Wszystko co programista powinien wiedzieć!

1. Spłacaj dług techniczny jak szybko to możliwe
2. Zaadaptowanie zasad programowanie funkcyjnego pozwoli zwiększyć jakość kodu
3. Użytkownik który użyje twojej aplikacji nie myśli tak jak Ty!
4. Zadbaj aby zespół korzystał z tego samego "Code-Style"
5. Pięknie drzemie w prostocie
6. Dobrze się zastanów zanim zabierzesz się za Refactoryzację
7. Strzeż się współdzielenia kodu
8. Zawsze pozostaw moduł w lepszym stanie niż go zastałeś
9. Uwierz, że popełnisz błędy.
10. Dobierz narzędzie rozważnie.
11. Uczynnej szczegóły implementacyjne pod zrozumiałym kodem.
12. Kod to konstrukcje, które wymaga wysokiej jakości
13. Styl i wygląd kodu ma znaczenie!
14. Proces Code-Review pozwala utrzymać wysoką jakość kodu.
15. Kod powinien być samo dokumentujący się.
16. Jeśli dodajesz komentarze to mają być proste i klarowne.
17. Komenty tylko to czego kod nie może powiedzieć
18. Aby zapewnić dobry rozwój kariery musisz poświęcić odrobinę wolnego czasu.
19. Nazywaj funkcje tak aby wyjaśniały co będą robić.
20. Nie zostawiaj procesu instalacji i deployment na koniec.
21. Opuszczaj swoją strefę komfortu.
22. Nie można bać się wprowadzać zmian, inaczej gdzieś jest błąd.
23. Zastanów się, czy możesz pokazać zmiany bez obaw na krytykę.
24. Nie ignoruj uwag i ostrzeżeń.
25. Nie ucz się języka! Poznaj jego kulturę.

26. Nie pokazuj użytkownikowi raportu z wyjątku
27. Każdy proces manualny musi być zautomatyzowany
28 Developerzy nie mogą mieć dostępu do środowiska produkcyjnego.
29. Hermetyzuj zachowanie, nie tylko stan
30. Liczby zmiennoprzecinkowe nie są liczbami rzeczywistymi
31. Realizuj projekty open-source.
32. Unit testy nie wystarczają, potrzebne są również testy kodu
33. Nie budujmy mitu guru wokół osób bardziej doświadczonych.
34. Dłuższy czas pracy i większy nakład energii nieuchronnie sprawi że będziesz lepszy.
35. Raport błędu powinien zawierać informację jak go odtworzyć, jaki powinien być rezultat oraz co się dzieje aktualnie.
36. Jeśli coś nie jest potrzebne TERAZ, to tego nie twórz (YAGNI - You aren't gonna need it)
37. Pierwsze kroki w aplikacji muszą być klarowne i jasne.
38. Najpierw zmierzaj czas komunikacji między procesami, następnie weryfikuj swoje algorytmy.
39. Proces budowania aplikacji powinien być czysty.
40. Poznanie opcji poleceń może być bardzo pomocne.
41. Dobry programista poznaje różne paradygmaty.
42. Poświęć trochę czasu aby poznać IDE.
43. Powinieneś znać złożoność Twoich algorytmów aby stworzyć optymalny system.
44. Dekomponuj kodowanie na mniejsze.
45. Musisz nauczyć się estymować projekty.
46. Nie bój się kopiować jakiś kod do projektu tylko po to aby go zrozumieć.
47. Informacja o błędzie ma dotrzeć jak najszybciej

48. Linker nie jest magicznym narzędziem.
49. Większości rozwiązań tymczasowych zostaje w kodzie na zawsze
50. Warto najpierw tworzyć oczekiwania, potem implementację.
51. Zadbaj o to, aby aspekty niewidoczne zostały ujawnione.
52. Podczas programowania współbieżnego wystrzegaj się współdzielenia pamięci
53. Pisz kod tak, by w przyszłości ktoś się nim zachwycił.
54. Poprawne stosowanie polimorfizmu pozwala zachować czysty kod
55. Testerzy, którzy Ci raportują błędy Twojego kodu to Twoi najlepsi przyjaciele
56. Projekt powinien być budowany tylko raz.
57. Tylko kod mówi prawdę.
58. Poświęć czas na naukę budowania

PROGRAMOWANIE NIE JEST ZAKOŃCZONE DOPÓKI NIE DOSTARCZYMY DZIAŁAJĄCEGO PROGRAMU

59. Wypróbuj programowanie w parach.
60. Lepiej używaj typów domenowych, które określają zachowanie danego bytu.
61. Spróbuj przewidzieć błędy jakie mogą popełnić potencjalni użytkownicy.
62. Profesjonalny programista to osoba odpowiedzialna!
63. Trzymaj wszystko w systemach kontroli wersji.
64. Gdy masz problem i możesz go rozwiązać zrób sobie przerwę.
65. Odkrywanie kodu na nowo pozwoli rozwinąć umiejętności programistyczne.
66. Unikaj stosowania wzorca Singleton.
67. Używaj narzędzia do analizy wydajności.
68. Zasada jednej odpowiedzialności SRP
69. Zatrzymaj się, cofnij się i zautomatyzuj pracę, którą lubisz wykonywać.
70. Stosowanie statycznej analizy kodu pozwoli Ci znaleźć potencjalne błędy.
71. Testy powinny sprawdzać wymagania, nie implementację.

72. Testy powinny być precyzyjne i dokładne.
73. Testuj podczas nocy i weekendów
74. Testowanie jest inżynierskim wyborem wytwarzania oprogramowania.
75. Naucz się myśleć "stanowo"

Odnośnik:

Programowanie kontraktowe - metoda organizowania kodu źródłowego w taki sposób, aby wynikało z niego nie tylko jak program ma działać, ale też w jaki sposób zweryfikować poprawność działania konkretnych elementów programu.

76. Co dwie głowy to nie jedna.
77. Kod nigdy nie kłamie, ale może przeczyć sam sobie.
78. Programista rozwija się dzięki innym programistom.
79. Dobieraj odpowiednie algorytmy i struktury danych.
80. Stosuj zasadę DRY - Don't repeat yourself.
81. Pisz kod tak jakbyś miał go rozwijać przez resztę życia.
82. Klienci na początku mogą nie wiedzieć czego potrzebują.