**Cuestionario sobre "Código limpio" de Robert Cecil Martin:  
Prólogo:**  
  
1. ¿Qué propósito tiene el prólogo en el libro "Código limpio"?  
2. ¿Quién es el autor del prólogo y qué relación tiene con Robert Cecil Martin?  
3. ¿Cuál es la importancia de escribir código limpio según el prólogo?  
  
**Introducción:**  
  
4. ¿Cuál es el objetivo principal del libro "Código limpio"?  
5. Mencionar al menos tres beneficios de tener código limpio en un proyecto de desarrollo de software.  
6. ¿Por qué es importante el aspecto económico al considerar el mantenimiento de código limpio?  
  
**Sobre la imagen de cubierta:**  
  
7. ¿Qué representa la imagen de cubierta del libro?  
8. ¿Cuál es el mensaje que se intenta transmitir a través de la imagen de cubierta?  
  
**Capítulo 1: Código limpio:**  
  
9. ¿Cuál es la metáfora utilizada para describir el código limpio?  
10. Mencionar al menos tres características de un código limpio según el autor.  
11. ¿Cuál es la importancia de que el código sea fácil de leer?  
  
**Capítulo 2: Nombres con sentido:**  
  
12. ¿Por qué los nombres bien elegidos son importantes en la escritura de código?  
13. Mencionar tres pautas sugeridas por el autor para nombrar variables, funciones y clases.  
14. ¿Cuál es el efecto de utilizar nombres genéricos o ambiguos en el código?

1. El propósito del prólogo es el dar algunos conceptos y puntos de vista (como el japonés) para darnos una idea y empezar a entender bien lo que significa hacer código limpio.

.

1. El autor del prologo es James O. Coplien es un escritor, conferencista e investigador en el campo de la informática, es un colega de Robert Cecil con el cual colaboro para dar sus visiones sobre lo que es hacer código limpio.
2. El prologo habla sobre la importancia del codigo limpio, Crear código legible es tan importante como crear código ejecutable. Con todo esto lo que se busca es comercializar rápidamente los productos. Se quiere que las fábricas produzcan software a pleno rendimiento. Se trata de fábricas humanas, programadores que piensan, que sienten y que trabajan para crear un producto.
3. Aprender a crear código limpio es complicado. Requiere algo más que conocer principios y patrones. Tiene que sudar. Debe practicarlo y fallar. Debe ver cómo otros practican y fallan. Debe observar cómo se caen y recuperan el paso. Debe ver cómo agonizan en cada decisión y el precio que pagan por tomar decisiones equivocadas.
4. El código limpio tiene muchos beneficios para el desarrollo de software. En primer lugar, el código limpio hace que sea más fácil para otros desarrolladores trabajar en el mismo software, lo que acelera el tiempo de desarrollo y reduce errores.

Además, el código limpio hace que sea más fácil realizar futuras actualizaciones y mejoras en el software, lo que aumenta su vida útil.

El código limpio también mejora la calidad del software al crear una base sólida y clara sobre la cual construir. Esto significa que el software será más confiable, escalable y seguro.

1. Es importante considerar el aspecto económico del mantenimiento del código limpio ya que un código limpio presupone la escritura de un código de legible y entendible, lo que lleva a una buena y eficaz capacidad de mantenimiento lo que lo hace más económico al no haber tanto contratiempos a la hora de desarrollar código.
2. La imagen de la portada es M104: la Galaxia del Sombrero. M104 se encuentra en Virgo, a unos 30 millones de años luz, y su núcleo es un súper agujero negro que pesa aproximadamente mil millones de masas solares.
3. Lo que intenta dar a entender la portada es que El código limpio es el centro de todo tipo de desarrollo.
4. Grady mantiene las mismas ideas que Bjarne, pero adopta una perspectiva de legibilidad. Me gusta especialmente que el código limpio se pueda leer como un texto bien escrito. Piense en un buen libro. Recordará que las palabras desaparecen y se sustituyen por imágenes, como ver una película. Mejor todavía. Es ver los caracteres, escuchar los sonidos, experimentar las sensaciones. Leer código limpio nunca será como leer El Señor de los Anillos. Pero esta metáfora literaria no es incorrecta. Como una buena novela, el código limpio debe mostrar de forma clara el suspense del problema que hay que resolver. Debe llevar ese suspense hasta un punto álgido para después demostrar al lector que los problemas y el suspense se han solucionado de forma evidente.
5. En orden de prioridad, el código simple:

-Ejecuta todas las pruebas.

-No contiene duplicados.

-Expresa todos los conceptos de diseño del sistema.

-Minimiza el número de entidades como clases, métodos,

-funciones y similares.

11) La importancia de que el codigo sea fácil de leer viene de que si es fácil de leer será mas fácil escribir, constantemente se tendrá que leer codigo antiguo para poder del modo que si es fácil leerlo será mas fácil crear codigo nuevo, el poder leerlo fácilmente gracias al clean code hace que cuando se haga codigo nuevo se pueda avanzar mas rápido y se podrá terminar el codigo cuanto antes.

12) En el software, los nombres son omnipresentes. Aparecen en variables, funciones, argumentos, clases y paquetes. Asignamos nombres a archivos y a directorios, a archivos jar, war y ear. Usamos nombres constantemente. Por ello, debemos hacerlo bien. Es fácil afirmar que los nombres deben revelar nuestras intenciones. Lo que queremos recalcar es la importancia de hacerlo.

13)Elegir nombres correctos lleva tiempo, pero también ahorra trabajo. La elección de nombres que revelen intenciones facilita considerablemente la comprensión y la modificación del código.

Debemos evitar palabras cuyo significado se aleje del que pretendemos. Por ejemplo, hp, aix y sco son nombres de variables pobres ya que son los nombres de plataformas o variantes de Unix. Aunque se trate del código de una hipotenusa y hp parezca la abreviatura correcta, puede no serlo.

Las clases y los objetos deben tener nombres o frases de nombre como Customer, WikiPage, Account y AddressParser. Evite palabras como Manager, Processor, Data, o Info en el nombre de una clase. El nombre de una clase no debe ser un verbo.

Los métodos deben tener nombres de verbo como postPayment, deletePage o save. Los métodos de acceso, de modificación y los predicados deben tener como nombre su valor y usar como prefijo get, set e is de acuerdo al estándar de javabean.

Algunos nombres tienen significado por sí mismos, pero la mayoría no. Por ello, debe incluirlos en un contexto, en clases, funciones y espacios de nombres con nombres adecuados. Cuando todo lo demás falle, pueden usarse prefijos como último recurso.

14) El efecto que genera el uso de nombres genéricos tiene efectos negativos, como reducirla legibilidad y la comprencion del codigo, ya que los nombres no terminan de dar información sobre la tarea que cumplen.