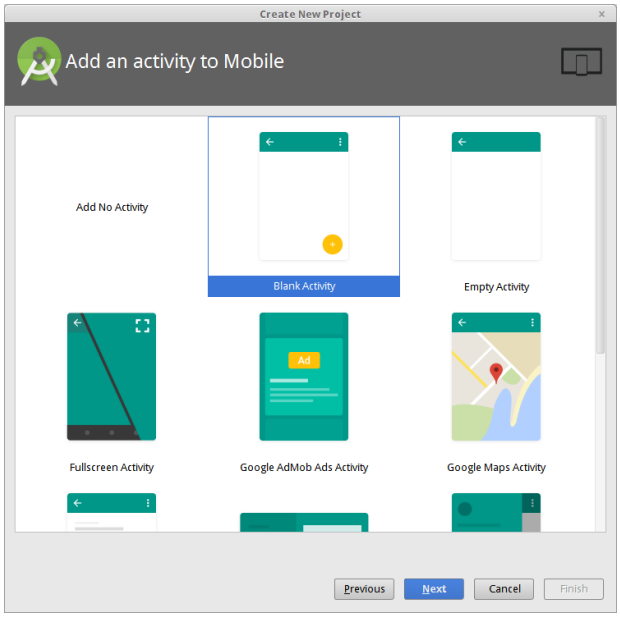
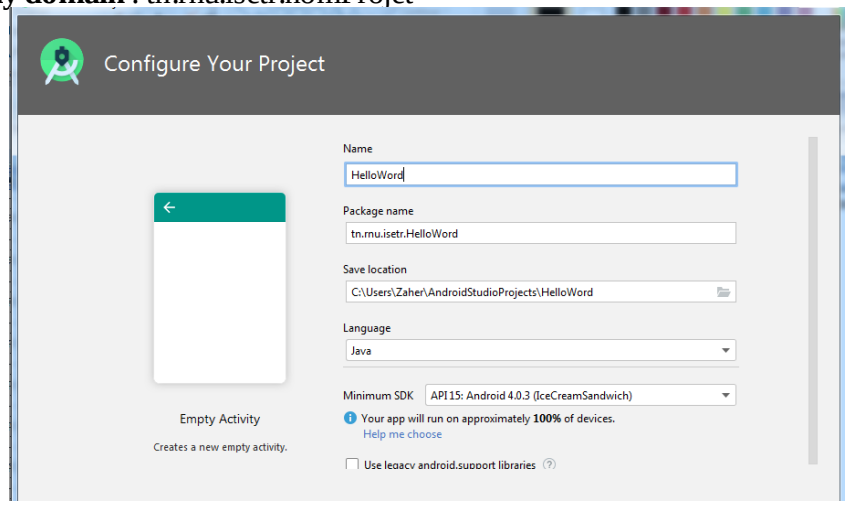
TP1 – Prise en main

1. Création d'un premier projet Android

Au tout début, vous n'avez aucun projet Android. Dans la fenêtre d'accueil, choisissez Start a new Android Studio project pour commencer à créer votre premier projet (Empty Activity).
Choisissez Empty Activity (NB: les modèles d'application dépendent de la version de Studio). Et cochez seulement Phone and Tablet.



La boîte suivante vous permet de choisir le nom de votre projet, son package et le dossier. Changez le nom et mettez **HelloWorld**. Ne cochez pas C++ et Kotlin support (autres langages que Java pour développer des projets).
Changez l'emplacement du projet, *Project **Location*** selon l'emplacement désiré de votre disque.
Changer le **company domain** : tn.rnu.isetr.nomProjet



Choisissez l'API 15 (IceCreamSandwitch). Cela définit le niveau de fonctionnalités de votre application : s'il est trop élevé, peu de smartphones seront capables de faire tourner votre application ; trop bas, vous n'aurez pas beaucoup de choses agréables à votre disposition.

Application name : c'est le nom qui va apparaitre dans la liste des applications sur l'appareil et dans le Play Store.

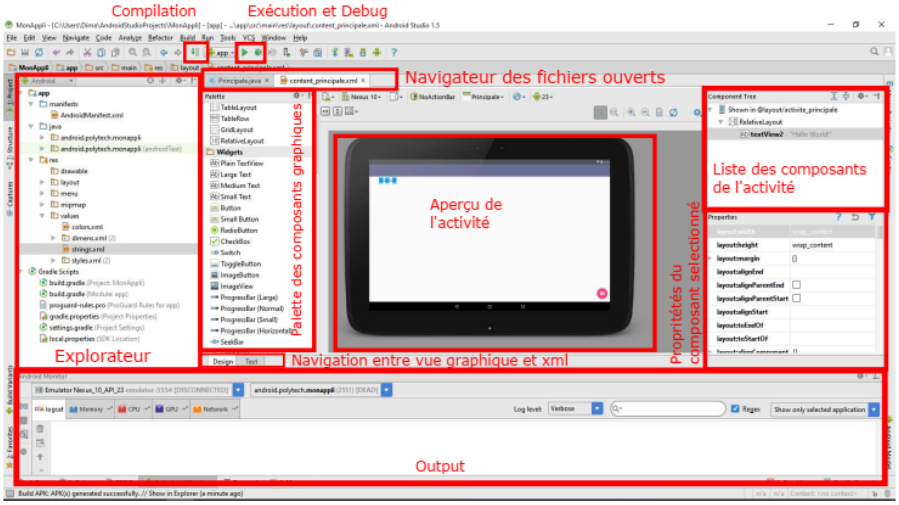
Company domain : c'est un qualifiant qui apparaitra dans le nom du package .

Package name : il est utilisé comme identifiant de l'application, il permet de considérer différentes versions d'une application comme étant une même application. Il doit être unique parmi tous les packages installés sur le système.

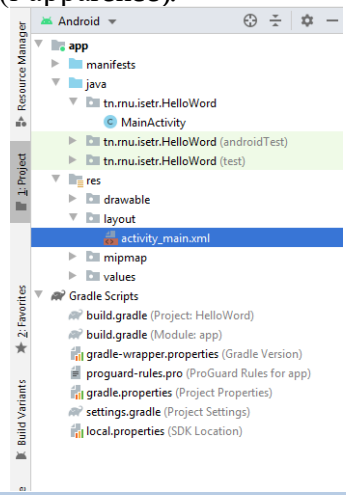
Minimum required SDK : c'est la version Android la plus ancienne sur laquelle l'application peut tourner. Il faut éviter de remonter trop en arrière, ça réduirait les fonctionnalités que vous pourriez donner à votre application.

Découverte de la fenêtre de travail

La figure suivante montre les principaux éléments de l'interface Android Studio. Tout projet Android doit respecter une hiérarchie bien précise qui permettra au compilateur de retrouver les différents éléments et ressources lors de la génération de l'application. Cette hiérarchie favorise la modularité des applications Android.



A la création du projet, Android Studio crée automatiquement des dossiers pour contenir les fichiers de code Java, les fichiers XML, et les fichiers multimédias.
L'explorateur de projet vous permettra de naviguer dans ces dossiers.
Les dossiers que nous utiliserons le plus sont ***java*** et ***res***. Le premier contient le code Java qui définit le comportement de l'application et le second comporte des sous dossiers où sont stockés les ressources qui définissent l'interface de l'application (l'apparence).



Tout ce qui touche à l'interface utilisateur sera intégré dans les sous dossiers de ***res***, dont voici une brève description :

- **layout** regroupe les fichiers XML qui définissent la disposition des composants sur l'écran.

- **drawable-****** contient tout élément qui peut être dessiné sur l'écran : images (en PNG de préférence), formes, animations, transitions, etc..
- **mipmap** contient les images de l'icône de votre application sous différentes résolutions.
- **values** contient les fichiers XML qui définissent des valeurs constantes (des chaines de caractères, des dimensions, des couleurs, des styles etc.)

gradle Android Studio utilise un système qu'on appelle *Gradle* pour compiler et générer les applications. Pour fonctionner le Gradle a besoin d'un script qui définit les règles de compilation et génération (configuration et dépendances). Android Studio crée ainsi un script gradle pour chaque module (*build.gradle (Module :app)*) du projet ainsi qu'un script pour le projet entier (*build.gradle (Project : MonAppli)*).
Dans le *build.gradle* de l'application on définit entre autre la version du SDK utilisée pour la compilation, la version minimale du SDK nécessaire pour faire tourner l'application (rétro-compatibilité), l'identifiant de l'application (le nom du package), etc.[4]
Vous trouverez également dans le dossier ***manifests*** du projet un fichier nommé *AndroidManifest.xml*. Ce fichier est obligatoire dans tout projet Android, et doit toujours avoir ce même nom. Ce fichier permet au système de reconnaitre l'application.

Prenez quelques instants pour déplier les dossiers manifests, javaet reset regarder ce qu'il y a dedans (ça a été décrit en haut).

2. Découverte du SDK Android

On va regarder les outils de développement : le *Software Development Toolkit* d'Android.

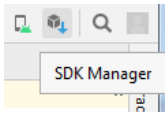
2.1. Software Development toolKit :SDK

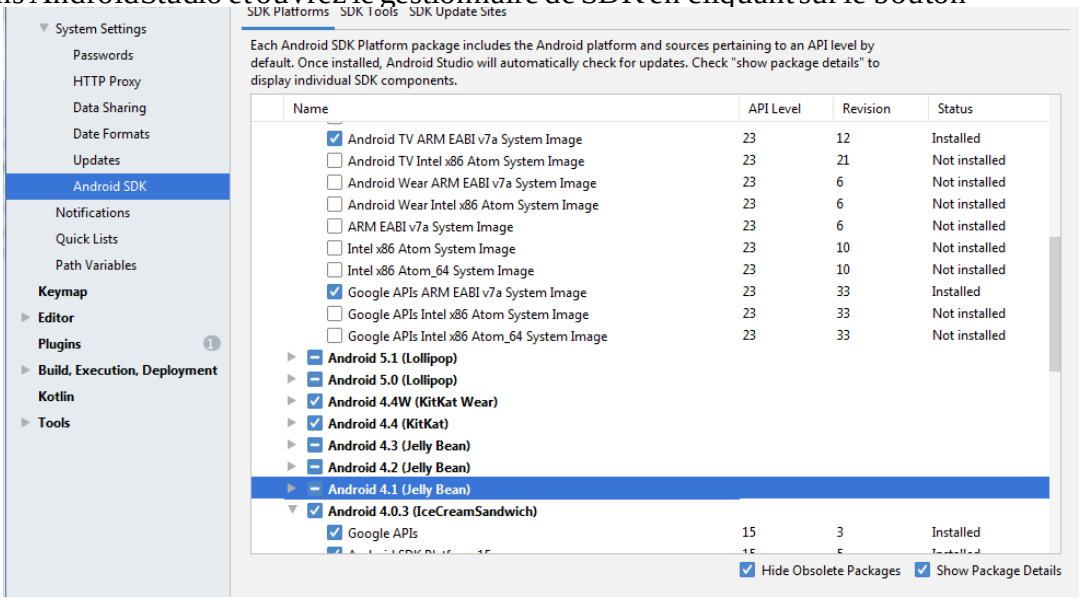
Le SDK Android a été téléchargé dans le dossier D:\android\sdk. Allez voir ce qu'il y a dedans : plusieurs dossiers, dont :

- tools qui contient les outils de gestion du SDK et des AVD (emulator.exe),
- platform-tools qui contient quelques programmes utiles comme adb.exe, system-images qui contient les images de la mémoire de tablettes virtuelles, par exemple API 17 et 25.

Gestionnaire de SDK

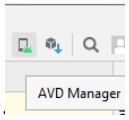
Revenez dans AndroidStudio et ouvrez le gestionnaire de SDK en cliquant sur le bouton



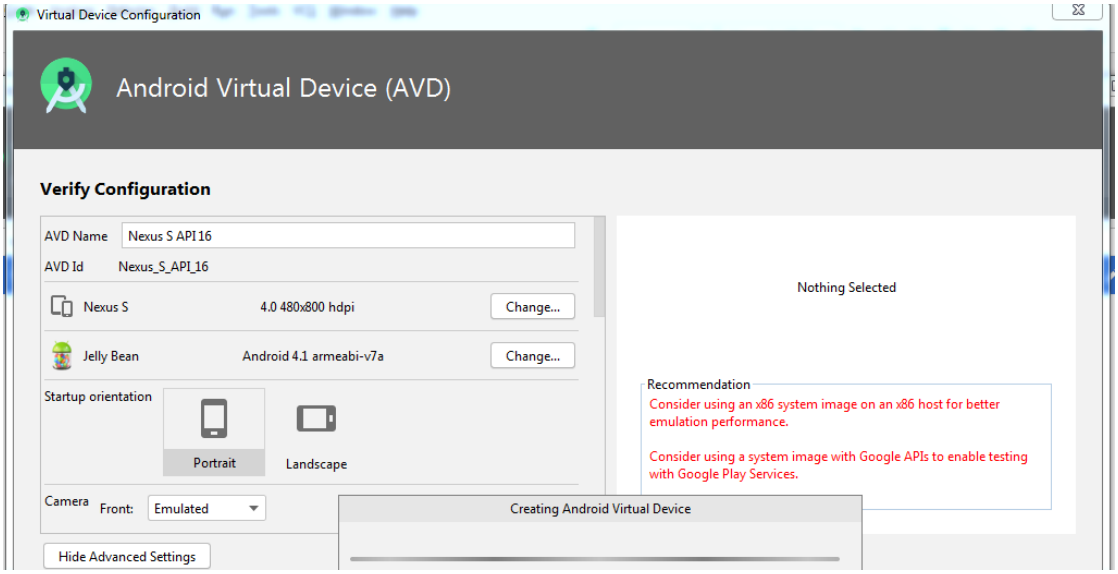


L'onglet SDK Tools montre les outils installés, par exemple Android Emulator, Android SDK Tools.

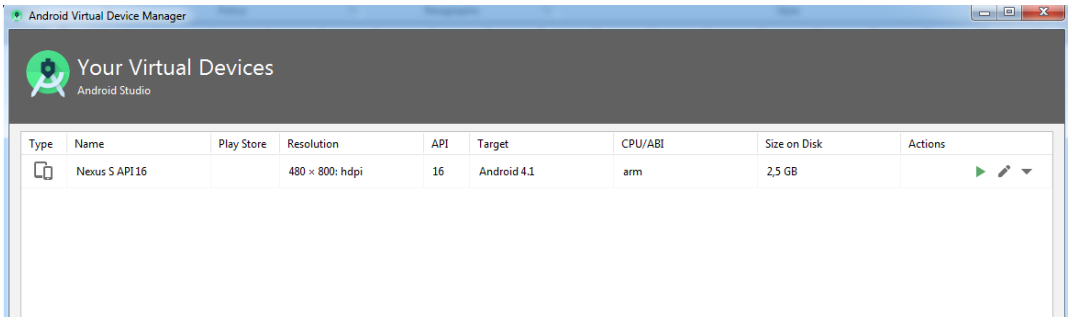
Android Virtual Device :AVD



On revient encore dans AndroidStudio, pour lancer le gestionnaire **d'AVD**.
Commencez à créer un smartphone virtuel, avec le bouton + Create Virtual Device. Choisissez le type Phone, modèle Nexus S ; puis dans l'écran suivant, changez d'onglet pour x86 images et choisissez API 16 (celle qui est installée), Validez la création de ce smartphone virtuel.



La création d'un AVD prend parfois du temps. Il faut générer l'image de la mémoire virtuelle. Une fois que c'est fini, l'AVD apparaît dans la liste et vous pouvez le lancer en cliquant sur le triangle vert à droite.



Notez qu'elle met beaucoup de temps à devenir disponible parce que c'est un émulateur de machine, comme VMware ou VirtualBox. Pour ne pas perdre de temps, vous devrez faire attention à ne pas la fermer.

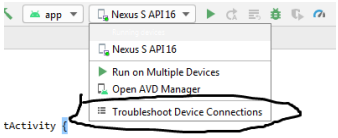
Modifiez les préférences de cette tablette : ouvrez l'application Settings.

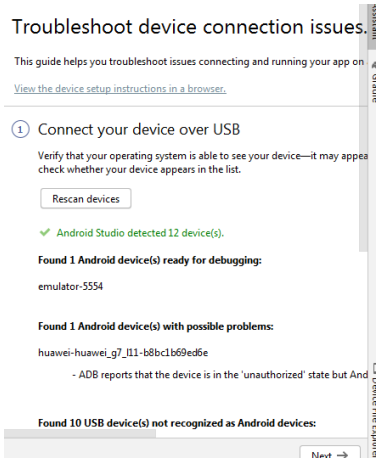
- Mettez-la en langue française : Language & Input, Language, ajoutez Français (France) et mettez-le en premier dans la liste (c'est selon la variante d'Android que vous avez choisie),
- Enlevez le verrouillage de l'écran en cas d'inaction : Sécurité, Verrouillage de l'écran, Aucun.

NB : faites attention à ne pas mettre cet AVD en veille (avec le bouton *Power*), car cela ferait créer une image de la RAM de 1024 Mo dans votre mémoire.

2.2. Android Debug Bridge :ADB

Avant de passer à la programmation, **adb devices** Affiche la liste des smartphones connectés : les réels par USB et les virtuels actifs.

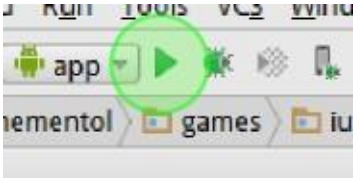




3. Programmation

3.1. Exécution

Le lancement de l'application est simple. Il suffit de cliquer sur le bouton triangulaire vert dans la barre d'outils (celui qui est entouré en vert ci-dessous).



Ensuite, s'il n'y a aucune tablette connectée ni AVD en cours, Studio vous demande quel AVD il faut lancer. Il en faut un qui corresponde au niveau d'API de votre application. Cochez Use same device for future launches.

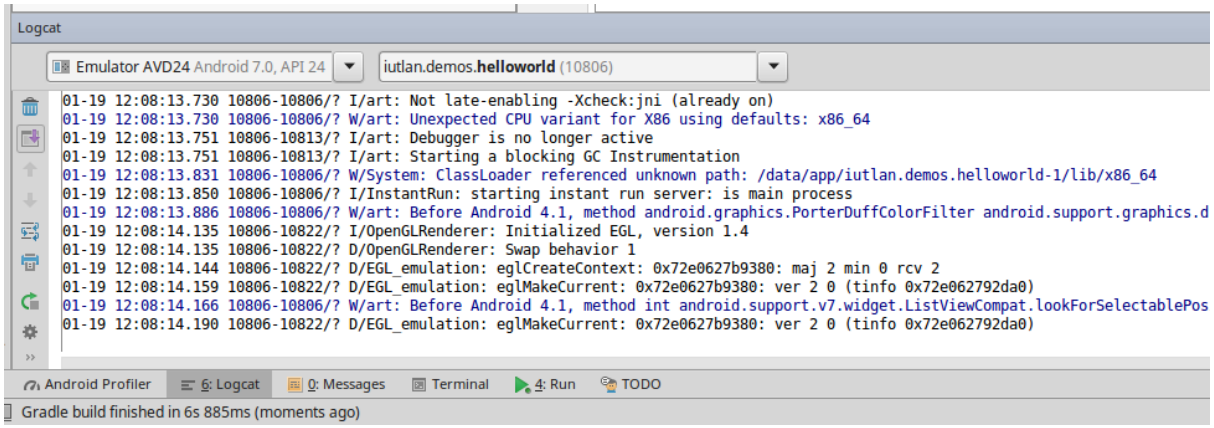
3.2. Affichage d'un message

Ouvrez le source **MainActivity.java** et trouvez la méthode onCreate. Cela doit ressembler au source suivant. Complétez-le avec les instructions qui permettent de faire afficher un message dans la fenêtre LogCat.

```
import android.util.Log;
...
public class MainActivity extends Activity {
public static final String TAG = "hello";
@Override
protected void onCreate(...) {
super.onCreate(...);
setContentView(R.layout.activity_main);
Log.i(TAG, "Salut !");
}
```

Ouvrir la fenêtre Android Monitor, onglet LogCat.

NB: l'interface est légèrement différente dans les nouvelles versions de Studio.



Cette fenêtre permet de voir les **messages** émis par la tablette. Le bouton en forme de poubelle vide ces messages. Vous pouvez changer le niveau d'affichage : verbose, debug, info. . . ou créer un filtre basé sur le *TAG* pour ne voir que les messages utiles.
Lancez l'exécution du projet. Vous devriez voir une ligne similaire à celle-ci

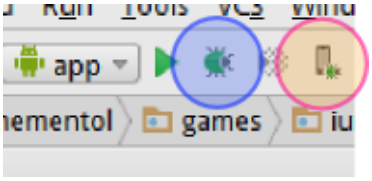
01-24 09:58:38.310 992-992/fr.iutlan.myapplication I/hello: Salut !

3.3. Mise au point

Nous allons expérimenter quelques aspects du débogueur intégré dans Studio. On va être un peu limités par la petitesse du programme. On va seulement pouvoir découvrir quelques techniques très utiles.
A chaque fois que vous modifiez un source Java, le débogueur vous signale qu'il doit se déconnecter de la tablette. En effet, si vous changez le source, ce n'est plus ce qui est exécuté sur la tablette, donc la mise au point ne peut pas continuer.

Lancement en mode debug

Normalement, il suffit de cliquer sur le bouton bleu pour lancer la mise au point.



Mais s'il y a une erreur de connexion (message Error running app: Unable to open debugger port dans la barre d'état tout en bas), il faut d'abord «attacher» Studio à l'AVD. Il faut cliquer sur le bouton Attach debugger to Android process entouré en fuschia ci-dessous. Vous serez peut-être obligé(e) de relancer l'AVD.

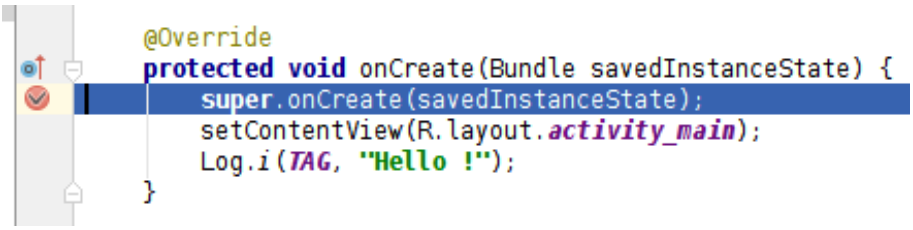
Voici les boutons utiles pour mettre le programme au point. Le bouton entouré en **vert** permet de faire repartir le programme jusqu'au prochain point d'arrêt. Le bouton **cyan** permet d'exécuter la prochaine ligne sans rentrer dans les méthodes imbriquées s'il y en a. Le bouton **orange** entre dans les appels imbriqués de méthodes. Le bouton **violet** exécute à pleine vitesse jusqu'à la sortie de la fonction puis s'arrête. Le bouton entouré en **bleu foncé** affiche les points d'arrêt. Nous n'emploierons pas les autres boutons.

La mise au point repose sur ces deux techniques :

- Lancer en mode normal et attendre que ça se plante. L'exécution s'arrête alors exactement sur l'instruction qui est mauvaise, par exemple celle qui déclenche une NullPointerException.
- Mettre un *point d'arrêt* avant l'endroit qu'on pense mauvais puis exécuter ligne par ligne pour voir ce qu'il y a dans les variables et guetter l'erreur.

Placement d'un point d'arrêt

Cliquez dans la marge gauche au début de la méthode onCreate(). Ça place un gros point rouge dans la marge.



La fenêtre Breakpoints, qu'on obtient en cliquant sur le bouton entouré en bleu foncé dans la figure précédente, liste tous les points d'arrêts que vous avez placé. N'en mettez pas trop dans un programme, vous aurez du mal à suivre tout ce qui peut se passer. Mettez vos méthodes au point les unes après les autres, pas tout en même temps.

Vidéo Explicative : Tutoriel Android Studio 2 : Déboguer et lire les logs | video2brain.com**:** **https://youtu.be/fY5qi9IbCgI**

Travail à tester : Exécution pas à pas

Rajoutez ceci à la fin de la méthode onCreate() de MainActivity :

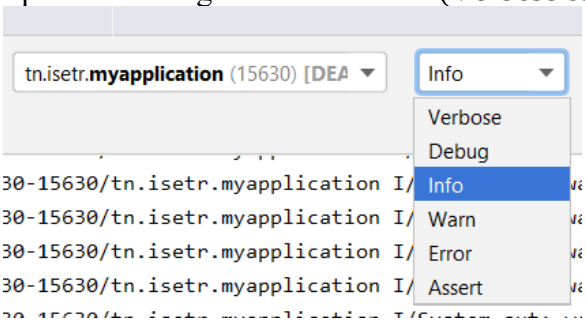
```
int v = 54;
int n = v / 9 + 3;
int f = factorielle(n);
Log.i(TAG, n+"! = "+f);
```

avec ceci un peu plus loin :

```
private int factorielle(int n)
{
int r;
if (n > 1) {
int fnm1 = factorielle(n-1);
r = n * fnm1;
} else {
r = 1;
```

```
}
return r;
}
```

Réglez LogCat pour n'afficher que des messages info au moins. (Verbose est la priorité la plus basse)



Mettez un point d'arrêt sur la première instruction, int v = 54; puis lancez en mode mise au point.
Puis essayez les modes suivants :

- Ligne par ligne : appuyer sur F8 (step over) pour exécuter la prochaine ligne. Suivez les modifications des variables dans la fenêtre Variables. Voyez que l'appel à factorielle a été fait à pleine vitesse.
- Instruction par instruction avec la touche F7 (step into) : elle rentre dans les appels aux méthodes. Alors pour factorielle, ça va un moment, mais c'est pénible. Vous pouvez couper court avec MAJ+F8 (step out).

Supprimez l'exécution en cours à l'aide du bouton Stop app (carré rouge dans la barre d'outils verticale à gauche).

3.4. Mise en place d'un bug

Normalement, c'est l'inverse qu'on fait. Les bugs, on les dégomme (renvoi). Là, on va créer volontairement un bug et essayer de voir si on peut le retrouver.
Ouvrez le fichier **app/res/layout/activity_main.xml**. Ça affiche une vue simulée de l'interface utilisateur. Cliquez sur le bouton text juste en dessous de cette vue. Ça affiche le source XML qui définit l'interface. Modifiez l'élément <TextView comme suit ;

```
<TextView
    android:id="@+id/texte"
    ...  />
```

Puis rajoutez ceci à la fin de la méthode onCreate de MainActivity, après le setContentView:

```
TextView texte = findViewById(R.mipmap.ic_launcher);
texte.setText("Ça marche");
```

Tapez sur ALT-entrée sur les erreurs signalées, par exemple un import manquant, pour les corriger.
Le problème qu'on a rajouté exprès consiste à se tromper d'identifiant de ressource. **R.mipmap.ic_launcher** est le numéro d'un icône, pas d'un TextView dans l'interface, alors que va-t-il se passer ? C'est une erreur assez fréquente. Il est important de savoir la réparer.

NB: AndroidStudio est capable de détecter ces erreurs, mais pas d'empêcher l'exécution. Vous devez :

1. Configurer LogCat pour ne voir que les messages d'erreur, vider LogCat avant chaque tentative.
2. Enlever les points d'arrêts.

3. Lancer le programme en mode normal (pas en mise au point), attendre de voir le popup « L'application HelloWorld s'est arrêtée ».
   Qu'est-ce qui est écrit dans LogCat, en rouge ? Ça vous dit où ça s'est planté, mais c'est dans une fonction du système ActivityThread.performLaunchActivity. Il faut chercher l'autre explication plus bas : Caused by: **java.lang.NullPointerException** et juste en dessous, la ligne problématique *nom de fichier:n° de ligne* dans votre programme. Vous pouvez double-cliquer sur cette ligne de LogCat pour aller dans la source.

4. En fait, en mode mise au point, ça ne s'arrête pas sur la ligne problématique, donc il faut poser un point d'arrêt sur le début de la méthode onCreate puis tracer ligne par ligne jusqu'à celle qui pose un problème.

Voici maintenant un autre bug, aussi fréquent :

```
Button bouton = (Button) findViewById(R.id.texte);
bouton.setText("Ça marche");
```

Cette fois, **R.id.texte** est un identifiant, mais ce n'est pas celui d'un Button, il va y avoir une erreur de conversion de classe : ClassCastException. Studio le signale, mais il laisse faire. Là également, le mode debug n'arrête pas l'exécution exactement là où elle se produit.

TextView est une superclasse de Button mais on n'a pas le droit de convertir de la superclasse vers une sous-classe. Seul l'inverse est possible.

4. Travail à rendre

Avec le navigateur de fichiers, allez dans le dossier de votre projet AndroidStudio. Pour le trouver, regardez le titre de la fenêtre tout en haut : C:\...\AndroidStudioProjects\... Ouvrez ce dossier avec Windows, puis descendez successivement dans app puis src. Vous devez y voir le dossier main. Cliquez droit sur le dossier main et puis Compresser.
Vous devrez déposer le fichier main zippé dans le dépôt Moodle Android à la fin du TP.