

Code Review

PR Software Engineering

Gruppe 3

259035 (2018S)

*Anwesende: ?*

*Programm: AppTimeTracker*

*Datum: ?*

*Typ: Kompletter Review vom Code*

Checkliste

Symbols:

🗸 – ok (i.e., code doesn’t need to be corrected)

x – not ok (i.e, code needs to be corrected)

? – not existent (i.e., adressed behaviour doesn’t apply to the current state of the code)

- General

1. [ ] The code works
2. [ ] The code is easy to understand
3. [ ] Follows coding conventions
4. [ ] Names are simple and if possible short
5. [ ] Names are spelt correctly
6. [ ] Names contain units where applicable
7. [ ] Enums are used instead of int constants where applicable
8. [ ] There are no usages of 'magic numbers'
9. [ ] All variables are in the smallest scope possible
10. [ ] All class, variable, and method modifiers are correct.
11. [ ] There is no commented out code
12. [ ] There is no dead code (inaccessible at Runtime)
13. [ ] No code can be replaced with library functions
14. [ ] Required logs are present
15. [ ] Frivolous logs are absent
16. [ ] Debugging code is absent
17. [ ] No System.out.println or similar calls exist
18. [ ] No stack traces are printed
19. [ ] Variables are not accidentally used with null values
20. [ ] Variables are immutable where possible
21. [ ] Code is not repeated or duplicated
22. [ ] There is an else block for every if clause even if it is empty
23. [ ] No complex/long boolean expressions
24. [ ] No negatively named boolean variables
25. [ ] No empty blocks of code
26. [ ] Ideal data structures are used
27. [ ] Constructors do not accept null/none values
28. [ ] Collections are initialised with a specific estimated capacity
29. [ ] Arrays are checked for out of bound conditions
30. [ ] Catch clauses are fine grained and catch specific exceptions
31. [ ] Exceptions are not eaten if caught, unless explicitly documented otherwise
32. [ ] APIs and other public contracts check input values and fail fast
33. [ ] Files/Sockets/Cursors and other resources are properly closed even when an exception occurs in using them
34. [ ] StringBuilder is used to concatenate strings
35. [ ] Null/None are not returned from any method
36. [ ] Floating point numbers are not compared for equality
37. [ ] Loops have a set length and correct termination conditions
38. [ ] Blocks of code inside loops are as small as possible
39. [ ] Order/index of a collection is not modified when it is being looped over
40. [ ] No methods with boolean parameters
41. [ ] No object exists longer than necessary
42. [ ] Design patterns if used are correctly applied
43. [ ] No memory leaks
44. [ ] Law of Demeter is not violated
45. [ ] Methods return early without compromising code readabilit
    * + Java only
46. [ ] Appropriate JCIP annotations are used
47. [ ] No use of Object class, use generics instead
48. [ ] Uses final modifier to prevent mistaken assignments
    * + Documentation
49. [ ] All methods are commented in clear language.
50. [ ] Comments exist and describe rationale or reasons for decisions in code
51. [ ] All public methods/interfaces/contracts are commented describing usage
52. [ ] All edge cases are described in comments
53. [ ] All unusual behaviour or edge case handling is commented
54. [ ] Data structures and units of measurement are explained
    * + Threading
55. [ ] Objects accessed by multiple threads are accessed only through a lock, or synchronized methods.
56. [ ] Race conditions have been handled
57. [ ] Locks are acquired and released in the right order to prevent deadlocks, even in error handling code.
58. [ ] StringBuffer is used to concatenate strings in multi-threaded code
    * + Security
59. [ ] All data inputs are checked (for the correct type, length/size, format, and range)
60. [ ] Invalid parameter values handled such that exceptions are not thrown
61. [ ] No sensitive information is logged or visible in a stacktrace

Bei folgenden Punkten wurden Fehler entdeckt: