**1. Quais as regras para implementar uma classe imutável em Java?**

The class must guarantee that after an attribute is set, a method is defined or even the class itself is generated, it will never change anymore. To guarantee that, the attribute, method and class must all be set to final. Furthermore, as the object state cannot be changed, no setters (or related) are allowed. The getters should return a defensive copy of the object’s attributes being requested instead of a reference to it. It is good that the class has only one constructor and validates it’s attributes properly when set (as they can nevermore be changed).

**2. Explique por que não devemos especificar, em um mesmo diagrama UML, um atributo na classe que representa também o nome de papel (rolename) em uma associação.**

If that is done, when the code is generated, we’d have a duplicate attribute. So, just one should be chose (preferably in the rolename).

**3. Explique navegabilidade entre relacionamentos UML. Como ela deve ser interpretada no momento da codificação?**

The cardinality, rolename, and type of relationship must all be respected.

**4. Explique a multiplicidade ou cardinalidade entre relacionamentos UML. Como ela deve ser interpretada no momento da codificação?**

The cardinality of the object will be the number farthest to it.

**5. Qual a diferença entre API, biblioteca e framework?**

The API are the methods/functions in a library that you can call. It is the interface to the library.

A library is a collection of codes that you can use to make it easier/faster to code. It is usually a specific to a certain area, like processing images, math operations…

A framework is a collection of libraries which provide a wide range of applications compared to a library.

* 1. **6. Explique os relacionamentos (UML): dependência, associação, agregação e composição.**
  2. **a. Quando cada um deve ser utilizado?**

Dependency is used when call only one method or get a single attribute. In other words, it is a quickly usage of another class. Client -> Service.

Association is stronger than dependency, requires that you use more the other class, it is not limited to just one call for example. Student -> Teacher

Aggregation is a “whole-part” weak relationship. For example Teacher -> Department. A teacher HAS a department, when department cease to exist, the Teacher keeps its existence.

Composition is also a “whole-part” strong relationship. For example Room -> House. A House possess a Room, when the House is destroyed, the Room cease its existence.

* 1. **b. Como cada um deve ser implementado?**

Dependency is always uniderctional. Composition cardinality, related to whole -> part, is always one, because the part cannot be shared with other “wholes”.

**7. O que é uma classe serializada em Java? Qual a sua função?**

Is a class that implements the interface Java.io.Serializable. To serialize an object means to convert its state to a byte stream so that the byte stream can be reverted back to a copy of the object.

**8. O que é um objeto persistente?**

Is a object that is somehow saved. So that it outlives the process that created it.

**9. O que é um objeto transiente?**

Is a object that is not transferred, they are lost intentionally. Which means, they are not serializable.

**10. Explique o padrão de projeto Repository.**