**What is High-Quality Code?**

*"To err is human, but to really foul things up you need a computer."  
Paul Ehrlich*

Computer programmers are like artists. What is so similar between those two seemingly unrelated worlds? Like an artist can pick colours, shades and hues, apply different brushes, and use various techniques to convey a certain mood and make a masterpiece, **a good computer programmer can write code which not only machines, but people are able to understand too**. Speaking about code is similar to discussing a design - it can be functional but really ugly to look at, or it can be really beautiful with not much good use, or better, something in between.

So, how do we understand what high-quality code is? It **is as simple to see as the good design**. It serves its purpose well, and it is easy to read and maintain. You can understand what it does by simply scanning through it, and you do not need to ask any questions about what a certain thing is or what it does. In fact, everyone can write code for the compiler, but not everyone can write code for the people. This is what distinguishes a good and a bad software developer.

High-quality code follows the basic principles of good design. It is **self-documenting** - ideally, one does not ever need to read anything besides the code itself to understand what everything is. Good code is not only easy to read, it is also **simple enough to maintain**. This means extending functionality, troubleshooting, and making changes takes as little time as possible. Reading high-quality code is not much different than reading a book, and making changes, however major they can be, is easy. The code is self-contained and pluggable. It is flexible and easy to change.

As years have passed, there have been some general principles set for high-quality code. Apart from their fancy names (like **DRY**, **SOLID**, **design patterns**, or **OO design**), they have one thing in common - they address a problem many people have encountered and attempt to solve it in the most elegant and suitable way. In fact, there are so many techniques, or principles, or pieces of advice on writing high-quality code, that some of them confront each other. But they are all parts of the solution to the given problem - how to write good code. Everyone may choose their own set of principles to follow, or things to do. The important thing is this leads to a better software design, and, ultimately, more happy customers.

Technically speaking, high-quality code can be defined in terms of the text. It has **comments** which explain tricky parts of it and document what everything does. The **variable names** are chosen well and are self-explanatory, the main parts (like classes, methods, or control statements) are **well-structured**, and one of the most important things - the resulting software **behaves correctly** at all times, in all conditions. High-quality code utilizes **object-oriented design** and **object-oriented programming** principles to make reading and extending much easier. It also incorporates **design patterns** (which are good solutions to certain problems which many programmers have), it applies the principles of **strong cohesion** and **loose coupling**, and most importantly, not only does the code work correctly, **it also proves that** by using various tests.

This is basically an outline of what high-quality code is and should be. But **why do we need high-quality code** after all? A piece of software is not written once and for all. As in nature, the software market principle is 'survival of the fittest, or the most adaptable'. Following the principles of high-quality code, programmers can fairly easily adapt their software, and even make great changes in it, to satisfy a customer's needs.

A common misconception is that software is ready once it has been written. Writing software is only a part of the picture, after that comes support and this is where a good design can help a lot. You had better write good code and take following support as a blessing, not as a curse.