Quality Assurance Checklist For Javadoc

CLASS DEFINITIONS (a.k.a. Class Skeletons, Javadoc)

1. [x] Does each class in the class diagram have an associated definition file that compiles? (Not needed for 3rd party classes, obviously.)
2. [x] Does each class header include the documentation required in the document format guidelines (especially an @author tag and a well written descriptive header)?
3. [x] Does each method header include the documentation required in the document format guidelines?
4. [x] Is each method name a verb phrase in active voice?
5. [x] (DBC teams) Does each method have documented pre/post conditions that are clear, complete, and correct? (Exception: accessor methods normally don't require pre/post conditions).
6. [x] (non-DBC teams) Does each method document how all possible error conditions will be handled?
7. [x] Does the name for each abstraction and its description use terms from the problem domain and not the implementation? E.g., GameBoard, not Grid. CustomerList, not CustomerArray.
9. [x] Are the data attributes consistent with the Data Dictionary?
10. [x] Has the range of data values been constrained so as to not allow values which are meaningless in the problem domain? E.g., don't use int for data that can never be negative. Instead, use something like this Natural class.
11. [x] Do collection classes handle their own persistent data?
12. [x] Are all instance variables private?
13. [x] Is there a comment included for each design element that traces it back to the SRS?

Comments:

4. We did not make the is_____ method names into a verb phrase in active voice because the name conformed with a reasonable rationale. Referenced here:

   http://www.cwu.edu/~gellenbe/javastyle/method.html

6. We decided to use DBC for majority of the Javadoc. However, we needed to perform user input checking, so we decided to document the methods, getBet(), validateInput(), getBlinds(), from HumanPanel class and NewGameDialog class.

10. We will justify our decision to use int instead of Natural in our Design Issue section that will be turned in with the rest of High Level Design deliverable.
13. We decided to create a Traceability List rather than comment on our actual source code. Referenced here:

https://wiki.csc.calpoly.edu/RRAMM/wiki/Traceability

QA Manager: Ryuho Kudo