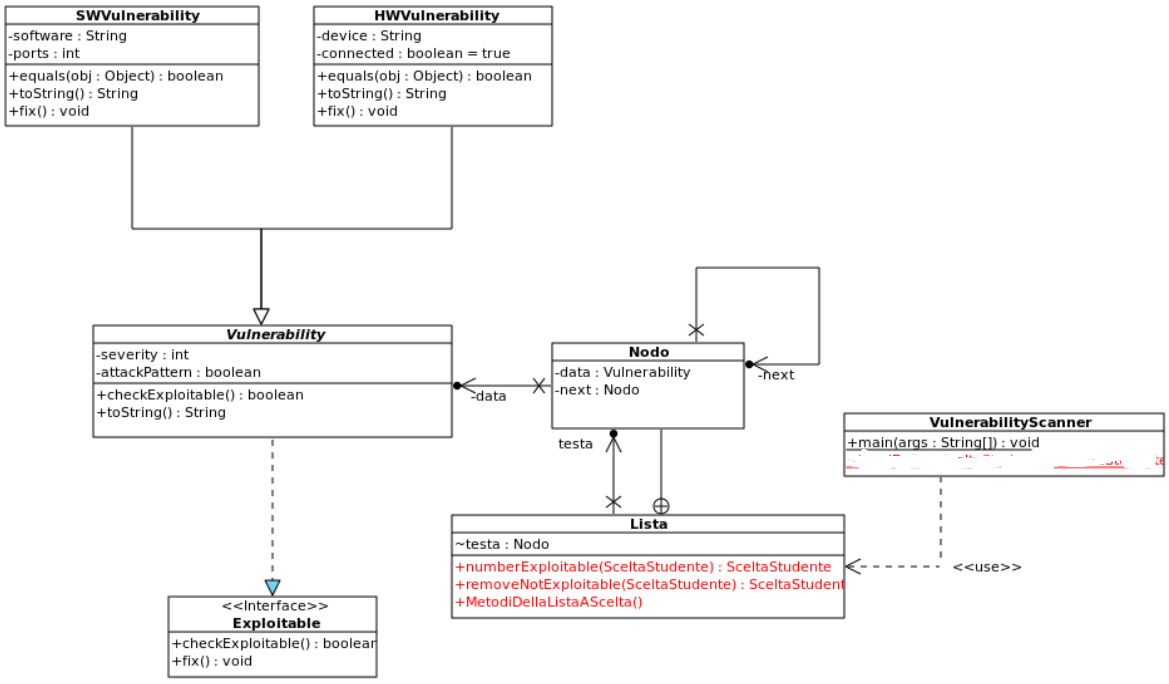
Corso di Programmazione A.A. 2024/2025

Esercitazione del 4 Giugno 2025

Prof. Domenico Amalfitano

Si vuole implementare uno scanner delle vulnerabilità di un sistema. Una vulnerabilità può essere exploitable, ossia può essere sfruttata da un utente malevolo tramite un attack pattern. Inoltre, una vulnerabilità può essere risolta, e il metodo di risoluzione dipende dal tipo specifico di vulnerabilità. Il software è modellato dal diagramma UML riportato di seguito. **N.B.** prestare attenzione agli attributi delle classi e di eventuali valori iniziali.



Requisiti del software.

1. Lo scanner utilizza una lista di vulnerabilità che si specializzano in hardware (HW) e software (SW).
2. La lista inserisce una nuova vulnerabilità sempre nell'ultima posizione.
3. Il metodo `checkExploitable` verifica se una vulnerabilità è exploitable, ovvero quando il livello di severity è maggiore o uguale a due ed esiste un attackPattern per quella vulnerabilità.
4. Il metodo `fix()` permette di risolvere una vulnerabilità e nello specifico.
    a. Una vulnerabilità HW è risolta disconnettendo il dispositivo, `connected= false`.
    b. Una vulnerabilità SW è risolta chiudendo i suoi port di comunicazione `ports=0`.
5. Due vulnerabilità HW sono uguali se riguardano lo stesso dispositivo, mentre due vulnerabilità SW sono uguali se riguardano lo stesso software.
6. Il metodo `numberExploitable` restituisce il numero di vulnerabilità SW e il numero di vulnerabilità HW che sono exploitable.
7. Il metodo `removeNotExploitable` rimuove dalla lista tutte le vulnerabilità che non sono exploitable. Di fatto dopo l'esecuzione del metodo, nella lista devono essere presenti solo vulnerabilità exploitable.
8. Nel diagramma UML mancano i costruttori e i metodi di get e set degli attributi. Lo studente implementi i metodi che ritiene necessari.
9. È cura dello studente, inoltre, dotare la classe Lista dei metodi canonici a supporto dell'implementazione dei metodi richiesti.
10. Si implementi la classe utente, `VulnerabilityScanner`, per testare le funzionalità del sistema tenendo conto anche di diversi scenari d'uso.