**Tables 1 and 2 are both from CH18 of Y&P page 346 and 347**

Table 1: Java Checklist: Level 1 Inspection (single-pass read-through, context independent)

| **FEATURES (where to look and how to check):** | **yes** | **no** | **comments** |
|---|---|---|---|
| **IMPORT SECTION: Are the following requirements satisfied?** | | | |
| Brief comment on each import with the exception of standard set: java.io., java.util. | | X | Not required as this is a solo project, and no one else will interact with the code. |
| Each imported package corresponds to a dependence in the design documentation | X | | |
| **CLASS DECLARATION: Are the following requirements satisfied?** | | | |
| The visibility marker matches the design document | X | | |
| The constructor is explicit (if the class is not static) | | X | Not required to be explicit for this class |
| The visibility of the class is consistent with the design document | X | | |
| **CLASS DECLARATION JAVADOC: Does the Javadoc header include:** | | | |
| One sentence summary of class functionality | X | | |
| Usage instructions | | X | Not required as I am the sole developer of this project, and no one else will be interacting with this software. |
| **IDIOMATIC METHODS: Are names compliant with the following rules?** | | | |
| Method name: capsAfterFirstWord | X | | |
| Local variables: capsAfterFirstWord. Name may be short (e.g., i for an integer) if scope of declaration and use is less than 30 lines. | X | | |

Table 2: Java Checklist: Level 2 Inspection (comprehensive review in context)

| **FEATURES (where to look and how to check):** | **yes** | **no** | **comments** |
|---|---|---|---|
| **METHODS: Are the following requirements satisfied?** | | | |
| The method semantics are consistent | X | | |
| Usage examples are provided for nontrivial methods | | X | No need for examples as I am the sole developer of the project. |
| **FIELDS: Are the following requirements satisfied?** | | | |
| The field is necessary | | X | |
| **DESIGN DECISIONS: Are the following requirements satisfied?** | | | |
| Each design decision is hidden in one class | X | | |
| Classes encapsulating a design decision do not unnecessarily depend on other design decisions | X | | |
| Adequate usage examples are provided | | X | Examples are not required, as I am the sole developer of this project. |
| Design patterns are used and referenced where appropriate | | X | |
| If a pattern is referenced: The code corresponds to the documented pattern | | X | |

| **Use-Case** | **Missing Functionalities?** | **Comments** |
|---|---|---|
| Submitting an order with an invalid card number | No | N/A |
| Submitting an order with an invalid expiry date | No | N/A |
| Submitting an order with an invalid CVV | No | N/A |
| Submitting an order with an incorrect total | No | N/A |
| Submitting an order with pizzas that don't exist | No | N/A |
| Submitting an order with more than 4 pizzas | No | N/A |
| Submitting an order with pizzas from differing restaurants | No | N/A |
| Submitting an order for pizzas from a close restaurant | No | N/A |

Table 3: Use Cases Evaluation For Order Validation