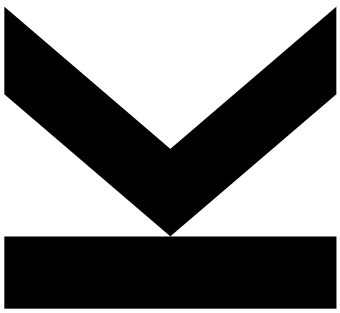
JOHANNES KEPLER
UNIVERSITÄT LINZ

PR SOFTWARE ENGINEERING

Gruppe 3

259035 (2018S)

Code Review



JOHANNES KEPLER
UNIVERSITÄT LINZ
Altenberger Straße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Anwesende: Milos Tomic

Programm: AppTimeTracker

Datum: 23.05.2018

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 [ ] Frivolous logs are absent
  15. [ ] Debugging code is absent
  16. [ ] No System.out.println or similar calls exist
  17. [ ] No stack traces are printed
  18. [ ] Variables are not accidentally used with null values
  19. [ ] Variables are immutable where possible
  20. [ ] Code is not repeated or duplicated
  21. [ ] There is an else block for every if clause even if it is empty
  22. [ ] No complex/long boolean expressions
  23. [ ] No negatively named boolean variables
  24. [ ] No empty blocks of code [ ] Ideal data structures are used
  25. [ ] Constructors do not accept null/none values
  26. [ ] Collections are initialised with a specific estimated capacity

27. [ ] Arrays are checked for out of bound conditions
28. [ ] Catch clauses are fine grained and catch specific exceptions
29. [ ] Exceptions are not eaten if caught, unless explicitly documented otherwise

30. [ ] APIs and other public contracts check input values and fail fast
31. [ ] Files/Sockets/Cursors and other resources are properly closed even when an exception occurs in using them
32. [ ] StringBuilder is used to concatenate strings
33. [ ] Null/None are not returned from any method
34. [ ] Floating point numbers are not compared for equality
35. [ ] Loops have a set length and correct termination conditions
36. [ ] Blocks of code inside loops are as small as possible
37. [ ] Order/index of a collection is not modified when it is being looped over
38. [ ] No methods with boolean parameters
39. [ ] No object exists longer than necessary
40. [ ] Design patterns if used are correctly applied
41. [ ] No memory leaks
42. [ ] Law of Demeter is not violated
43. [ ] Methods return early without compromising code readabilit

- Java only

44. [ ] Appropriate JCIP annotations are used
45. [ ] No use of Object class, use generics instead
46. [ ] Uses final modifier to prevent mistaken assignments

- Documentation

47. [ ] All methods are commented in clear language.
48. [ ] Comments exist and describe rationale or reasons for decisions in code
49. [ ] All public methods/interfaces/contracts are commented describing usage
50. [ ] All edge cases are described in comments
51. [ ] All unusual behaviour or edge case handling is commented
52. [ ] Data structures and units of measurement are explained

- Threading

53. [ ] Objects accessed by multiple threads are accessed only through a lock, or synchronized methods.
54. [ ] Race conditions have been handled

55. [ ] Locks are acquired and released in the right order to prevent deadlocks, even in error handling code.
56. [ ] StringBuffer is used to concatenate strings in multi-threaded code

- Security

57. [ ] All data inputs are checked (for the correct type, length/size, format, and range)
58. [ ] Invalid parameter values handled such that exceptions are not thrown
59. [ ] No sensitive information is logged or visible in a stacktrace