**Situation:** Explaining the structure of software programs to potential investors.

The goal of this assignment is to demonstrate the importance of defining unfamiliar terms when trying to explain something to an unfamiliar audience. Especially important is tailoring the details presented to the audience and situation, and understanding the level of detail required to best suit the situation.

**Parenthetical definition:** Refactoring is the process of changing the structure of a software program to make it easier to understand, without changing its function.

**Sentence definition:** Refactoring is a software practice that enables future readers of the software to understand and improve the code. Refactoring makes code easier to read, allows the code to be adapted to new features, and reduces the likelihood of any mistakes in the program propagating into other areas.

**Expanded definition:**

Although code refactoring has always been a process existing within software development, formal definitions only started to appear in the 1990s ("Code refactoring", 2015). Before that, refactoring was an informal process with no real guidelines or support. Once it became commonplace, many development environments started offering automated refactoring systems to eliminate potential errors, such as incorrect cut-and-pasting and forgetting to change the names of code pieces. Refactoring can occur at any stage of the process, and as many times as needed. It is most commonly used to make code readable for other people, and is used in all well-designed code that is intended to be read or modified.



This diagram demonstrates the use of refactoring in object-oriented design, where refactoring is quite useful (Koffman & Wolfgang, 2006). Object-oriented design is a system in which programming concepts are represented as "objects" that simulate real-world existences, and contain functionality related only to their purposes in the program.

Refactoring does not include improvements to either the functionality or the performance of the software. This would introduce the risk of modifying the code such that existing tests no longer work in the updated program. In this case, we would no longer know whether the errors were a result of mistakes in the refactoring or errors introduced with the updated functionality.

An example of refactoring would be the extraction of several lines of code that are repeated elsewhere in the program (Gallardo 2003). In one situation, a program might want to display a user's username, email address, and phone number on the user's profile page. Instead of having separate code that displays each of those things individually, refactoring would allow the developer to create a consistent method for displaying text on the page, and use that method as part of each piece of information. This means that any changes to the display method at a later date only have to be made in one place, and the developer does not have to remember all of the other places the same code is used.

References:

Code refactoring. (2015, April 7). Retrieved May 27, 2015 from Wikipedia: http://en.wikipedia.org/wiki/Code\_refactoring

Gallardo, D. (2003, September 9). Refactoring for everyone. Retrieved from https://www.ibm.com/developerworks/library/os-ecref/os-ecref-pdf.pdf.

Koffman, E., & Wolfgang, P. (2006). Refactoring the employee and student classes. In Abstraction, Data Structures and Design Using C++, 212. WIley & Sons, Temple University