**SOLID CHECKLIST**

**This is by no means an exhaustive checklist. The possibilities for violation are as diverse as the problems an application can be developed to solve.**

**But these are a good point of reference to get you into good habits and heighten your awareness of SOLID principles.**

**TIPS:**

* **Make sure you fill in the solution you chose and why for your own reference. You’ll find that many violations have the same optimal solution, conceptually speaking. So, if you can look back at previous checklists, you can see where you’re tripping up and where you need to focus more of your attention to reduce future violations.**
* **Don’t overwrite the “solid-checklist-myclass.docx” every time you refactor the class. Save each one: e.g. “solid-checklist-myclass-v0.1-220109.docx”. Or, better still, put them in git! This way you can track errors in how you approach resolving violations.**
* **Don’t feel bad. The human brain isn’t naturally predisposed to think this way. Nobody gets it right first time. Why else d’you think I came up with the idea of a checklist? 🤣**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| | **PROJECT:** | **CLASS:** | **DATE:** | | --- | --- | --- | | | | |
| **Vio.** | **Violation indicators** | **Warning with solid fill** | **Solution chosen and why.** |
| **SRP** | **Does the class have a lot of public methods?**  **(Having a lot of any type of method could also be an indicator but if they’re public, it’s a stronger code smell)**  **Do those methods deal with things that aren’t that closely related?**  **Are any of those methods really long (40+ lines) and involved? Is there a lot of indentation?** |  |  |
| **SRP** | **Does the class implement a lot of interfaces?**  **Are the interfaces closely related?**  **Do the interfaces have a lot of methods (possible ISP violation)?** |  |  |
| **SRP** | **Identify the class’s responsibility.**  **Do any of the methods perform tasks that do not directly fulfil that responsibility?** |  |  |
| **SRP** | **Identify the class’s dependencies.**  **Do any of the methods appear to be resolving any of those dependencies directly?** |  |  |
| **SRP** | **Does the class have a lot of fields?  Do those fields provide flexibility across too many variations in instances of the class?** |  |  |
| **ISP** | **Does the interface have a lot of methods?**  **Do those methods deal with things that aren’t that closely related?**  **Could those methods be divided into groups?** |  |  |
| **ISP** | **Do the method parameters indicate a gap in the “processing story”? Are the types not that closely related?**  **Do any of the methods have too many parameters and/or overloads?** |  |  |
| **ISP** | **Identify the interface’s single responsibility as if it were a class.**  **Do any of the methods perform tasks that do not directly fulfil that responsibility?** |  |  |
| **ISP** | **Identify the dependencies of the interface’s implementation.**  **Do any of the method parameters indicate the implementation would use those values to resolve a dependency directly? (E.g. a connection string or a file path)** |  |  |
| **DIP** | **Does the class directly**  **instantiate any**  **external/service objects?** |  |  |
| **DIP** | **Does the class receive any concrete types thru the constructor?  Do any of the public methods receive concrete types to resolve class level dependencies?**  **I.e. if the instance is only used by the method in question, it is not a class level dependency.** |  |  |
| **DIP** | **Does the class directly access a data source to load values into its data structures?   Unless the class is a data implementation level object, this is a dependency that needs inverting but this violation should have been caught by the SRP checklist.** |  |  |
| **OCP** | **Before beginning the OCP checklist, ensure all SRP, ISP & DIP violations are resolved.** | **N/A** | **N/A** |
| **OCP** | **Was the new requirement implemented by adding a parameter to a method? Does the parameter require a value be provided? I.e. it’s not practical to give it a default value?** |  |  |
| **OCP** | **Does the change necessitate the modification of an interface?** |  |  |
| **OCP** | **Does the change accommodate a variation in the class’s functionality that introduces too much complexity or a comparable variation is already provided in a child class?** |  |  |
| **OCP** | **Does the change necessitate any “dead code”? Are there circumstances where the new logic would need to be skipped or dummy values used to ensure the existing functionality still worked?  I.e. is it an “edge case”?** |  |  |
| **LSP** | **LSP violations are actually a \*design\* smell, but that makes the resultant code smells quite…pungent 😊**  **Again, make sure you’ve completed the whole checklist before starting this section.** | **N/A** | **N/A** |
| **LSP** | **Find the uppermost parent of the class that is the subject of this checklist.  Does this child throw an exception that the parent (and hence clients of the parent) wouldn’t expect?** |  |  |
| **LSP** | **Does this child override methods (including properties, anonymous methods etc) that do nothing?** |  |  |
| **LSP** | **Do/would clients of the parent have/need type-checking code in order to sidestep issues raised by this child?  I.e. can a client unconditionally call the same sequence of methods on any class in the inheritance chain, safely? If not, then LSP has been violated.** |  |  |