Module : Composant Android (M205)
TP N° 10

Année de Formation 2024/2025
Filière : Développement Mobile Groupe : DEVOAM 201 - DEVOAM 202 Niveau : 2ème année

Exercice 1 :

Créer une classe **MathUtils** contenant les méthodes suivantes :

- add(a: Int, b: Int): Int : retourne la somme de deux entiers.
- subtract(a: Int, b: Int): Int : retourne la différence entre deux entiers.
- divide(a: Int, b: Int): Double : retourne le résultat de la division de a par b. Si b vaut 0, une exception IllegalArgumentException doit être levée.
- isEven(n: Int): Boolean : retourne true si le nombre est pair, sinon false.

Écrire une classe de tests avec **JUnit 5** et utiliser des assertions pour :

- Vérifier que les résultats des méthodes add et subtract sont corrects.
- Valider que divide fonctionne correctement pour des valeurs valides et qu'elle lève une exception pour une division par 0.
- Tester la méthode isEven pour des valeurs paires et impaires.

Utiliser plusieurs types d'assertions, comme :

- assertEquals
- assertThrows
- assertTrue / assertFalse

Exercice 2 :

1. Créez une fonction **`validerMotDePasse`** qui prend une chaîne de caractères en paramètre et retourne **`true`** si elle respecte les règles suivantes :
    - Le mot de passe contient au moins 8 caractères.

- Il contient au moins une lettre majuscule.
- Il contient au moins un chiffre.

2. Ajoutez des assertions pour valider chaque règle dans la fonction.
3. Testez votre fonction avec des mots de passe valides et invalides.

Exercice 3 :

- **Créer la classe ProductManager :**
  - Implémentez les fonctionnalités suivantes :
    - addProduct(name: String, price: Double) : Ajoute un produit à la liste. Le prix doit être positif.
    - removeProduct(name: String) : Supprime un produit par son nom.
    - getProductPrice(name: String): Double? : Retourne le prix d'un produit, ou null s'il n'existe pas.
    - calculateTotalPrice(): Double : Retourne la somme des prix de tous les produits.
- **Écrire les tests unitaires :**
  - Créez une classe de test **ProductManagerTest**.
  - Écrivez des tests pour couvrir les cas suivants :
    - Ajout de produits valides.
    - Tentative d'ajout d'un produit avec un prix négatif.
    - Suppression d'un produit existant.
    - Recherche du prix d'un produit.
    - Calcul du prix total des produits.

Exercice 4 :

Créer une classe **Book** avec les propriétés suivantes :

- title: String : le titre du livre.
- author: String : l'auteur du livre.
- isAvailable: Boolean : indique si le livre est disponible (par défaut, true).

Créer une classe **Library** qui offre les fonctionnalités suivantes :

- Ajouter un livre à la bibliothèque (addBook(book: Book)).
- Rechercher un livre par titre (findBookByTitle(title: String): Book?).
- Marquer un livre comme emprunté (borrowBook(title: String): Boolean).
    - Si le livre existe et est disponible, il devient indisponible, et la méthode retourne true.
    - Si le livre est introuvable ou déjà emprunté, la méthode retourne false.

Écrire des tests unitaires pour valider le comportement de chaque méthode :

- Vérifier que les livres sont ajoutés correctement.
- Tester la recherche de livres par titre (cas où le livre est trouvé et non trouvé).
- Valider le comportement de l'emprunt (livre disponible, livre déjà emprunté, livre introuvable).

Utiliser des assertions comme :

- assertEquals, assertNotNull, assertNull et assertFalse / assertTrue…