# Anti-pattern definitions[4]

| Name | Description | Refactoring solution |
|---|---|---|
| Blob (BL) [1] | A large class that absorbs most of the functionality of the system with very low cohesion between its constituents.. | Move method (MM). Move the methods that does not seem to fit in the Blob class abstraction to more appropriate classes[3]. |
| Lazy Class (LC)[1] | Small classes with low complexity that do not justify their existence in the system. | Inline class (IC). Move the attributes and methods of the LC to another class in the system. |
| Long Parameter List (LP) [1] | A class with one or more methods having a long list of parameters, specially when two or more methods are sharing a long list of parameters that are semantically connected. | Introduce parameter object (IP). Extract a new class with the long list of parameters and replace the method signature by a reference to the new object created. Then access to this parameters through the parameter object. |
| Spaghetti Code (SC) [2] | A class without structure that declares long methods without parameters. | Replace method with method object (EC). Extract long methods into new classes so all local variables become fields on that object. |
| Speculative Generality (SG)[1] | There is an abstract class created to anticipate further features, but it is only extended by one class adding extra complexity to the design. | Collapse hierarchy (CH). Move the attributes and methods of the child class to the parent and remove the \textit{abstract} modifier. |

# On-line resources

| |
|---|
| https://sourcemaking.com/refactoring |
| https://refactoring.guru/ |
| https://martinfowler.com/books/refactoring.html |
| http://antipatterns.com/ |

# References


| | |
|---|---|
| [1] | Fowler, M. (1999). Refactoring: Improving the Design of Existing Code. Boston, MA, USA: Addison-Wesley. |
| [2] | Brown, W. H., Malveau, R. C., McCormick, H. W., & Mowbray, T. J. (1998).AntiPatterns: refactoring software, architectures, and projects in crisis. John Wiley & Sons, Inc.. |
| [3] | Seng, O., Stammel, J., & Burkhart, D. (2006). Search-Based Determination of Refactorings for Improving the Class Structure of Object-Oriented Systems. GECCO 2006: Genetic and Evolutionary Computation Conference, Vol 1 and 2, 1909-1916. doi:10.1145/1143997.1144315 |

| | |
|---|---|
| [4] | Morales, R., Sabane, A., Musavi, P., Khomh, F., Chicano, F., & Antoniol, G. (2016, 14-18 March 2016). Finding the Best Compromise Between Design Quality and Testing Effort During Refactoring. Paper presented at the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER). |