1. What is a difference between Association, Aggregation and Composition in UML class diagram notation? (5 points)

   Association – type of relationship between connected objects. There are flexible and directed associations.
   Aggregation – type of relationship between classes with part/whole relationship, where the "part" is not an integral part of whole. It indicates if the whole object is destroyed, then none of other parts associated with the whole object are deleted (they continue to exist).Example: object TV inside object Room.
   Composition - – type of relationship between classes with part/whole relationship, where the "part" is actually an integral part of whole. It indicates if the whole object is destroyed, then all other parts associated with whole object are deleted as well. Example: object Room inside object Building, since there is no way for room to exist without being in building.
   <Composition(the most narrow definition)> is included into <aggregation> is included into <association(the widest definition)>

2. Explain Liskov substitution principle in detail. Provide an example of violation of Liskov's principle and the way of fixing it. (5 points)

   The purpose of this principle is that derived classes can be used instead of the parent classes from which they are derived without disrupting the program. For example, derived class in override method change logic of method by throwing a new exception. Then the solution is to change method to make it work properly.

3. Which pattern defines a placeholder for another object without changing its interface? (5 points)

   Strategy pattern

4. When is it better to use abstract class instead of the interface? (5 points)

   They have different logic: an abstract class identify the object itself, while an interface identify it's behavior.
   It is better to use an abstract class when we have common realization for some methods therefore there is no necessity to copy the same logic every time. If there is necessity in unique for every class method realization, then the method might be marked as an abstract. But if all methods is supposed to be abstract, then the interface usage is more recommended.
   Secondly, an abstract class might have fields, therefore if there is necessity in such fields application, then the usage of an abstract class is preferable.