Summaries

chapter 1 Clean Code

There Will Be Code

We will never be rid of code, because code represents the details of the requirements of the users.

Bad Code

We know good code matters because one code bad is an obstacle for a programmer.

The Total Cost of Owning a Mess

To modify a code, you need to know all the details of the consequences and the effects that may occur, since a code may have errors.

The Grand Redesign in the Sky

To modify a code or redesign it is necessary that the company invest budget for the new code, since the new project should do what the old one could no.

Attitude

To fix a disaster it is necessary that the team of programmers have a good attitude.

The Primal Conundrum

Programmers face a great situation since they are also aware that if the code is incorrect, their work be slow.

What Is Clean Code?

A clean code can be made in the simplest, since this could be said to be an elegant code, to the programmer.

Summaries chapter 2

Meaningful Names

In a software the names can appear within the variables, functions arguments, classes, directories or files such as jar, war or ear.

Use Intention-Revealing Names

For a code to be understandable, appropriate names must be used for each variable.

Avoid Disinformation

In this case, programmers should avoid leaving false clues, since it may be difficult to identify the word in this case.

Make Meaningful Distinctions

In a code the programmer must give names to each variable that makes sense in the development of the code.

Use Pronounceable Names

Give meaningful and coherent words for classes and variables.

Use Searchable Names

You should not use numbers as variables; however, you should use very simple words that facilitate your search.

Interfaces and Implementations

A user-friendly interface must be used.

Avoid Mental Mapping

In this case you must use the full names and not initials, because then you can forget.

Class Names

Avoid putting verbs to class names.

Method Names

Preferably for the methods you can use the names with verbs.

Use Solutions Domain Names

Use known names since these are the pattern of the code.

Capítulo 3 Funciones

Dentro de la programación es muy necesario que usemos las funciones ya que estoy nos da un mejor entendimiento dentro del código. En esto es muy necesario usar bien las instrucciones tales como un else, if ó while. Una función solo debe hacer una cosa y debe cumplir con todas las instrucciones que pida el código y debe hacerlo bien hecho, ya que esto ayuda mucho dentro de la codificación, una función se utiliza dentro de un código ya que estoy ayuda al programador a entender mejor el código ya que estaría dividido por fragmentos.

Es más fácil corregir un código hecho con funciones ya que estas estarán determinadas una sola cosa y mejoraría la estructura del código, también se debe utilizar nombres muy descriptivos en las funciones ya que cualquier programador lo puede entender.

También las funciones se pueden evitar poner muchos argumentos ya que seria necesario uno hasta dos y tres en caso especial ya que no es preferible utilizar, en el caso de que la función tenga mas de dos argumentos es muy difícil de entender ya que se le puede decir que es una función monádica, hay un caso especial de las triadas que tienen tres argumentos es valido pero llega el problema al momento de ordenar, en el caso de que la función tenga dos o más argumentos puede que una de estas sea propia, para entender mejor a la función se lo debe utilizar un verbo seguido por un sustantivo.

En una función se debe evitar los argumentos de indicador ya que son horribles y se pueden pasar como booleano, una función hace una cosa, pero a veces esta misma función hace coas ocultas dentro del código o es el completo de otras, también hacen algo o responden algo, pero nunca puede hacer ambas cosas para una función de error es mejor no separar los comandos es mejor no repetir el mismo argumento dentro de otra función.

Todo sistema se lo debe crear a partir de un leguaje especifico, el arte de la programación ha sido y será siempre, el arte del diseño del lenguaje.

Capítulo 4 Comentarios

Dentro de un código no es más útil que un comentario bien colocado, ya que a veces puede resultar beneficioso, a veces el programador es obligado a poner comentarios por motivos legales, a veces es bueno proporcionar un comentario para saber que va a pedir o devolver ese pedazo de código, el uso correcto de los comentarios permite compensar nuestra incapacidad para expresarnos en el código, ya que se puede decir que un comentario siempre son fallos, para dar mantenimiento a un software es necesario que los comentarios también deben estar actualizados.

Los comentarios son muy utilizados en los códigos incorrectos, pero a veces esto puede confundir mas al programador ya que esto no compensa al código que esta mal hecho o mal estructurado.

Pero para utilizar un comentario es mejor usar una palabra adecuada a la función, y si escribe asegurar escribir el código que lo rodea se puede decir que un comentario puede ser muy útil, pero a veces puede estar incorrecto.

En el caso de utilizar un comentario el programador debe avisar a su equipo para que todos entiendan ya puede tener problemas de que no sea necesario, ya que se usa un comentario si el programador esta con insuficiencia de información sobre el tema, evitar usar cometarios redundantes si usa uno escribir lo mas claro posible sino puede ser confuso.

Chapter 3 Functions

Within the programming it is very necessary that we use the functions since I am given a better understanding within the code. In this it is very necessary to use well the instructions such as an else, if or while. A function must only do one thing and must comply with all the instructions requested by the code and must do it well done, since this helps a lot within the coding, a function is used within a code since I am helping the programmer understand better the code since it would be divided by fragments.

It is easier to correct a code made with functions since these will be determined only one thing and would improve the structure of the code, you should also use very descriptive names in the functions since any programmer can understand it.

Also the functions can be avoided to put many arguments since it would be necessary one until two and three in special case since it is not preferable to use, in the case that the function has more than two arguments it is very difficult to understand since it can be To say that it is a monadic function, there is a special case of the triads that have three arguments is valid but the problem comes at the time of ordering, in case the function has two or more arguments it may be that one of these is proper, to a better understanding of the function should be used by a verb followed by a noun.

In a function the indicator arguments should be avoided since they are horrible and can be passed as a Boolean, a function does one thing, but sometimes this same function does hidden things within the code or is the complete one of others, they also do something or They answer something, but you can never do both for an error function. It is better not to separate the commands. It is better not to repeat the same argument within another function.

Every system must be created from a specific language, the art of programming has been and always will be, the art of language design.

Chapter 4 Comments

Within a code it is not more useful than a well placed comment, since sometimes it can be beneficial, sometimes the programmer is obliged to put comments for legal reasons, sometimes it is good to provide a comment to know that he will ask or return that piece of code, the correct use of comments allows us to compensate our inability to express ourselves in the code, since it can be said that a comment is always a failure, to maintain a software it is necessary that the comments must also be updated.

Comments are widely used in the wrong codes, but sometimes this can confuse the programmer more since this does not compensate the code that is badly done or poorly structured.

But to use a comment it is better to use a word appropriate to the function, and if you write ensure you write the surrounding code you can say that a comment can be very useful, but sometimes it may be incorrect.

In the case of using a comment, the programmer must notify his team so that everyone understands that they may already have problems that are not necessary, since a comment is used if the programmer is lacking information on the subject, avoid using redundant comments If you use one write as clearly as possible but it can be confusing.

Resumen capítulo 5

Formato

La función del formato. – En primer lugar, debe ser claro ya que en un código es muy importante.

Formato vertical. – En java en tamaño de los archivos esta relacionado con el tamaño de las clases.

La metáfora del periódico. –Oculta detalles y muestra conceptos, los elementos superiores del archivo debe proporcionar al nivel superior del algoritmo.

Densidad vertical. – La densidad vertical implica asociaciones, por lo tanto, las líneas de código con una relación directa deben aparecer verticalmente densas.

Distancia Vertical. – Esta regla no funciona no funciona con archivos independientes, es uno de los motivos por los que se debe evitar el uso de las variables protegidas.

Declaración de variables. – Deben declararse de la forma mas aproximada a su uso.

Variables de instancia. – se deben declararse en la parte superior de la clase, no debe aumentar la distancia vertical entre las variables.

Funciones dependientes. – Es mejor pasar la constante desde un punto en el que se tiene sentido de la posición en el que realmente se usa.

Afinidad Conceptual. – Cuanto mayor sea la afinidad, menor distancia vertical debe existir entre ellos.

Sangrado. – Cada nivel esta jerarquía es un ámbito en el que se puede declarar nombres en el que interprete instrucciones ejecutables, la versión sin sangrar por su lado es impenetrable.

Resumen capítulo 6

Objetos y estructuras de datos

Abstracción de datos. – Las interfaces permiten al usuario manipular la esencia de los datos sin necesidad de conocer su implementación.

Antisimétrica de datos y objetos. – La estructura de datos carece de funciones con significado, en ocasiones se requiere estructuras sencillas de datos con procedimientos que operen por sí mismas.

Ley de Deméter. – El método no debe invocar métodos de objetos devueltos por ninguna de las funciones permitidas.

Choque de trenes. – El uso de funciones de acceso implica el problema.

Híbridos. – Estos híbridos dificultan la inclusión de nuevas funciones o nuevas estructuras de datos.

Ocultar la estructura. – No preguntar sobre sus detalles internos.

Objetos de transferencia de datos. – Son estructuras muy especiales en este caso para conectarse con la base de datos.

Registro activo. – Son estructuras de datos son variables públicas.

Conclusión. – Los objetos muestran comportamientos y ocultan datos.

Summary chapter 5

Formatting

The function of the format. - First of all, it must be clear since in a code it is very important.

Vertical formatting - In java, file size is related to class size.

The newspaper metaphor. –Hide details and show concepts, the top elements of the file must provide at the top level of the algorithm.

Vertical Density. - Vertical density implies associations, therefore, lines of code with a direct relationship must appear vertically dense.

Vertical Distance - This rule does not work does not work with separate files; it is one of the reasons why you should avoid using protected variables.

Instance Variables - they must be declared at the top of the class, the vertical distance between the variables must not be increased.

Dependent Functions - It is better to pass the constant from a point where it makes sense of the position in which it is actually used.

Conceptual Affinity. - The higher the affinity, the less vertical distance must exist between them.

Bleeding - Each level this hierarchy is an area in which you can declare names in which you interpret executable instructions, the indented version by its side is impenetrable.

Summary Chapter 6

Objects and Data Structures

Data abstraction - The interfaces allow the user to manipulate the essence of the data without knowing its implementation.

Data/Object Anti-Symmetry. - The data structure lacks meaningful functions, sometimes simple data structures with procedures that operate by themselves are required.

The law of Demeter. - The method must not invoke object methods returned by any of the allowed functions.

Train Wrecks. - The use of access functions implies the problem.

Hybrids - These hybrids make it difficult to include new functions or new data structures.

Hiding structure. - Do not ask about your internal details.

Data transfer objects. - These are very special structures in this case to connect to the database.

Active Record- They are data structures are public variables.

Conclusion. - Objects show behaviors and hide data.