Curso de Buenas Prácticas para Escritura de Código

Aprobaste el curso 😎

Felicidades, ya puedes acceder a tu diploma digital.

9,0	27
Mi Nota	Correctas

Resumen

1 ¿A quién beneficia contar con código bien escrito?

2 ¿En qué nos basamos para decir que un código es de alta calidad?

3 ¿Qué hace a la prolijidad del código?

4 ¿Cuáles de estos identificadores son mnemotécnicos, específicos y precisos?

5 ¿Cuál es la principal característica del código correctamente modularizado?

6 ¿Cómo se logra código reutilizable?

7 ¿Cómo se determina si un código está correctamente organizado?

8 ¿Cuál de estos no es un problema del hardcoding?

9 ¿Cómo puede evitarse el hardcoding?

10 ¿Cuál es el principal problema derivado de la existencia de efectos colaterales?

11 ¿Qué son los principios SOLID?

12 ¿Qué beneficios aporta usar los principios SOLID?

13 ¿Qué nos enseña el Single Responsibility Principle?

14 ¿Qué nos enseña el Open Closed Principle?

15 ¿Qué nos enseña el Liskov Substitution Principle?

16 ¿Qué nos enseña el Interface Segregation Principle?

17 ¿Qué nos enseña el Dependency Inversion Principle?

18 ¿Qué es un patrón de diseño?

19 ¿Cuántas instancias de un Singleton existen en una aplicación?

20 ¿En qué casos conviene utilizar el patrón Factory?

21 ¿En qué casos conviene utilizar el patrón Command?

22 ¿Para qué sirve el Testing Automatizado?

23 ¿Qué ventaja tiene el testing automatizado respecto del testing manual?

24 ¿Qué prueba el Unit Testing?

25 ¿Qué prueba el Integration Testing?

26 ¿Qué propone el Test Driven Development?

27 ¿Cuál de esos es un beneficio de usar Test Driven Development?

28 ¿Para qué sirve un Pull Request?

29 ¿Para qué sirve documentar nuestro código?

30 ¿Qué tan complejo es escribir código de alta calidad?

Regresar