UNIT TESTING GUIDELINES

1. Mantener las pruebas unitarias pequeñas y rápidas: Mantener las pruebas rápidas reduce el tiempo de respuesta del desarrollo.

2. Las pruebas unitarias deben ser automatizadas.

3. Haga que las pruebas unitarias sean fáciles de ejecutar: Que se pueda ejecutar con un solo comando o un click todo el conjunto.

4. Mida las pruebas: Aplicar análisis de cobertura para saber qué partes del código no se ejecutan.

5. Corrija las pruebas fallidas inmediatamente: Cada desarrollador debe ser responsable de que una nueva prueba se ejecute correctamente y todas las demás sigan andando bien.

6. Mantenga las pruebas a nivel de unidad: Debe haber una clase de prueba por clase ordinaria y el comportamiento debe probarse de forma aislada.

7. Empiece simple: Una prueba simple es mejor que no tener ninguna.

8. Mantenga las pruebas independientes: Las pruebas no deben depender de otras pruebas ni del orden de ejecución.

9. Mantenga las pruebas cerca de la clase que se está evaluando: Si la clase a probar es Foo, la clase de prueba debe llamarse FooTest y mantenerse en el mismo paquete (directorio) que Foo. Asegúrese que el entorno de compilación esté configurado para que las clases de prueba no lleguen a las bibliotecas de producción o ejecutables.

10. Nombre las pruebas correctamente: Cada método de prueba debe probar una característica distinta de la clase que se prueba y se deben nombrar los métodos en consecuencia.

11. Prueba la API pública: Algunas herramientas de prueba posibilitan probar el contenido privado de una clase, pero debe evitarse.

12. Piense en la caja negra: Pruebe si la clase cumple con sus requisitos.

13. Piense en la caja blanca: Probar la lógica más compleja.

14. Probar casos triviales.

15. Céntrese primero en la cobertura de ejecución: Distinguir entre cobertura de ejecución y cobertura de prueba real. Se debe garantizar una alta cobertura de ejecución. Cuando esto esté en su lugar, se debe mejorar la cobertura de la prueba. La cobertura

de prueba real no se mide facilmente.

16. Cubrir casos límite: Cubrir casos límite de parámetros. Para números, número negativo, 0, positivo, más chico, mas grande, infinito, etc. Para cadenas, cadena vacía, un solo carácter, cadena no ASCII, etc. Para colecciones probar vacía, con uno, primero, último, etc.

17. Proporciona un generador aleatorio: Generar parámetros aleatorios para que las pruebas se ejecuten con distintas entradas cada vez.

18. Pruebe cada función una vez: Probar exactamente la función indicada por el nombre del método de prueba. (Mantener baja la cantidad de código de prueba).

19. Utilice afirmaciones explícitas: Preferir assertEquals que assertTrue, ya que el primero da más información sobre qué falla.

20. Proporcionar pruebas negativas: Hacen un mal uso intencional del código y verifican la solidez y el manejo apropiado de errores.

21. Diseñe el código teniendo en cuenta las pruebas: Haga que los miembros de la clase sean inmutables estableciendo el estado en el momento de la construcción (reduce necesidad de setters). Restrinja el uso de herencia excesiva y métodos públicos virtuales. Reduzca la API pública usando clases de amigos (C++), alcance interno (C#) y alcance de paquete (Java). Evitar ramificaciones innecesarias. Mantener la menor cantida de código posible dentro de las ramas. Hacer uso intensivo de excepciones y aserciones para validar argumentos en API públicas y privadas, respectivamente. Restringir uso de métodos de conveniencia.

22. No se conecte a recursos externos predefinidos: Las pruebas unitarias deben escribirse sin un conocimiento explícito del contexto del entorno donde se ejecutan.

23. Conozca el costo de las pruebas: El estandar típico de cobertura de ejecución de la industria es 80%.

24. Priorizar las pruebas.

25. Preparar código de prueba para fallas: Prepárese siempre para que la falla de una sola prueba no reduzca la ejecución del conjunto.

26. Escribe pruebas para reproducir errores.

27. Conoce las limitaciones: Una prueba fallida puede indicar que el código tiene errores, pero una exitosa no prueba nada en absoluto.