What is high quality code

1. Easy to change. Robust code is easier to change because external factors don’t affect its behaviour .
2. Less bugs. The more robust the code is, the less combinations that cause a bug in the code.
3. Better integration. Robust code is predictable, so from an integrator point view, is always going to have the same behavior.
4. Consistency. Repeat this with me “Robust code, works fine no matter who, when or how it gets called”.

To help spot fragile code here there is a list of common fragile code patterns.

1. Public getters / setters / fields.
2. Bad equals / hashCode / compareTo methods.
3. Code duplication.
4. Use of plain constants when an enum should be used.
5. Not providing with any factory or build pattern for complex objects.
6. Dependency on global variables.
7. Coding against implementations and not interfaces.

One of the things that upsets me from modern frameworks is that they push you to create non robust code, they sometimes force you to have default constructors, getters and setters for all instance variables… but there’s no much we can do about it, and sometimes the advantage of using them is worth the hacking, but, from my point of view, fragile code shouldn’t be accepted in any other scenario.

Test code deserves a special mention. Test code is usually seen as not as important as production code, and this causes lots of developers to write very fragile tests. I am sure that anyone who has work in a large project using automated tests will absolutely agree with me that there is nothing more tedious than changing 10 lines of test code for each line of production code that changes, and that can be prevented with robust test code. So when writing test codes, please keep in mind that you have to care for it just as much as if was production code!