Quality Inspection Checklist for Golden Dragon Software Solutions

This checklist is intended to be used with a Findings Report for inspecting and reporting on Java class, SQL, and JavaFX class items. It is important that our reporting is meant to be suggestive, not prescriptive.

Inspection Instructions

1. Familiarize ourselves with the group member's code and understand how it works, such as: how it is used, what it does, and what integrated functionality they used. If the code is unfamiliar , consult our class diagrams, external documentation, or just Google it.
2. Inspect the code using the checklist noted below.
3. For any code that does not meet any of the checklist criteria, make detailed note of it where it occurs in the code (Class, line number, function, etc.) and what it contains.
4. For any code elements that appear to be absent in the project, based on our class diagrams, make note of the absence.

Quality Criteria and Inspection items

1. Completeness
    a. Functions are declared as private or public based on necessity.
    b. All parameters of a method are used.
    c. All functions should be used at least once, including functions for testing purposes.
2. Consistency
    a. Naming conventions are consistent throughout the project.
    b. Coding style/grammar is consistent throughout the project.
    c. Documentation within the project is logical throughout.
    d. If there are more than three imports from a package use '.*' (e.x. import Controller.*)
3. Efficient Processing

    a. Project uses ArrayLists for passing information for ease of use and parsing.
    b. Loops denote simple, yet repetitive tasks to simplify the code base.
    c. Frequently used code should be in its own function/class to eliminate repetition of the code.
4. Efficient Storage
    a. All required information is stored in the database, even if not immediately used.
    b. Data structures are only as big as they need to be
    c. All local and class variables are used somewhere.
    d. All data is accessed with designated getters and setter functions.
5. Simplicity
    a. Standard coding idioms are used where appropriate (e.x. for (int i=0; i<arraySize; i++)).
    b. Code is not longer than a page (60 lines of code) per function.
    c. Use no more then 120 characters per line, break over an addition line if necessary
6. Documentation
    a. Any obscure or non-standard code is commented to clarify functionality.
    b. Functions are described to clarify functionality, restrictions, and parameters
    c. Names of variables, functions, and classes are meaningful, pronounceable, and not obscure.
    d. Booleans are not compare to constants: do this (!found); NOT this (found == false)

Findings Report

Inspector's Name: Kyle Klenk inspecting Erling Lefsrud

Date Started: March 12, 2020

Date Completed: March 13, 2020

Project Name: Day Planner

Time Report: 1.25 hours

Time spent on Familiarization with the Code: 30 mins

Time spent on Filling in Findings Report: 15 mins

Time spent on Actual Inspection: 30 mins

Findings:

#	Description	Class Name	Method Name	Line #	Associated Checklist Item #
1	Unnecessary Comments, code itself is clear enough	CalendarItem	Calendar Item Instance Variables	7, 9	6. a)
2	Commented out methods, should be implemented or deleted	CalendarItem	getMark(), setMark()	58, 62	1)
3	Unused Variables, do not need to exist if not being used.	Calendar	Calendar Instance Variables	21, 23	4

4	Unused Methods, Should be used somewhere or deleted	Calendar	newEvent	141	1
5	Commented out code in a method, adds confusion to function. Should implement the code or delete it if not using it.	Calendar	insertEvent	131	1/6

Further Comments from the Inspection of this Class:

Code looks good throughout and is easy to follow. It follows the guidelines of simplicity and consistency well. The main things found were unnecessary comments and what looked to be either leftover code and or non working code that was commented out.