Verify Checklist

	Type	Description / Pointers
☐	Comments	Comments in code may obscure bad naming, long methods, etc. Therefore comments in code are by default a code smell. Refactoring: Extract method Rename Method / Field Introduce Assertion
☐	Duplicate Code	Duplicated lines in different methods or duplicated methods in different classes, is what people call duplication, but duplicated object usages (see Feature Envy) is a form of duplication as well and may be less apparent. Refactoring: Extract Method / Class, Pull Up Method, Form template
☐	Conditional Complexity	Complex conditional statements often grow from a simple if. We should be critical about conditionals and boolean expressions, as they appear more difficult to read and maintain than you might expect. Refactoring: Replace conditional with Strategy Move Embellishment to Decorator Decompose Conditional Replace State-Altering Conditionals with State Introduce Null Object
☐	Long Method	A long method is hard to understand because of too much detail. They tend to become linger, introduce hard-to-detect duplication and often suffer from Primitive Obsession. Refactoring: Extract Method, Decompose conditional, etc.
☐	Magic Number	Magic numbers are literal values that appear in the code. The meaning of the values is unclear from the code. Refactoring: Replace with Symbolic constant
☐	Uncommunicative name	A name that doesn't communicate its intent well enough. Refactoring: Rename method / class / field / variable
☐	Divergent change	If a certain class is often changed for different reasons, the class may have to many responsibilities Refactoring: Extract class
☐	Shotgun Surgery	Your code smells when changes affect many classes. Refactoring: Move Method / Field. Inline class
☐	Feature Envy	A piece of behavior seems more interested in data from another class than its own. Refactoring: Move method / Extract method
☐	Inappropriate Intimacy	Two of more classes that spend a lot of time together. Bidirectional dependencies and touch their private parts (data). Refactoring: Move method / Field, Change bidirectional behavior to Unidirectional
☐	Train Wrecks	object.getThis().getThat().getSuch().doThis() Refactoring: Extract Method / Move Method / Hide Delegation
☐	Switch statements	Switch statements are almost never a good object oriented construct. (procedural) Refactoring: Replace Conditional with Polymorphism / Strategy / State / Explicit methods
☐	Indecent Exposure	Code that is not important to clients should not be exposed. Refactoring: Encapsulate Classes with Factory
☐	Primitive Obsession	Using many primitive types in implementation obfuscates the intent of the implementation. Refactoring: Replace data value with Object Replace Type code with Class / Subclasses / State / Strategy Extract Class