Présentation du squelette du programme Java avec : Nessus uniquement

Main class : SCANAPP

Rôle : classe principale qui démarre l'application et lance un scan Nessus.

Attributs :

- Aucun (classe d'exécution).

Méthodes (entêtes) :

- `main(String[] args)` — point d'entrée.
- `runScan(String target)` — lance le scan pour la cible (IP, URL ou CIDR).

Classe : SCAN (ScannerRun)

Rôle : représente une exécution de scan (un run).

Attributs :

- `id` : identifiant unique.
- `tool` : nom de l'outil utilisé (ex. `"Nessus"`).
- `target` : cible scannée (IP, domaine, ou CIDR).
- `startTime` : date et heure de début.
- `endTime` : date et heure de fin.
- `status` : état (`running` | `done` | `error`).

Méthodes (entêtes) :

- getters / setters.

- `updateStatus(String newStatus)`.

Classe : FINDING

Rôle : représente une faille détectée par Nessus.

Attributs :

- `id` : identifiant unique.
- `scannerRunId` : référence vers le SCAN qui a produit la faille.
- `severity` : gravité (`LOW` | `MEDIUM` | `HIGH` | `CRITICAL`).
- `title` : titre court de la faille.
- `description` : description détaillée.
- `target` : hôte/port affecté (ex. `192.168.1.12:22`).
- `cve` : identifiant CVE (si disponible).
- `evidence` : preuve textuelle (ex. `plugin_output`).
- `rawData` : export brut du rapport (JSON/XML).
- `createdAt` : date d'enregistrement.

Méthodes (en têtes) :

- getters / setters.
- `summary() : String` — résumé court.

Classe : NessusService (service principal)

Rôle : pilote l'API Nessus : création, lancement, suivi, export et récupération des rapports ; convertit ensuite en Findings puis sauvegarde en base.

Attributs :

- `apiUrl : String` — ex. `https://localhost:8834`.

- `apiToken : String` — jeton/session d'authentification.
- `db : DatabaseManager` — gestion simple de la base.

Méthodes (entêtes) :

- `login(String user, String pass) : boolean` — authentification.
- `createScan(String name, String target) : int` — crée le scan, retourne `scanId`.
- `launchScan(int scanId)` — démarre le scan.
- `checkStatus(int scanId) : String` — renvoie le statut (`running` / `completed` / `error`).
- `exportReport(int scanId) : int` — demande l'export, retourne `fileId`.
- `downloadReport(int scanId, int fileId) : String` — télécharge et retourne le chemin du fichier.
- `parseAndSave(String filePath, ScannerRun run)` — parse et enregistre les findings.

Classe : Database Manager

Rôle : gère la connexion à MySQL et les opérations d'enregistrement simples.

Attributs :

- `connection` : objet de connexion JDBC (ou paramètres de connexion).

Méthodes (entêtes) :

- `connect()` — ouvre la connexion.
- `saveScan(ScannerRun run) : long` — insère le run et retourne son id.
- `saveFindings(List<Finding> findings, long runId)` — insère les findings.

- `updateScanStatus(long runId, String status)` — met à jour le statut.
- `close()` — ferme la connexion.

Classe : NessusParser

Rôle : convertit le rapport exporté par Nessus (`.nessus` / JSON) en objets `Finding`.

Méthodes (entêtes) :

- `parseXml(String filePath, ScannerRun run) : List<Finding>`