MISSION 2 ***Mise à jour***

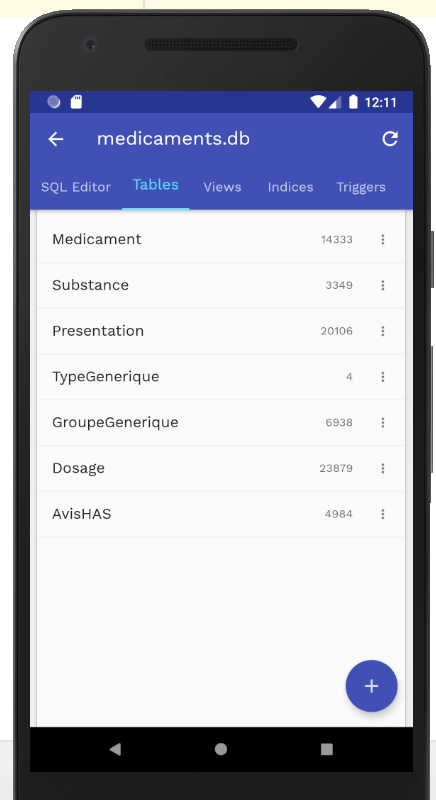
***TRAVAIL A FAIRE***

**3)** Le code CIS (Code Identifiant de Spécialité) pour un médicament correspond à un code numérique de 8 chiffres qui permet d’identifier un médicament quel que soit sa présentation.

**4)** Le code CIP (Code Identifiant de Présentation) est un code numérique à 7 ou 13 chiffres qui permet d’identifier la présentation d’un médicament.

**5)** **Compter le nombre de médicaments et le nombre de présentations. Les deux résultats sont ils cohérents ?**

Selon la capture ci-dessous, on peut voir qu’il y a 14333 médicaments et 20106 présentations.



Après analyse des définitions de CIS et CIP, on peut dire que le résultat est cohérent car les médicaments se déclinent en plusieurs présentation.

**6)** **Calculer, en une seule requête, le ratio nombre de présentations / nombre de médicaments. Attention, le résultat d’un COUNT est une valeur entière qu’il faut caster en valeur réelle pour que la division fonctionne**

SELECT COUNT(code CIP)\*1.0 / COUNT(codeCIP7)

FROM Medicament

INNER JOIN Presentation

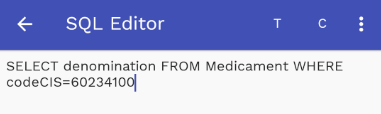
ON Medicament.etatCommercialisation = Presentation. EtatCommercialisation ;

**7) Rechercher le médicament dont le codeCIS est 60234100.**

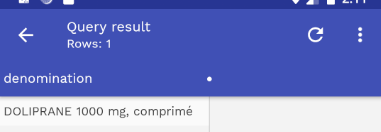
SELECT denomination FROM Medicament

WHERE codeCIS = 60234100 ;

Dans SQLite, on exécute la requête :

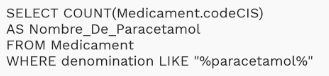


>> Résultat :



**8) & 9)** **Compter les médicaments qui ont « Paracétamol » et « Paracetamol » (sans le caractère accentué) dans leur dénomination.**

\*\* Requête le nom du médicament **sans accent :**

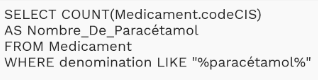


>> Résultat :

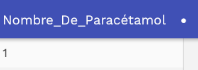


\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*\* Requête le nom du médicament **avec accent :**



>> Résultat :



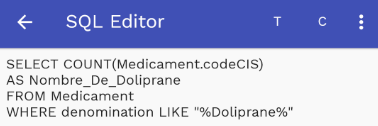
**10)** **Proposer une requête unique qui répond aux deux questions précédentes.**

**11) Existe-t-il un médicament qui a plus d’un codeCIS ?**

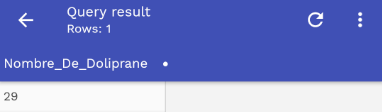
Un code CIS est un code unique a chaque médicament, donc il ne peut pas y en avoir plusieurs.

**12) Rechercher les médicaments qui ont Doliprane dans leur dénomination. Vérifier que le résultat est bien équivalent quelle que soit la casse de la chaîne recherchée (majuscule, minuscule doivent retourner le même résultat).**

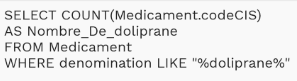
\*\* Requête D majuscule et le reste du nom en minuscule donnant : « Doliprane »



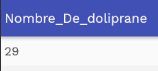
>> Résultat :



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\*\* Requête avec « doliprane » tout en minuscule :

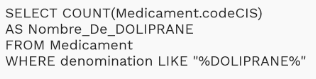


>> Résultat :

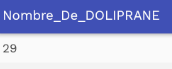


**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

\*\* Requête avec doliprane tout en majuscule :

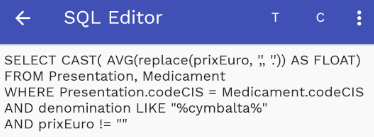


>> Résultat :

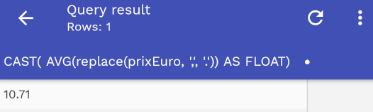


**13) Exécuter la requête suivante et donner son fonctionnement dans le détail.**

Exécution de la requête :



>> Résultat :



**Détail de la requête :**

Cette requête calcule la moyenne des prix des médicaments dont le nom contient Cymbalta et ayant un prix indiqué dans le champ prixEuro.

Ces fonctions ci-dessous s’exécutent entre les tables Présentation et Medicament :

\*\* CAST() permet de convertir une donnée d’un type à un autre

Dans cette requête, CAST convertit la colonne prixEuro de type TEXT en FLOAT

\*\* replace() est une fonction qui remplace dans la colonne prixEuro les virgules en points.

**14) Rechercher toutes les présentations du médicament 60234100. A qui sont destinées les deux présentations ?**

Requête :



>> Résultat :



Comme on peut le voir dans la capture ci-dessus : Ces deux présentations sont destinées à

SAFONI Aventis France

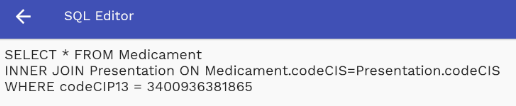
**Compte rendu du 19/09/2020**

**en équipe sur Discord**

**15) Retrouver dans la base de données l’ensemble des informations de présentation et du médicament associé à l’image *(montré dans l’énoncé)***

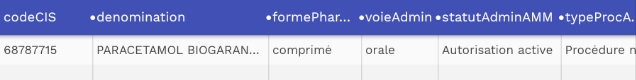


Requête :

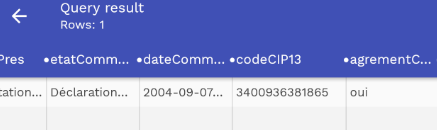


>> Résultat :

Extrait de la présentation



Dans la colonne codeCIP13, nous avons qu’il s’agit du bon code comme indiqué sur l’énoncé



**17)**

**a) Donner les informations du médicament correspondant à la présentation dont le code CIP13 est 3400931863014.**

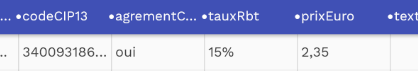
Requête :



>> Résultat :

Extrait de la présentation





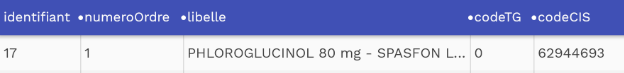
**b)** **Retrouver le numéro du groupe générique correspondant à ce médicament (ce devrait être 17)**

Étant donné que le codeCIS est unique pour chaque médicament, on a décidé d’utiliser ce code *(encadré en vert)* pour identifier le numéro du groupe générique du médicament présenté ci-dessus

Requête :



>> Résultat :

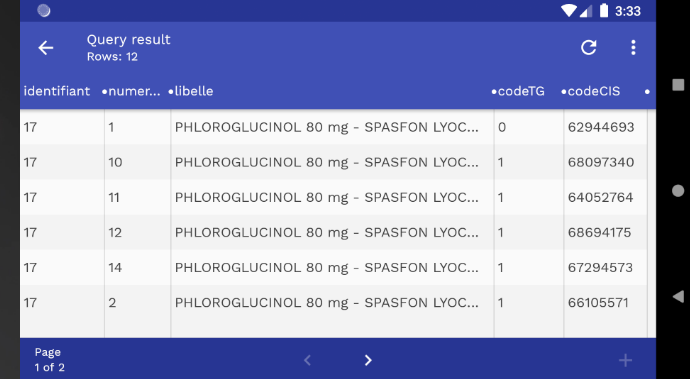


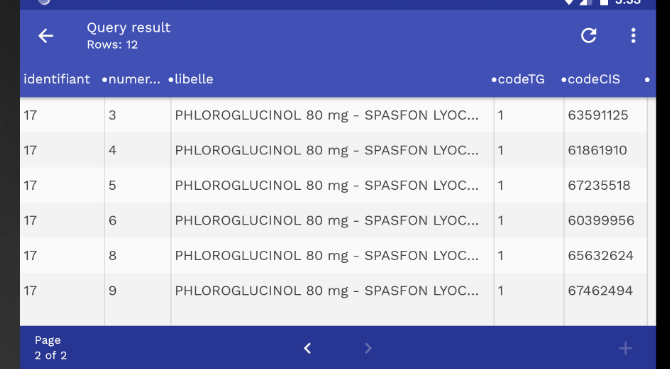
**c) Donner les codes CIS des autres médicaments faisant partie du même groupe générique.**

Requête :



>> Résultat :





**d)** **Vérifier que le médicament initial est bien le princeps.**

Dans le champ **codeTG** présenté dans les captures ci-dessus :

Lorsqu’il est égal à 0 c’est un princeps, s’il est à 1 c’est un générique.

Donc concernant le médicament initial avec comme code CIP = 3400931863014 et un code CIS = 62944693 *(encadré en rouge)* c’est un princeps.

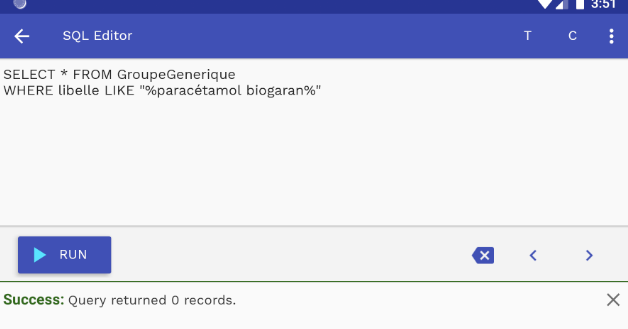
**e)** **Pourquoi la clé primaire de la table GroupeGenerique est-elle composée du couple (identifiant, numeroOrdre) ?**

Les clés primaires de la table GroupeGenerique sont « identifiant » et « numeroOrdre » car cette table permet de stocker l’ensemble des médicaments génériques d’un même groupe selon leur code et numéro d’ordre. Ce sont donc 2 informations essentiels.

**Compte rendu du 24/09/2020**

**18)** **Le site lepoint.fr indique que le paracétamol n’a pas de générique. Regarder si un groupe générique du médicament Paracétamol Biogaran existe.**

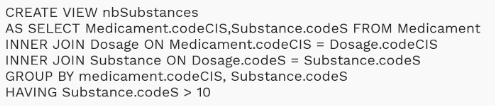
>> Requête + Résultat :



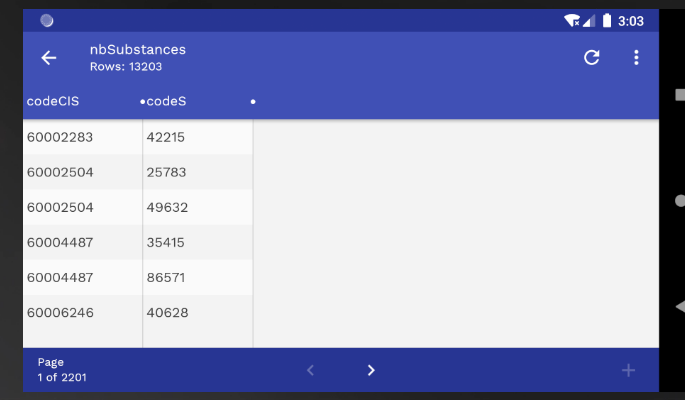
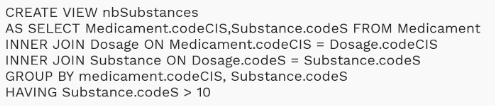
Selon les résultats, on a donc aucun générique du médicament Paracétamol Biogaran existant sur la table.

**20) et 21) On souhaite savoir quels sont les médicaments qui ont plus de 10 substances. Créer une vue qui permet de stocker le codeCIS d’un médicament et son nombre de substances (seulement ceux qui en ont plus de 10) + Afficher le résultat de la vue**

Requête :



>> Résultat :



**Problèmes :**

- Rechercher comment réaliser une vue **(résolu en fin d’heure)**

>> Problème de Jean-Charles :

L'application démarre, au moment d'ouvrir le dossier Files, l'émulateur freeze puis l'ordinateur affiche un écran noir m'obligeant à redémarrer l'ordinateur.

**(Toujours à la recherche de solution)**

>> Problème de Bianay :

La machine virtuelle plante souvent. De plus, les fichiers ne sont pas générés.

**(Toujours à la recherche de solution)**

**Solutions :**

Toutes les missions seront réalisées grâce à la machine virtuelle de Gouptar qui fonctionne très bien.

Donc l’exécution des missions seront collective tout en réglant les problèmes de chacun.

De plus, on a commencé la présentation pour le projet. (Début de diapo)

**>> A terminer**

**Compte rendu du 25/09/2020**

Recherche pour exécution des requêtes…

+ test sur SQL Editor

Durant le 28/09 et le 30/09

**Problème avec la machine virtuelle :**

**-** Aucune interaction et manipulation possible, la navigation était impossible sur la machine virtuelle… >> Bloqué

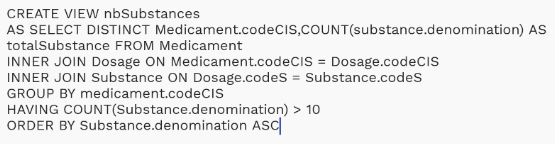
- La synchronisation du projet après mise à jour du logiciel Android Studio lançait un message d’erreur

**Problème résolu le 30/09**

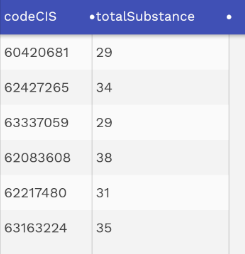
Réinitialisation de la synchronisation, et redémarrer la machine virtuelle à froid.

**Compte rendu du 01/10/2020**

**Correction de la requête 20**



>> Résultat :

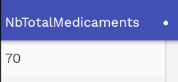


**22)** **Quel est le nombre de médicaments qui ont plus de 10 substances ?**

Requête :



>> Résultat :

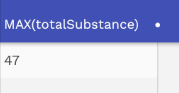


**25)** **Quel est le nombre de médicaments qui ont plus de 10 substances ?**

Requête :

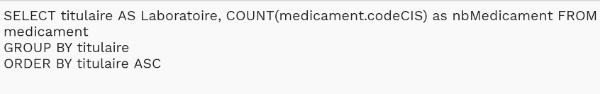


>> Résultat :

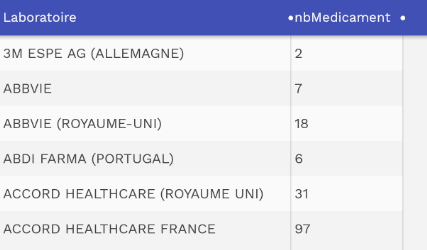


**29**) **Compter le nombre de médicaments par laboratoire.**

Requête :

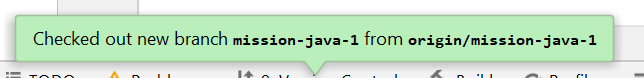


>> Résultat :



MISSION 3

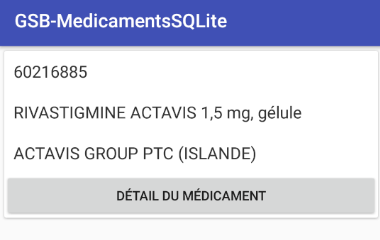
\*\* Récupération du dossier Java :



\*\* Nettoyage du code compilé dans les missions précédentes sur la machine virtuelle :



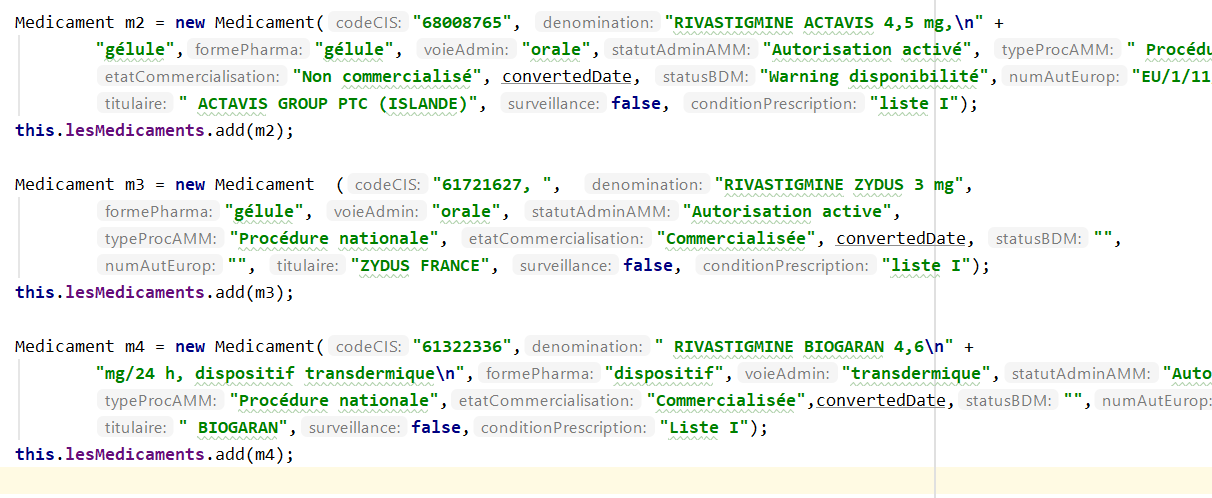
\*\* Installation de l’application :



**I. Créer des objets, appeler des constructeurs**

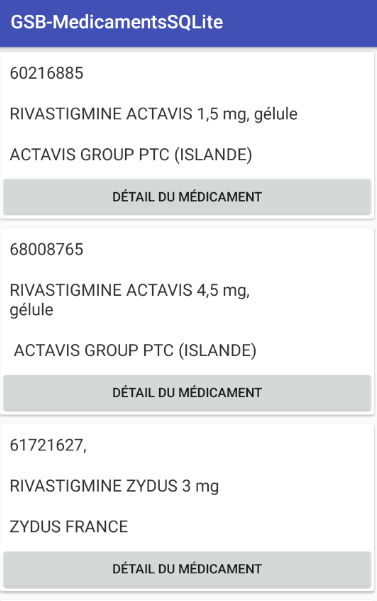
1)

\*\* Instanciation des médicaments :

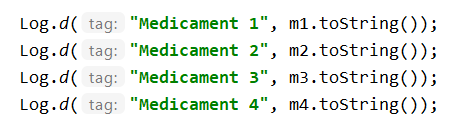


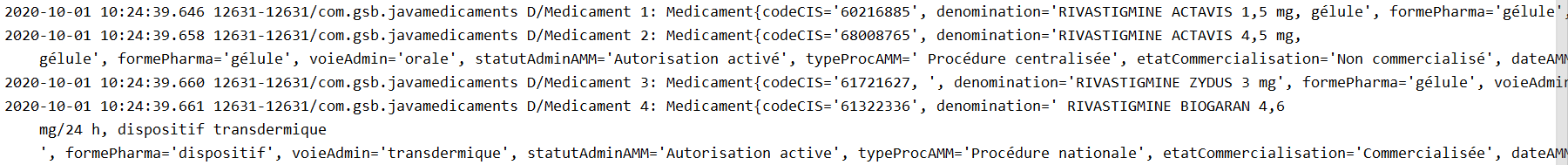
2)

Test :



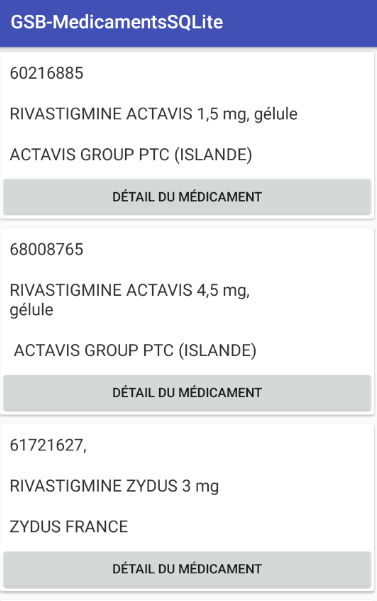
3)

Dans la méthode ***instancierLesObjets()*** : 

>> Résultat :

4)

Les données sont bien stockées dans la mémoire comme on peut le constater dans le ***Logcat***, cependant toutes les informations ne sont pas affichées



↑

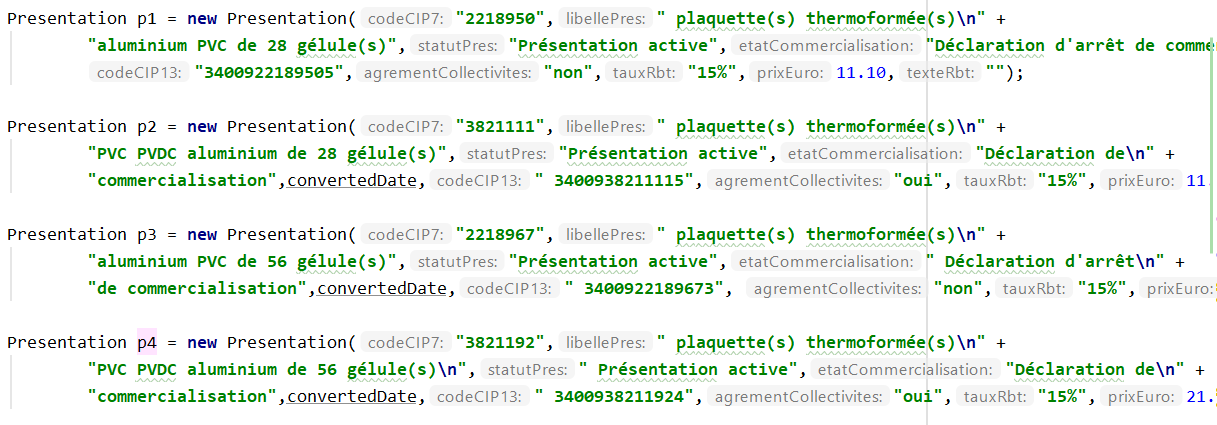
On a juste sur le smartphone :

- le codeCIS

- le nom du médicament et son titulaire (le laboratoire)

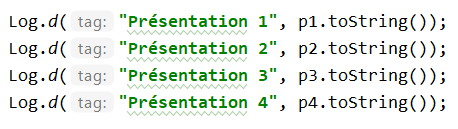
5)

\*\* Instanciation des présentations :

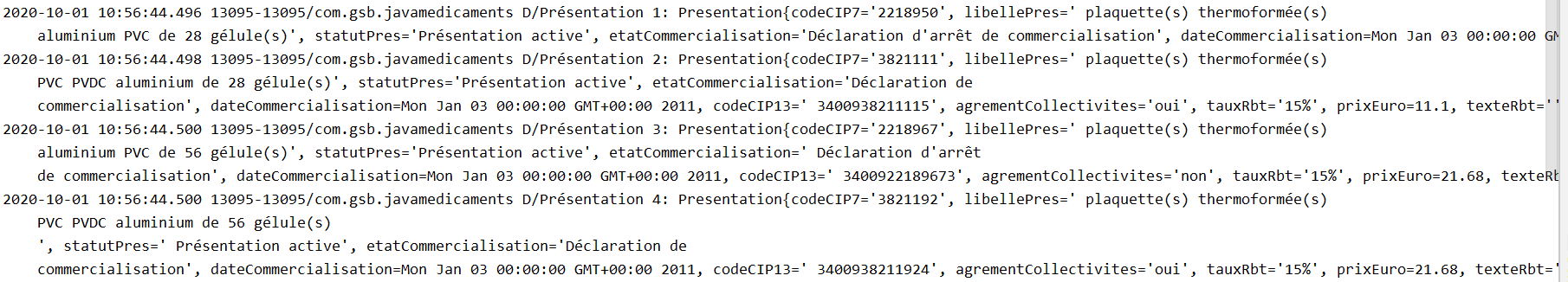


6)

Dans la méthode ***instancierLesObjets()*** :



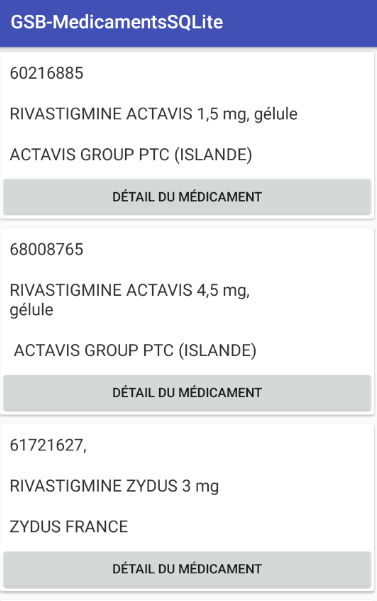
>> Résultat dans le Logcat



7)

En cliquant sur le bouton « Détail du médicament »

↓



Nous observons aucun résultat car le lien entre les objets de médicaments et présentation n’a pas encore été programmé.

**Compte rendu du 02/10/2020**

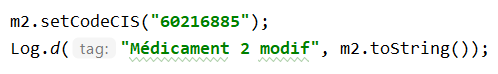
**II. Différencier une classe d’un objet**

8)

Modifier le codeCIS de **m2** avec la valeur du codeCIS de **m1**

Avec le code suivant :

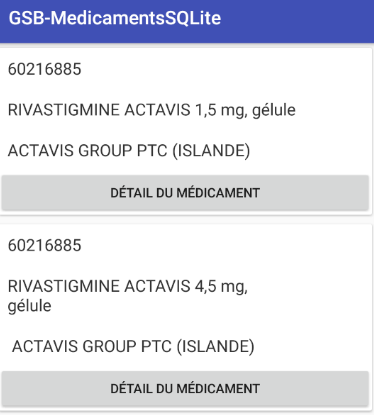
↓



>> Résultat dans le Logcat :



Sur le smartphone, on observe ceci :

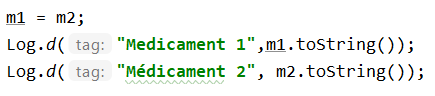


↑

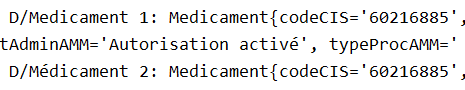
Le codeCIS encadré en rouge a bien été modifié pour le médicament 2

9)

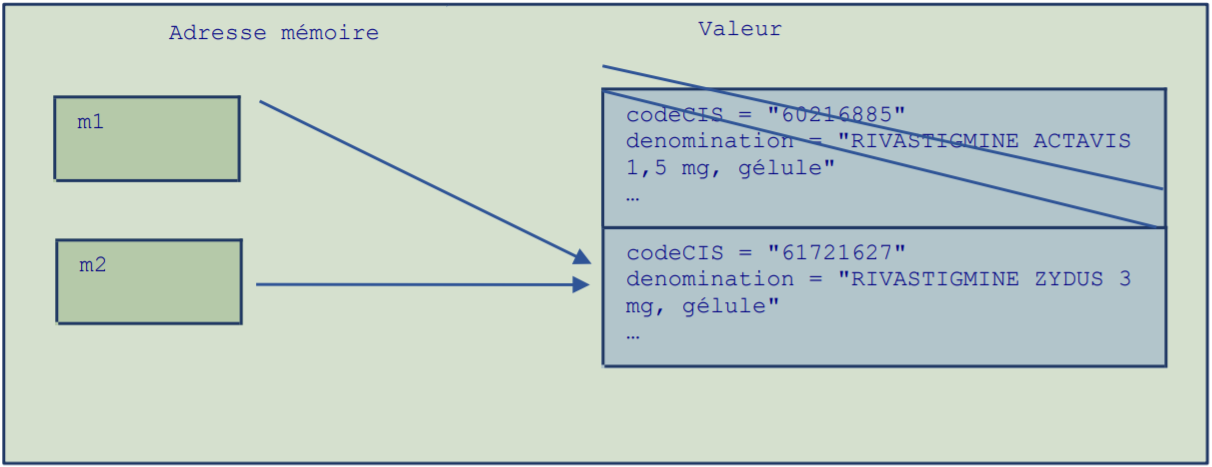
Affecter m2 dans m1



>> Résultat dans le Logcat :



Oui le résultat correspond au schéma ci-dessous car après avoir crée les deux objets avec l’instruction new, les deux adresses pointeront sur le même espace de valeur



10)

Mettre en commentaire la méthode toString() de la classe médicament



>> Résultat dans le Logcat :



L’identité des objets en Java ne correspond pas directement à une adresse mémoire mais à un hash code. Le **hash code** est un nombre entier sur 32 bits qui permet à la JVM de trier l’ensemble des objets et pour y accéder plus rapidement.

Ce n’est pas l’adresse mémoire qui est affichée mais le hash code de l’objet.

A l’exécution, la JVM trie les nouveaux objets en mémoire et utilise ce hash code pour y accéder plus rapidement.

A la différence de l’adresse mémoire, on considère que :

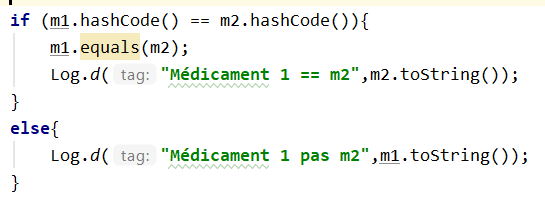
- deux objets n’ayant pas le même hash code sont forcément différents

- deux objets ayant le même hash code ne sont pas forcément égaux, il faut donc comparer leurs « vraies » adresses avec le == (ou la méthode equals()).

11)

Comparer les hash code des objets :

Test avec les médicaments m1 et m2



>> Résultat dans le Logcat



m1 et m2 ont le même hash code

On test avec m3 et m4

>> Résultat dans le Logcat :



On remarque que m3 et m4 n’ont pas le même hash code

12)

Pour afficher le hash code des présentation, il faut d’abord mettre en commentaire la méthode toString() de la classe présentation



>> Résultat dans le Logcat :

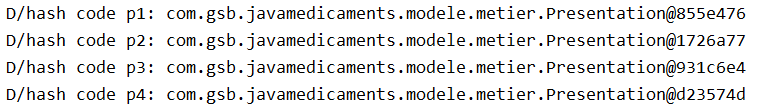
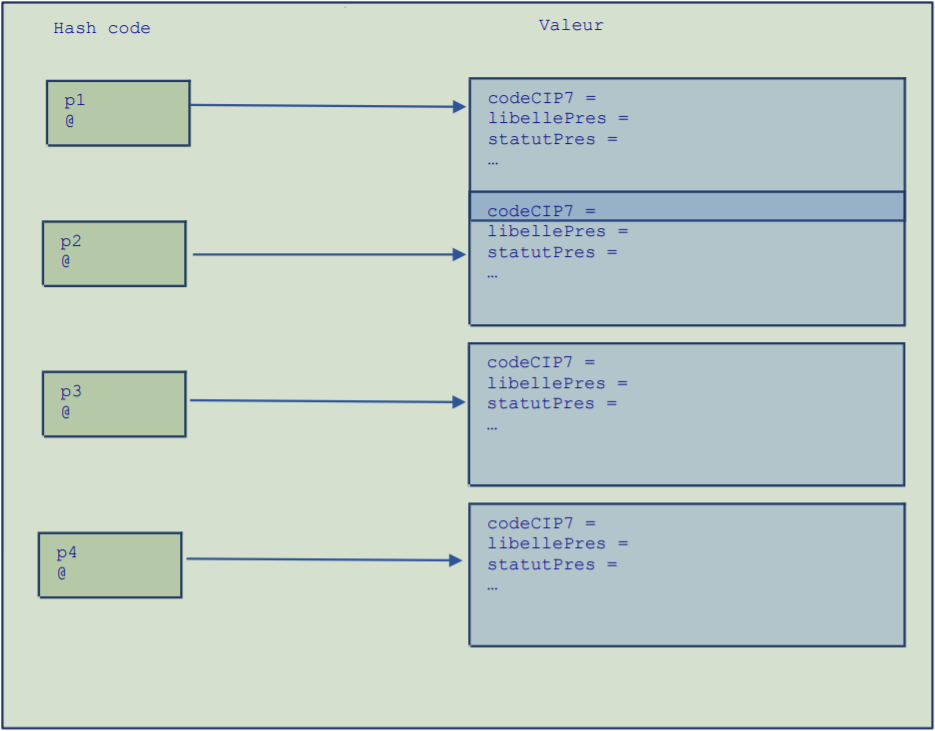


Schéma complété :



d2357d

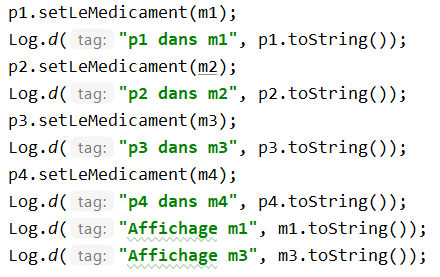
931c6e4

1726a77

855e476

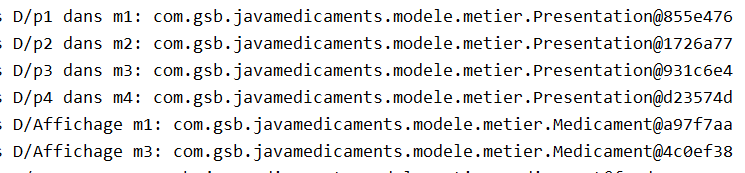
13)

Associer les présentations à chaque médicament :



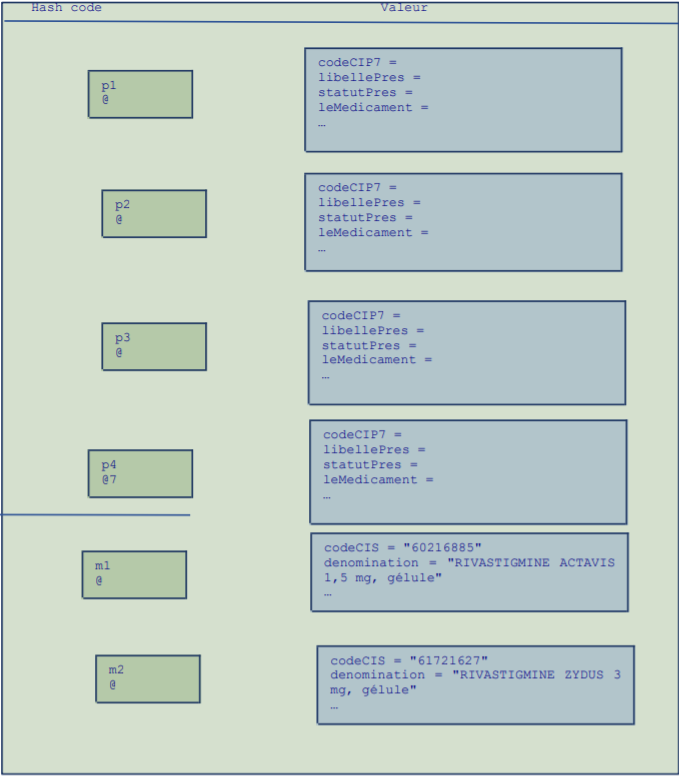


>> Résultat dans le Logcat :



14)

Schéma à compléter :



4c0ef38

a97f7aa

d23574d

931c6e4

1726a77

855e476

Le médicament m2 dans le schéma correspond en réalité au médicament m3 d’après la signature du constructeur

15)

**III. Manipuler des objets à l’aide de méthodes**

Principe de l’encapsulation.

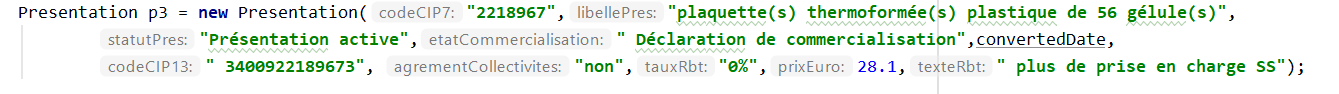
Pour récupérer le code CIS du médicament m1 créé avec un new, on utilisera la notation pointée :

String s = m1.getCodeCIS();

16)

Modifier la valeur des présentations crée (p1 et p3)





17)

Afficher les valeurs dans ***Logcat***

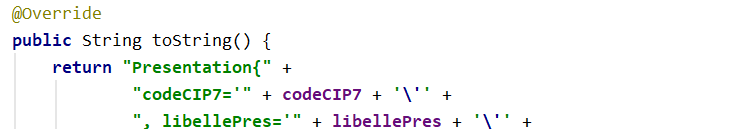


>> Résultat :



Vérification des modifications

- Enlever la méthode toString() de la classe Présentation :



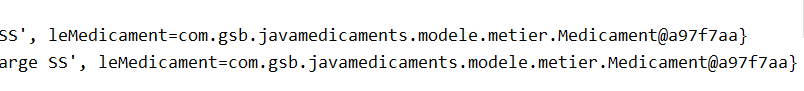
- Extrait dans Logcat :



Intégrer le médicament m1 dans la présentation p1 et p3 :

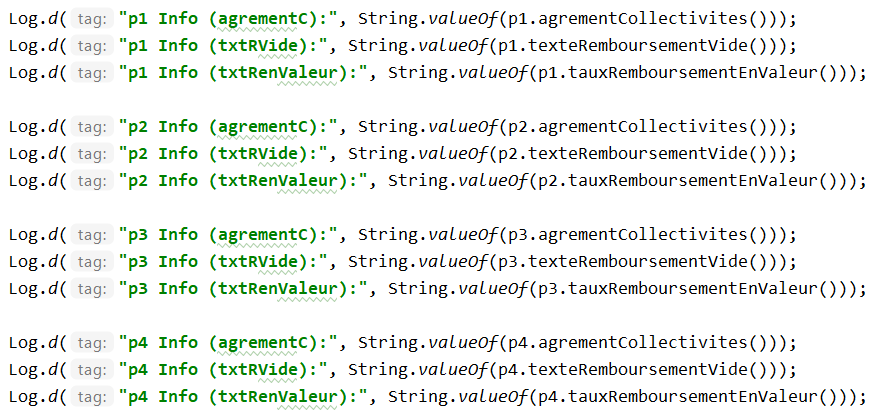


>> Extrait du ***Logcat*** :

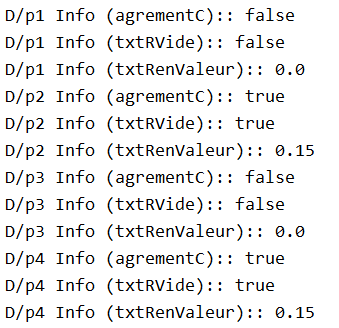


18)

Utilisation des méthodes pour les



>> Résultat dans le Logcat



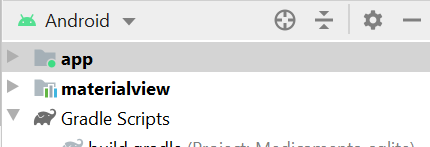
MISSION 4

**ETAPE 1 : Comprendre ce qui a été dit dans le daily scrum**

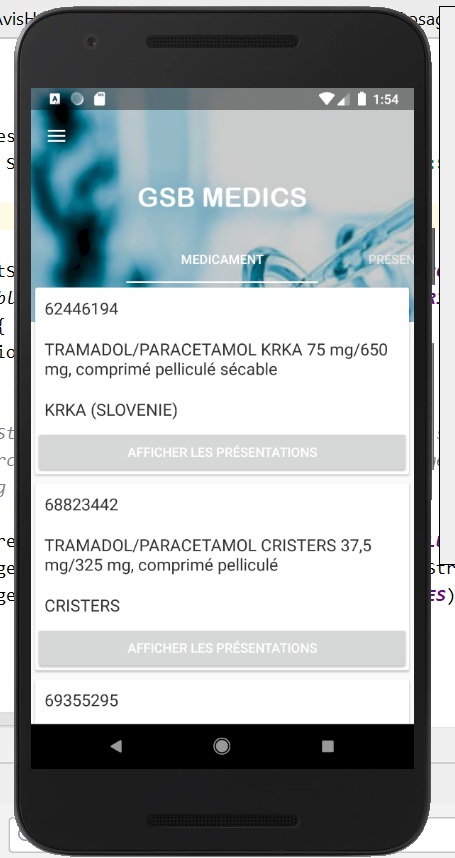
**1) & 2)**

On bascule sur la branche mission-fusionMView

Puis on a l’apparition d’une nouvelle application materielView

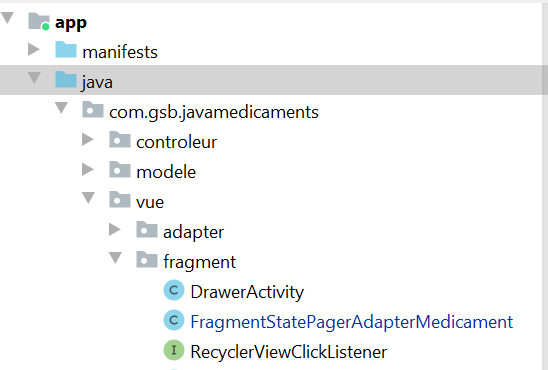


Puis quand on lance la machine virtuelle on a ceci :



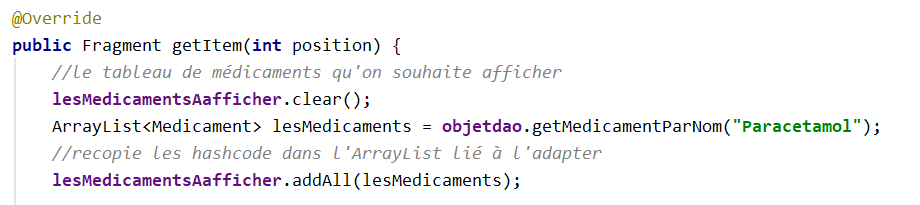
**4)**

Comprendre le fonctionnement des tableaux et des classes métiers :



Les tableaux sont récupérés dans la classe DAO.

Instanciation dans un objet nommé **objetdao** puis utiliser une méthode de la classe DAO :



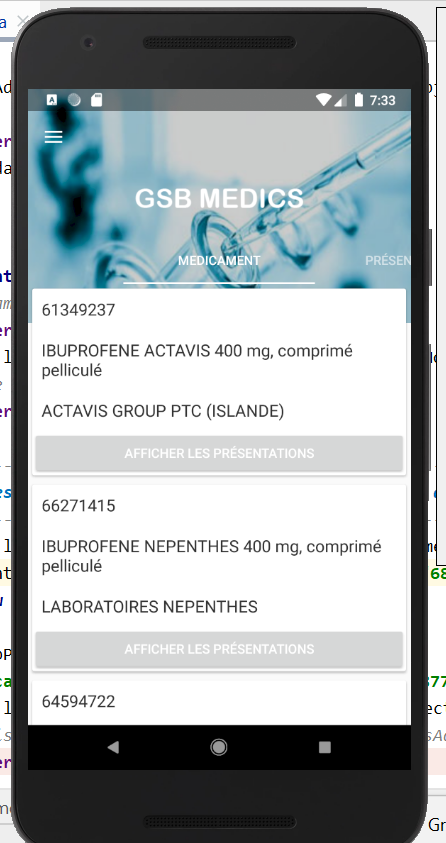
La méthode **getMedicamentParNom()** retourne un tableau dynamique (ArrayList) ou sont stockés les type *Medicament* qui sont « paracétamol » en paramètre.

**5) Rechercher les médicaments ayant « ibuprofene » dans leur nom.**

Code :



>> Résultat

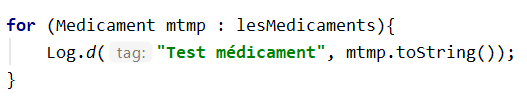


6)

Dans le tableau de médicament on va afficher les médicament dont le nom est Paracetamol :



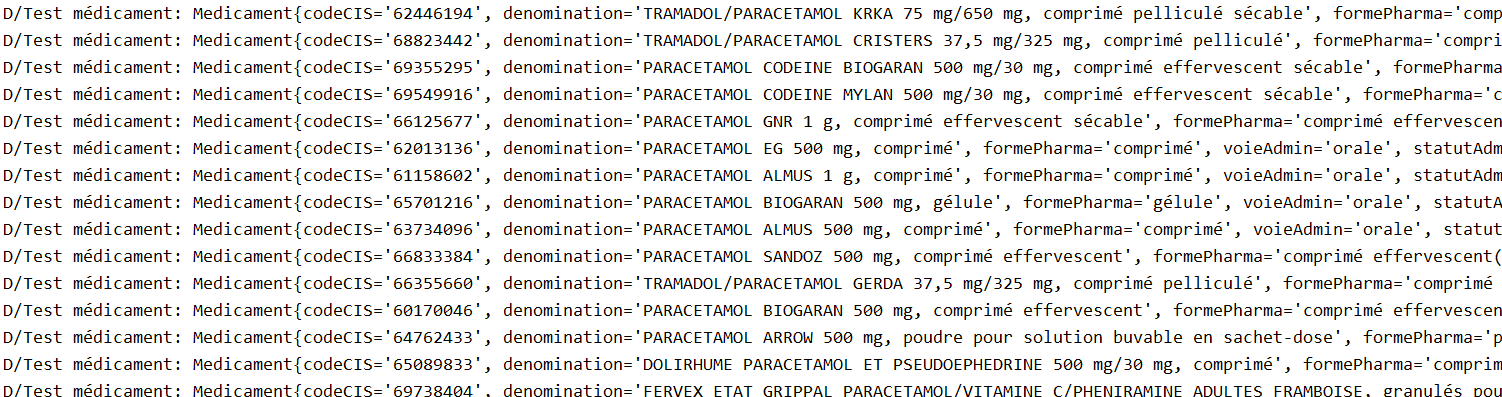
Puis écrire la fonction qui permet de récupérer le hash code des médicaments stocké dans l’ArrayList :



Afin d’afficher les détail d médicament

On décommente la méthode toString de la classe Présentation et on test :

>> Extrait dans le ***Logcat*** :



>> Résultat sur la machine virtuelle

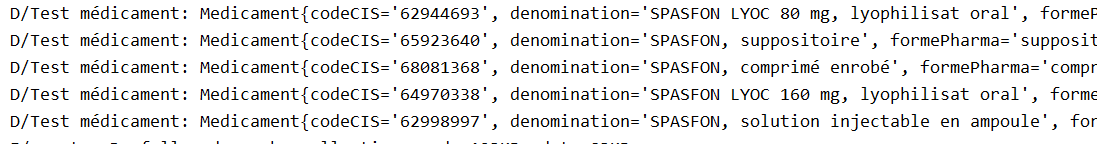


7)

Juste changer le nom du médicament :



>> Résultat dans le Logcat :

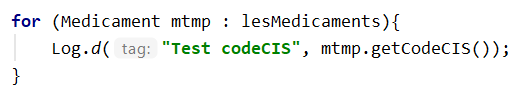


>> Résultat dans la machine virtuelle :

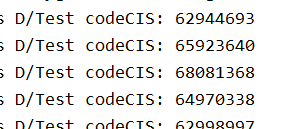


8)

La méthode pour afficher le codeCIS est la suivante :

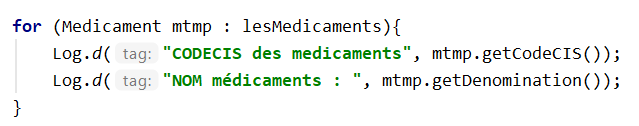


>> Résultat dans le ***Logcat*** :



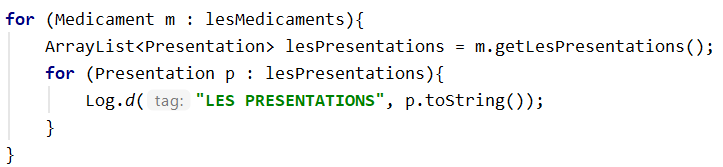
9)

Afficher le codeCIS et le nom des médicaments :



10)

Afficher dans le Log l’ensemble des présentations de tous les médicaments qui contiennent « Spasfon » dans leur nom avec le code suivant :



**ETAPE 2 : Travailler sur les ArrayList**

Une classe permettant de traiter les ArrayList nommé TraitementAL dans le package DAO.

11)

La fonction principale de cette méthode de retourne le nombre moyen de le médicament.

Tous les médicaments sont stocké dans une ArrayList permettant de récupérer leur noms

Puis on a une itération :

On a un nombre total initialisé à 0

Et le nombre de présentation par médicament initialisé à 0

Dans l’itération :

On récupère les médicaments dans une variable nommé m

On récupère les présentations pour chaque médicaments avec la méthode :

m.getLesPrésentation() ;

Puis dans cette liste de présentation

On fait une vérification

Le prix différent de 0 on cumule ce prix au total et on incrémente au fur a mesure le nombre de présentations

Ensuite après avoir restituer toutes les informations précédentes pour tous les médicaments en fonction de leur nom

Si le nombre de présentation égale à 0 donc le médicament n’a pas de présentation :

On retour 0 pour le nbPres

Sinon on effectue le calcul total du prix cumulé précédemment / nbpres

Pour avoir la moyenne des prix des médicaments.

12)

13)