|  |  |
| --- | --- |
| **General** | |
| Avoid code duplication | Code duplication is always source of maintenance problem and bug. |
| Avoid hard-coded values | Hard-coded values cannot speak for themselves. Use well-named constants/variables instead. |
| Avoid Large Class | A class that is trying to do too much can usually be identified by looking at how many instance variables it has. When a class has too many instance variables, duplicated code cannot be far behind. |
| Avoid Long Method | The longer the method the harder it is to see what it's doing. |
| Long Parameter List | Don't pass in everything the method needs; pass in enough so that the method can get to everything it needs. |
| Avoid Uncommunicative Name | Choose names that communicate intent (pick the best name for the time, change it later if necessary). |
| Minimize number of “switch” statements | Can very quickly become the source of duplication. Should be refactored with “strategy” and “state” patterns. |
| Avoid control flags | Code often becomes unreadable with control flags |
| Always follow project’s coding convention | This is a must to make code more readable and maintainable by the team |
| Master not only your programming language but also its important APIs/libraries/utilities | For Java: JDK collection framework, Hibernate API, Apache Commons, Guava …  For C#: LINQ, NHibernate, Entity Framework … |
| If you don’t know how your program works, don’t hesitate to debug it | Debugging is an extremely useful way to understand how a program works |
| **Exception handling** | |
| Never leave empty catch block | Exception must always be handled (being traced at least). Otherwise, application error (once occurred) can never be traced/analyzed and resolved. |
| Be specific and contextual when throwing exception | Contextual information is extremely helpful for analyzing application error |
| Always put try…catch in finally | To avoid exception loss if error happens in finally clause |
| All resources must be closed after used | To avoid running out of resources |

**IMPORTANT PROGRAMMING PRACTICES**