**Class Practice :**

You are designing an intruder detection system that has following modules (here module refers to one or more classes)

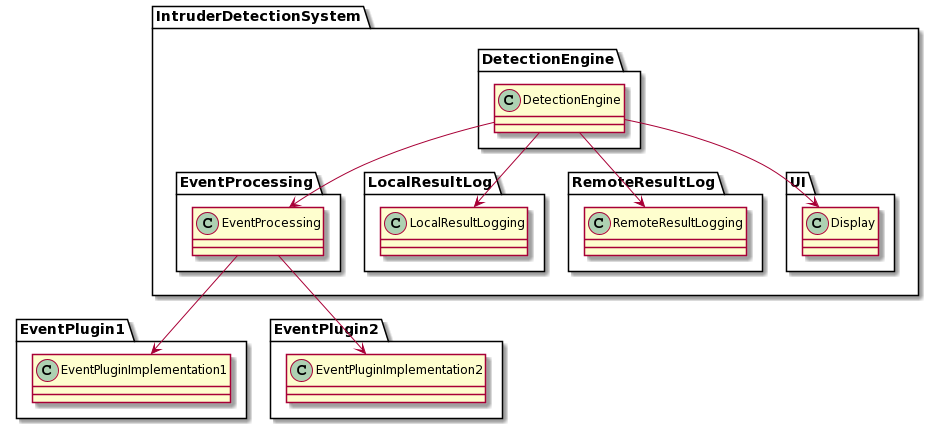
**EventProcessing**: This module processes events coming from EventPlugins and makes them available to the DetectionEngine module for analysis.

**EventPlugins**: These are external components that can be plugged into the system. They provide events which are then processed by the system to detect network intrusion.

**DetectionEngine**: This module implements actual algorithms that detect abnormal activity in the network based on available events.

**ResultLog**: These are two modules that log the detected intrusions locally on hard disk as well as send them remotely to the cloud.

A naive design is the one presented in the UML diagram below.



Your task is to improve this design keeping in mind SOLID design principles. You can add interfaces in any package if required.

After the improvement the design should reflect following characteristics:

1. High level modules should not depend on lower level modules.(hint: DetectionEngine is the highest level module, think about others yourself)
2. Interface with plugins to be designed such that new plugins can be added to the system without a need to make changes to the system
3. Currently there is only two types of logging available, the new design should support adding additional logging modules without impacting the DetectionEngine module
4. Show all relationships (e.g. inheritance, directional association between classes and interfaces)

**Solution:**

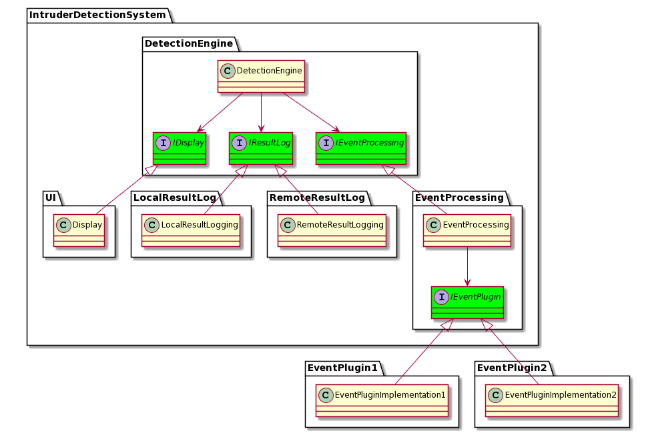
The solution involves adding interfaces in appropriate packages and showing inheritance and association relationships accordingly.

Following principles are applied in the solution:

Open Closed Principle

Dependency Inversion Principle

Note that all the arrows/dependencies are from low level to high modules

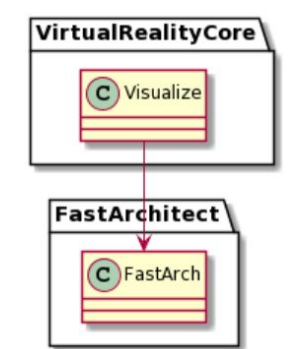


Question #2:

You are building a software system that helps visualize the architecture of a house for the customers. Your system is a virtual reality system that accepts an architecture from a software developed by a third party

vendor (software supplier) and visualizes it as a virtual reality house. The customer can walk into the house to get a feel of it using the VR glasses. At this stage you are working with only one vendor and developing the system. The initial design of your system looks like below where your system “VirtualRealityCore” module interacts with the third party software “FastArchitect” module to extract architectural information.

Note: In future you would be required to integrate with software developed by other vendors.



Modify the design to also include integration with another software “CleanArchitect” from a

different vendor. How can you ensure that the “VirtualRealityCore” module does not directly depend

on the software developed by other vendors? Note that you can not modify the software packages

developed by other vendors. Also do not worry about the internal design of the “VirtualRealityCore”

module of your system.

**Solution:**

This design is not appropriate because it makes the VirtualRealityCore module directly depend on the FastArchitect module. As we will be required to integrate other modules in future, we need to remove this dependency using dependency inversion.

Define an interface in the VirtualRealityCore module and implement it in another module “ArchInterface” to invert dependency for VirtualRealityCore module.

