1. In my opinion, the prediction will be suitable in the future. For a software, the best case is developing a software without accidental complexity, and it is impossible to remove essential complexity. From textbook, complexity in software is an essential property, not an accident. This complexity we speak of seems to be an essential property of all large software systems. By essential we mean that we may master this complexity, but we can never make it go away.
2. (A) Abstraction
   The goal of abstraction is reducing the complexity. In the refactored code, writer used the "Performance" class, implementing the same function with original code.
   (B) Encapsulation
   The goal of encapsulation is keeping changes local. In the refactored code, three "get" methods is deleted and only returns a message object, which hides the information. Encapsulation is achieved by information hiding.
   (C) Hierarchy
   It's obvious that the refactored code is using hierarchy, but the original code does not use. Manager extends Employee, because there may be some common elements can go into a base class. The advantage is that other classes can also reuse the base class.
   (D) Separation of concern
   The refactored code moves the Character (Vector characters) from Character to ChracterMamager, and it can store a group of characters, and it's also able to add new ones. And the refactored code implements a Character class which can deal with all attributes of an object.
   (E) No design improvement made
   The user only needs to use range instead of calling unnecessary methods. However, this method did not reduce the complexity, hide the secrets, and did not make changeable/reusable. Thus, no design improvement made.
   (F) No design improvement made
   The refactored code adds the error message after catching an exception, and nothing else has changed.
   (G) Modularity
   The refactored code adds a class which contains createCPU and createMMU, and call those two methods in Emulator class, making this code changeable and reusable.
   (H) Encapsulation
   The refactored code change sthe public class to private class, therefore, the variables cannot be directly accessed from other classes. And Encapsulation is achieved by information hiding.
   (I)No design improvement made
   The refactored code looks simpler than original code, but the refactored code does not improve the code, such as making changeable/reusable, hiding information, reducing the complexity, and providing structure to dependencies.
   (J) Modularity
   Separating one package to two package makes the code changeable and reusable, which is more convenient than the original code.