Projet Java (POO) — Système de gestion de la sécurité et d'accréditation pour la CAN 2025

1. Objectif

Réaliser une application console de gestion de la sécurité de la CAN 2025 permettant de gérer les badges d'accréditation, les zones sécurisées, les accès et les contrôles, avec suivi des incidents, validation des droits et blocage des badges.

2. Exigences techniques (obligatoires)

2.1 Héritage & polymorphisme

- Classe abstraite UtilisateurCAN (id, nom, categorie) + 3 sous-classes : Spectateur, Staff, VIP.
- Méthode abstraite getNiveauAcces() dans UtilisateurCAN (niveau différent selon le profil).
- Stocker et manipuler les utilisateurs via ArrayList<UtilisateurCAN> (polymorphisme réel).

2.2 Interfaces

- Interface Accreditable : genererBadge(), invaliderBadge(), estBadgeValide().
- Interface Securisable : autoriser(Zone), refuser(Zone), verifierAcces(Zone).
- Badge implémente Accreditable ; ZoneSecurisee et ControleAcces utilisent Securisable.

2.3 Exceptions personnalisées (au moins 4)

- AccesRefuseException (zone non autorisée pour l'utilisateur).
- BadgeInvalideException (badge expiré, bloqué ou introuvable).
- ZoneInterditeException (zone inexistante ou interdite).
- DonneeInvalideException (nom vide, catégorie inconnue, dates invalides, etc.).

2.4 Identifiants uniques + static

- UtilisateurCAN : id auto-incrémenté via compteur static.
- Badge : numeroBadge auto-incrémenté via compteur static.

2.5 Collections + tri/recherche

- Stockage en collections pour utilisateurs, badges, zones et logs d'accès.

- Rechercher par numéro de badge, nom, zone ; lister accès par utilisateur/zone/date.
- Au moins un tri (par date des accès, par niveau d'accès, ou par nom).

3. Modèle minimal attendu

- ZoneSecurisee : idZone, nomZone, niveauRequis.
- UtilisateurCAN (abstraite) : id, nom, categorie + badge associé.
- Badge : numeroBadge, utilisateur, dateDebut, dateFin, statut.
- AccesLog : dateHeure, badge, zone, resultat (OK/REFUS).
- SecurityService : opérations métier (créer badge, vérifier accès, bloquer badge, rapports).

4. Règles métier obligatoires

- Accès : un utilisateur ne peut accéder à une zone que si son niveau d'accès >= niveau requis.
- Badge : un badge expiré ou bloqué entraîne un accès refusé.
- Un badge est unique et ne peut être associé qu'à une seule personne.
- Seul Staff (catégorie sécurité) ou AdminSécurité (optionnel) peut bloquer/débloquer un badge.

5. Interface console (menu minimal)

- 1. Ajouter utilisateur (spectateur/staff/VIP)
- 2. Ajouter zone sécurisée
- 3. Générer badge
- 4. Vérifier accès (badge + zone)
- 5. Bloquer / débloquer badge
- 6. Afficher historique des accès
- 7. Rapports (taux de refus, zones les plus sensibles, badges bloqués)
- 8. Quitter

6. Scénarios de test obligatoires

- Créer au minimum 20 utilisateurs (spectateurs, staff, VIP) et 8 zones de niveaux différents.
- Générer des badges avec des dates de validité variées (valide, expiré).
- Simuler au moins 25 tentatives d'accès (OK et REFUS) et enregistrer les logs.
- Bloquer au moins 3 badges et vérifier que l'accès devient systématiquement refusé.
- Provoquer au moins 4 erreurs : badge invalide, zone interdite, accès refusé, données invalides.

7. Livrables

- Code Java (packages propres, encapsulation, toString, gestion d'erreurs via exceptions).
- Mini-UML (1 page) montrant les classes principales, héritages, associations et interfaces.
- Scénario de test dans Main conforme à la section 6, avec sorties console explicites.
- Rapport 1–2 pages (choix techniques, difficultés rencontrées, pistes d'amélioration).

8. Bonus (+2 max)

- Sauvegarde/chargement (CSV/JSON).
- Tri multi-critères via Comparator.
- HashMap pour accès rapide + statistiques avancées.