**Review Ques 7**

**Patterns:**

1. What is a visitor pattern and why is it useful?
2. Give an example where factory pattern is suitable in class design. How could factory be usefully combined with visitor.
3. What is a singleton pattern? Give an example. What is the connection between factory and singleton?
4. What is coupling and cohesion among classes? Ideally, how should design alter these values? Justify your answer.
5. What is the layered patterns? Name one advantage and one disadvantage for layers.
6. What is the proxy pattern? What does it offer to coupling? What does it offer to layers?
7. A body has several limbs. Limbs have extremities (hands or feed). Extremeties contain toes or fingers.
   * Which design pattern is similar to this structure?
   * Briefly describe the components of that design pattern.

**Anti patterns:**

1. What is a god class? How to recognize it? How to fix it?
2. For OO purists, "type switching" statements are bad. What is an example of such a statement. How to fix it? What general principle of OO is demonstrated by that fix?
3. Do all bad smells really matter? Explain.
4. Give two examples of each of the following bad smells:
   * Bloating
   * Dispensable
   * OO abusers
5. Below are a list of OO designs and Arthur Riel's OO heuristics. Describe how to fix the design so that it agrees with the heuristic:
   * **Design:** One central class has 20 distinct methods for account creation, 60 distinct methods for account auditing, 50 distinct methods for writing objects to a SQL database.  
     **Heuristic 3.3** Beware of classes that have many accessor methods defined in their public interface, many of them imply that related data and behavior are not being kept in one place.
   * **Design:** You need to use a commercial API that highly couples the business logic and GUI layers.  
     **Heuristic 3.5:** In applications which consist of an object-oriented model interacting with a user interface, the model should never be dependent on the interface. The interface should be dependent on the model.
   * **Design:** In your math API, integer class inherits from fractions class.  
     **Heuristic 5.1** Inheritance should only be used to model a specialization hierarchy.