capacity available for dateFrom in a TechDataCalendar. If the dateFrom is not in an available TechDataCalendar period, it returns zero value.
- Moves a day in TechDataCalendar, by producing the Timestamp for the end of previous day available and its associated capacity. If the dateFrom is not in an available TechDataCalendar period,it returns the previous day available.
- Moves backward in a TechDataCalendar, starting from the dateFrom and moves backward only on available period. If the dateFrom is not an available TechDataCalendar period, the startDate is the end of previous available day.

These are just some routing checks that are performed and the methods of this class is implicitly used or called by other classes (from other packages).

Chapter 2

Results of inspection

This chapter contains the results of the code inspection that we did on the assigned class and methods. All the points of the checklist reported in the assignment were checked, and we also found other bad practices not listed in the checklist.

2.1 Notation

- The items of the checklist reported in the assignment will be referred as **C1**, **C2**, ..., while the general errors will be indicated with **Gen1**, **Gen2**, ...
- A specific line of code will be referred as follows: L.1234.
- An interval of lines of code will be referred as follows: L.1234∼1289.

2.2 Report

2.2.1 Checklist

1. **C11** L.50 L.51 L.53 L.55 L.58 L.60 L.144 L.146 L.147 L.285 L.289 L.291 L.329 L.350 L.360 L.439 L.443 L.445 L.485 L.506 L.516
   These lines contain *if* statements containing only one statement, not surrounded by curly braces.
2. **C12** L.264 L.294 L.336 L.418 L.448 L.492
   There are not blank lines between the end of a method and the beginning of the documentation of the following one.

3. **C13**
   For at least half of the code the line length exceed 80 characters.

4. **C14**
   L.62 L.75 L.90 L.111 L.134 L.144 L.160 L.204 L.212 L.216 L.273 L.283 L.288 L.294 L.305 L.315 L.319 L.330 L.337 L.370 L.427 L.442 L.448 L.449 L.459 L.469 L.474 L.486 L.493
   These lines exceed the length of 120 characters. Some of them are comments or part of the javadoc.

5. **C17** L.161
   The *else if* statement is not aligned with the beginning of the expression at the same level as the previous line.

6. **C17** L.132 L.88 L.179 L.186 L.196 L.205 L.276 L.308 L.430 L.462
   The *catch* statement is not aligned at the same level. This could be avoided either during the code development by proper checking.

7. **C17** L.189 L.350 L.360 L.506 L.516
   The *else* statement is not aligned at the same level. This could be avoided either during the code development by proper checking.

8. **C18**
   While methods are well explained ("field knowledge" is still needed to understand), the class entirely lacks the description.

9. **C19** L.96 L.97 L.273∼280
   Four TODO comments are present in the code. The former are poorly explained and lack the removal time. The latter are better explained but still no date.

10. **C24**
    The package and import statements were found in the correct order as per the code inspection document.

11. **C25**
    The class or interface declarations order is followed correctly.

12. **C27** L.224 L.378
    Both the *switch* blocks are a copy paste in our service which paves way for duplicates. But this is quite preferable here rather than using different piece of code for the same logical implementation (i.e. for a better understanding).

13. **C27** L.218 L.271 L.301 L.344 L.372 L.425 L.455 L.500
    All the methods are long more than 10 lines and some methods are even more than 20 lines. This should be optimized by using fewer conditional checks covering all the requirements.

14. **C26**
    All the methods are grouped by functionality rather than by scope or accessibility.

15. **C33** L.313∼319 L.352 L.435∼438 L.467∼474 L.508
    These lines contain declarations within the blocks the belong to, instead of being at the beginning.

16. **C40** L.92 L.124 L.137 L.144∼147 L.182 L.183 L.192 L.201 L.223 L.337
    We can find the inappropriate use of '==' and '!=' for object comparison many times in the code. The list of the lines:
    - *L.92*: LinkedList
    - *L.124*: String
    - *L.137*: List
    - *L.144∼157*: Timestamp
    - *L.182*: GenericValue
    - *L.183*: String
    - *L.192*: List
    - *L.201*: TechDataCalendar
    - *L.223*: Double
    - *L.377*: Double

17. **C42**
    The error messages are comprehensive and are in-line as per the code inspection document.

18. **C44** L.224 L.378
    Two switch loops that could be considered brutish programming. Anyway since the switch loop cycles constants, I think that adopting a String vector to replace it, would damage the readability.

19. **C54**
    All switch statements are addressed by a break.

20. **C55** L.224 L.378
    There is no default branch for any of the switch statements. The solutions we suggest for this problem is to leave the code undisturbed, because all the days are present in the switch cases such that it executes any one of the switch case and so no default is needed. Without knowing the dayStart or dayEnd it would be wrong to define a default branch here.

2.2.2 General

Other inaccuracies found during the code inspection:

21. **Gen1** L.261 L.284 L.317 L.328 L.415 L.438 L.471 L.483
    No import statement used for Integer. Import that has to be used here was: java.lang.Integer or import java.lang.* can be used.

22. **Gen2** L.221 L.287 L.356 L.375 L.441 L.472 L.512
    No import for Double was done: java.lang.Double or import java.lang.* could have solved the issue.

23. **Gen3** L.418 L.448 L.264 L.394 L.336
    TechDataCalendar was spelled wrong here within the comments. The final letter is missing.