Принципы разработки v1.1

Руководство. ООО “Пикабу”

1. Дробите логику класса на небольшие методы, не превышающие высоту экрана в 80 строк [1];
2. Дробите длинные выражения и литералы на несколько строк, не превышающих ширину экрана в 80-120 символов [4] [5] [6];
3. Стремитесь использовать как можно меньше вложенных друг в друга конструкций (for, if, и тд) [2];
4. Избегайте сложные конструкции управления потоком, такие как goto и рекурсии [3];
5. Добавляйте в циклы защитный ограничитель на максимальное количество итераций [3];
6. Избегайте лишний else блок, который можно опустить [2];
7. Стремитесь использовать самодостаточные/самоговорящие имена классов, функций, переменных и тд [1] [13];
8. Комментируйте код там, где это имеет смысл; пишите код так, чтобы он не нуждался в комментировании [1] ;
9. Следуйте принципам Абстракции и DRY (Don’t Repeat Yourself), избегайте дублирования кода [10] [12];
10. Стремитесь к слабому сцеплению (low coupling) между зависимыми классами разных модулей системы, применяйте IoC, DI, Service Locator и тд [9] ;
11. Пишите классы и модули с сильной связностью (high cohesion) [9];
12. Храните магические числа в enum, константах, конфиге или словаре, избегайте хардкодов [1];
13. Всегда используйте фигурные скобки для структур с одним выражением [1] [5] ;
14. Избегайте создания Божественного объекта (God object), следуйте SRP (Single Responsibility Principle) [8];
15. Удаляйте код, который больше работать не будет, согласно YDNIA (You Don't Need It Anymore) [8] [13];
16. Разделяйте методы для получения информации и выполнения действий, согласно CQS (Command-Query Separation) [11];
17. Следуйте методикам Безопасного программирования (Secure coding) [7];
18. Не заставляйте классы реализовывать то, что им не надо, согласно ISP (Interface Segregation Principle) [12];
19. По возможности всегда выбирайте простые решения (KISS) [14];
20. Инкапсулируйте то, что изменяется [12];
21. Код должен зависеть от абстракций, а не от конкретных реализаций [12];
22. Держите баланс между эффективностью и ясностью кода, не допускайте погоню за ложной эффективностью [13] [14];
23. Не добавляйте функциональности, пока вы точно не уверены в её надобности, согласно YAGNI (You aren't gonna need it) [8] [13];

24. Если есть возможность явно указать что-то - указывайте явно [15];
25. Делегируйте, не делайте всю работу сами, поручите её соответствующему классу [12];

Ссылки

1. Coding Guidelines: Clean Code From Day 1 https://medium.com/@luisacarrion/general-coding-guidelines-clean-code-from-day-1-9ab0804e5d91
2. Object Calisthenics https://williamdurand.fr/2013/06/03/object-calisthenics/
3. The Power of 10 http://web.eecs.umich.edu/~imarkov/10rules.pdf
4. Google JS Code Style https://google.github.io/styleguide/jsguide.html#formatting-column-limit
5. PHP Code Style https://www.php-fig.org/psr/psr-2/
6. Android Code Style https://source.android.com/setup/contribute/code-style#limit-line-length
7. OWASP Secure Coding Practices https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf
8. Code Smell https://wiki.c2.com/?CodeSmell
9. Cohesion and Coupling https://enterprisecraftsmanship.com/posts/cohesion-coupling-difference/
10. Abstraction Principle https://en.wikipedia.org/wiki/Abstraction_principle_(computer_programming)
11. CQS Principle https://martinfowler.com/bliki/CommandQuerySeparation.html
12. Принципы ООП https://habr.com/ru/company/skillbox/blog/454314/
13. Rules of Trumb https://medium.com/@vedantsopinions/software-engineering-rules-of-thumb-63060ca51b94
14. Rules of Trumb https://wou.edu/las/cs/csclasses/cs161/Lectures/rulesofthumb.html
15. Zen of Python https://www.python.org/dev/peps/pep-0020/