**Rationale of my implementation**

**Project structure**

I structured this project in 3 packages, one containing each other, besides the test. One package contains everything related with instruments and his management. It also contains a different subpackage, containing everything related with modules. Finally there are a third package inside modules that represents the different commands that we can apply to these modules. I decided to put one inside other one to represent better the class diagram structure.

**Patterns and SOLID principles used**

**Command design pattern**

For apply operations to different modules that makes the instrument, I propose a implementation using Command design pattern. The idea of have a different class for each different behavior allows us to fulfills the different solids principles and make our system extensible and easy to maintain. It also abstract the behavior from the entities and allow us to apply one behavior to more than one entity in runtime execution.

**Open/Close principle**

Instrument contains a list of commands and a list of modules where this commands will be applied. To add new functionalities to existing instrument, we just need to insert a new command or module in the instrument. Our previous code will be isolated and not affected, and as all new commands fulfill our contract, they are compatible with existing code.

**Single responsibility**

Our command will just do one operation over a given element.

**LISKOV replacement principle**

In my proposal is any extension of an existing class, so we are not using this principle.

**Interface Segregation Principle**

Other SOLID principle used was the interface segregation principle. As you can see in the code skeleton, all the objects that manages classes instances has just interfaces. These interfaces just contains the real needed public methods and all other things are encapsulated in the classes that implements this interfaces or in abstract classes for common things.

**Dependency inversion principle**

Dependencies (in our case modules) are passed in the concrete commands by injection. In our commands we pass the modules where the command will be applied as parameter in the constructor.