

TD1 - Prise en main

Pré-requis : Android Studio installé sur votre PC/MAC

Si cela n'est pas déjà fait veuillez installer via le lien suivant la dernière version d'Android Studio : https://developer.android.com/studio/?gclid=Cj0KCQjwnqH7BRDdARIsACTSAdsfnAgcAgp2aqkY1i1JmWyoQni63QEg0NVOiLXHapP7klbvQ3aYlrgaAljQEALw_wcB&gclidsrc=aw.ds

Attention, l'outil télécharge beaucoup de dépendances, une bonne connexion internet est requise.

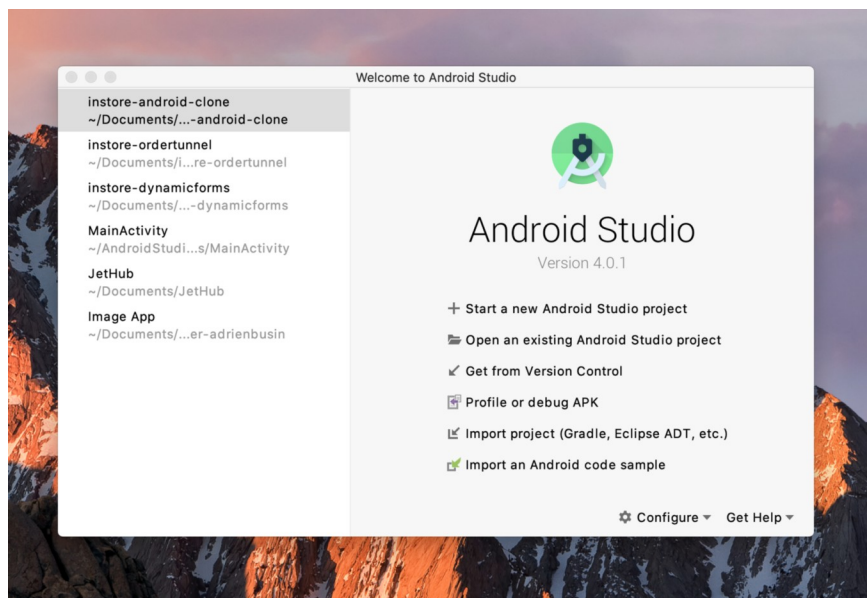
Voici la configuration requise minimale pour pouvoir faire tourner Android Studio. Si votre ordinateur n'est pas assez puissant, je vous conseille d'utiliser les PC de l'école.

System requirements

Windows	Mac	Linux	Chrome OS
<ul style="list-style-type: none">• Microsoft® Windows® 7/8/10 (64-bit)• 4 GB RAM minimum, 8 GB RAM recommended• 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)• 1280 x 800 minimum screen resolution	<ul style="list-style-type: none">• Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)• 4 GB RAM minimum, 8 GB RAM recommended• 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)• 1280 x 800 minimum screen resolution	<ul style="list-style-type: none">• GNOME or KDE desktop <i>Tested on gLinux based on Debian.</i>• 64-bit distribution capable of running 32-bit applications• GNU C Library (glibc) 2.19 or later• 4 GB RAM minimum, 8 GB RAM recommended• 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)• 1280 x 800 minimum screen resolution	<ul style="list-style-type: none">• 8 GB RAM or more recommended• 4 GB of available disk space minimum• 1280 x 800 minimum screen resolution• Intel i5 or higher (U series or higher) recommended <p>For more information on recommended devices as well as Android emulator support, visit chromeos.dev</p>

1) Découverte d'Android Studio

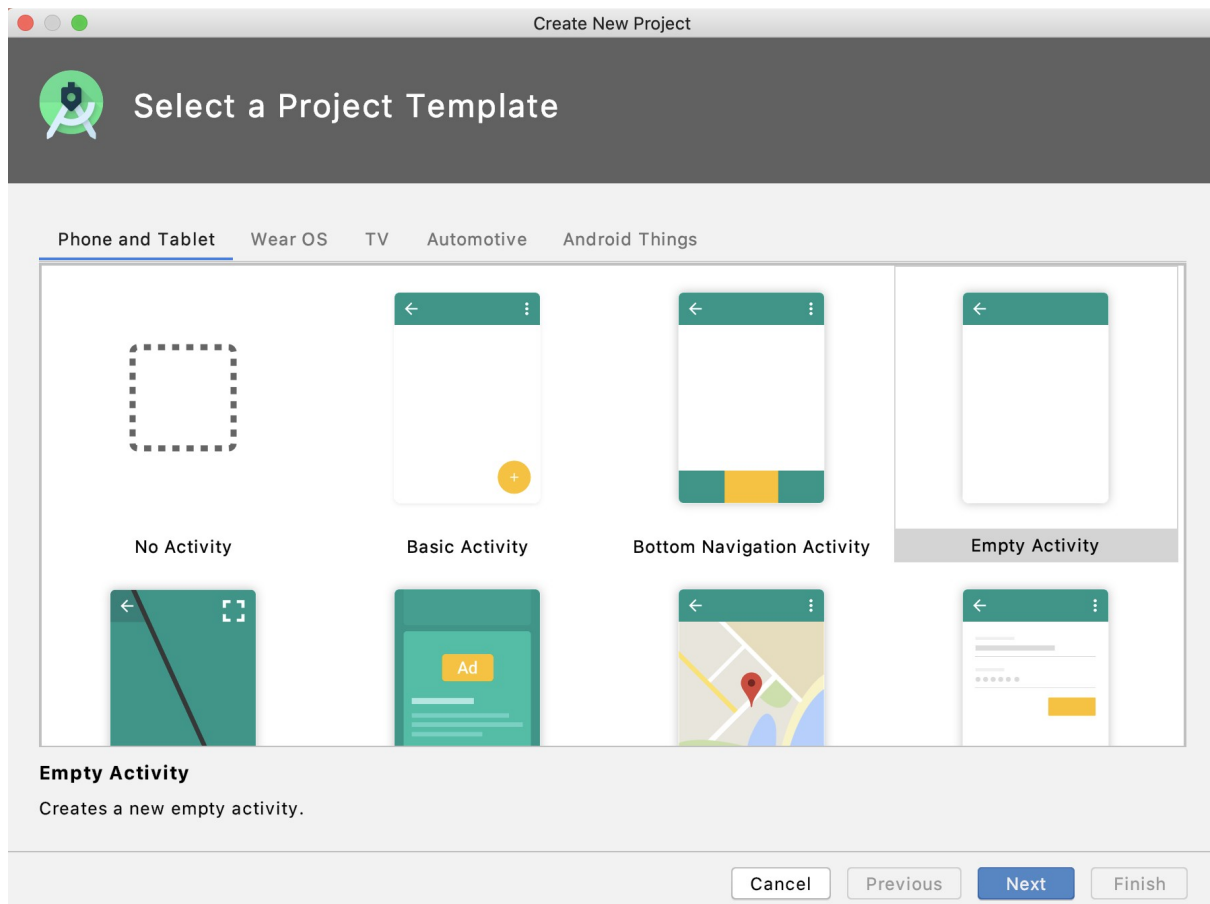
Lancer Android Studio sur votre PC/MAC, la fenêtre principale d'Android Studio apparaît.



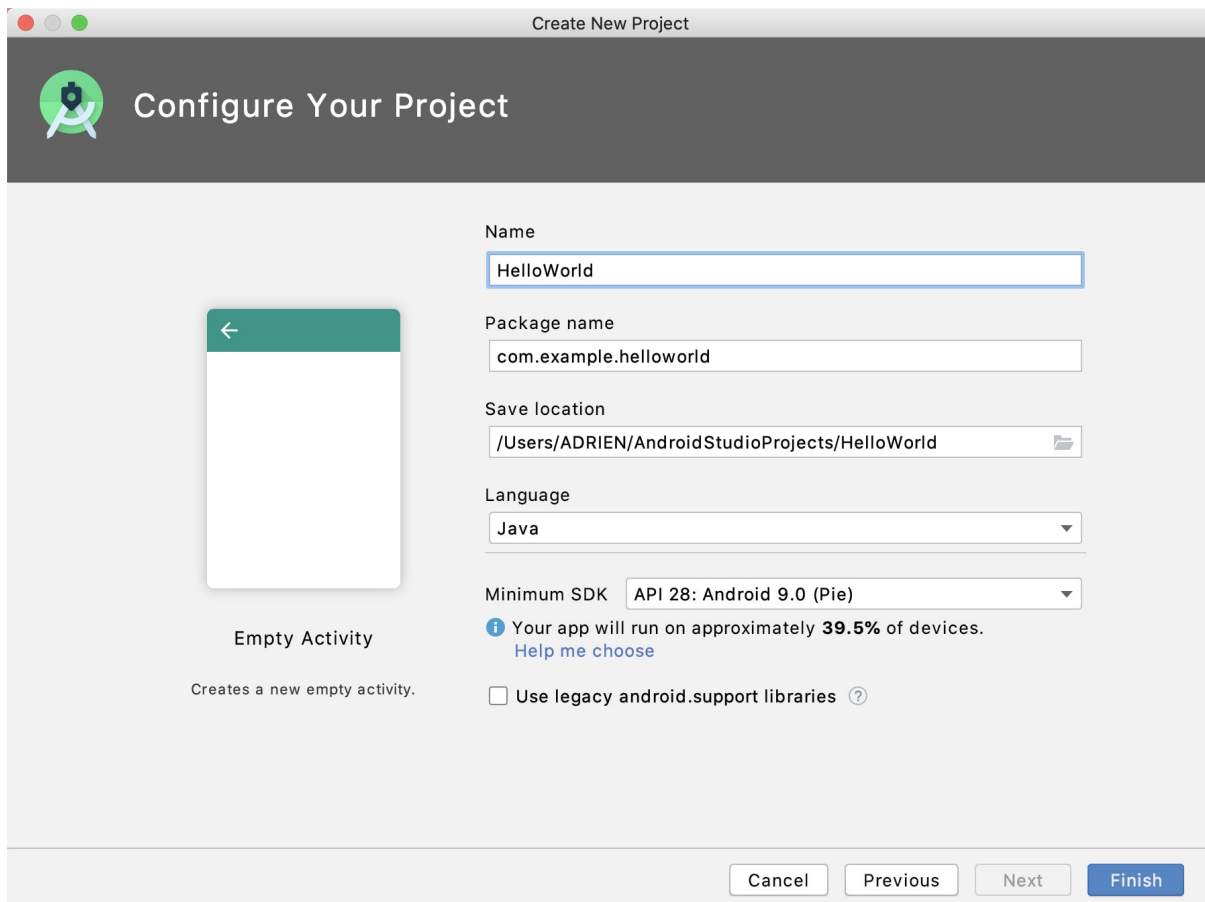
2) Création d'un premier projet Android

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.

La fenêtre suivante vous permet de choisir le type de projet que vous souhaitez développer. Sélectionnez ici **Phone et Tablet** puis le template **Empty Activity** (Création automatique d'une Activity avec un layout vide).



La fenêtre suivante vous permet de choisir le nom de votre projet, son nom de package et le dossier dans lequel l'installation s'effectuera. Changez le nom et mettez **HelloWorld**.



The screenshot shows the 'Create New Project' dialog in Android Studio. The dialog is titled 'Configure Your Project' and has a dark header bar with the Android Studio logo. On the left, there is a preview of an 'Empty Activity' with a green header bar and a white body. Below the preview, it says 'Empty Activity' and 'Creates a new empty activity.' On the right, there are several input fields and a checkbox. The 'Name' field is set to 'HelloWorld'. The 'Package name' field is set to 'com.example.helloworld'. The 'Save location' field is set to '/Users/ADRIEN/AndroidStudioProjects/HelloWorld'. The 'Language' dropdown is set to 'Java'. The 'Minimum SDK' dropdown is set to 'API 28: Android 9.0 (Pie)'. Below the 'Minimum SDK' field, there is a blue information icon and text: 'Your app will run on approximately 39.5% of devices. Help me choose'. At the bottom, there is a checkbox labeled 'Use legacy android.support libraries' which is unchecked. At the very bottom, there are four buttons: 'Cancel', 'Previous', 'Next', and 'Finish'.

Sélectionnez le langage **JAVA** ou **KOTLIN** pour ceux qui voudraient apprendre un nouveau langage plus permissif et le nouveau standard pour le développement Android.

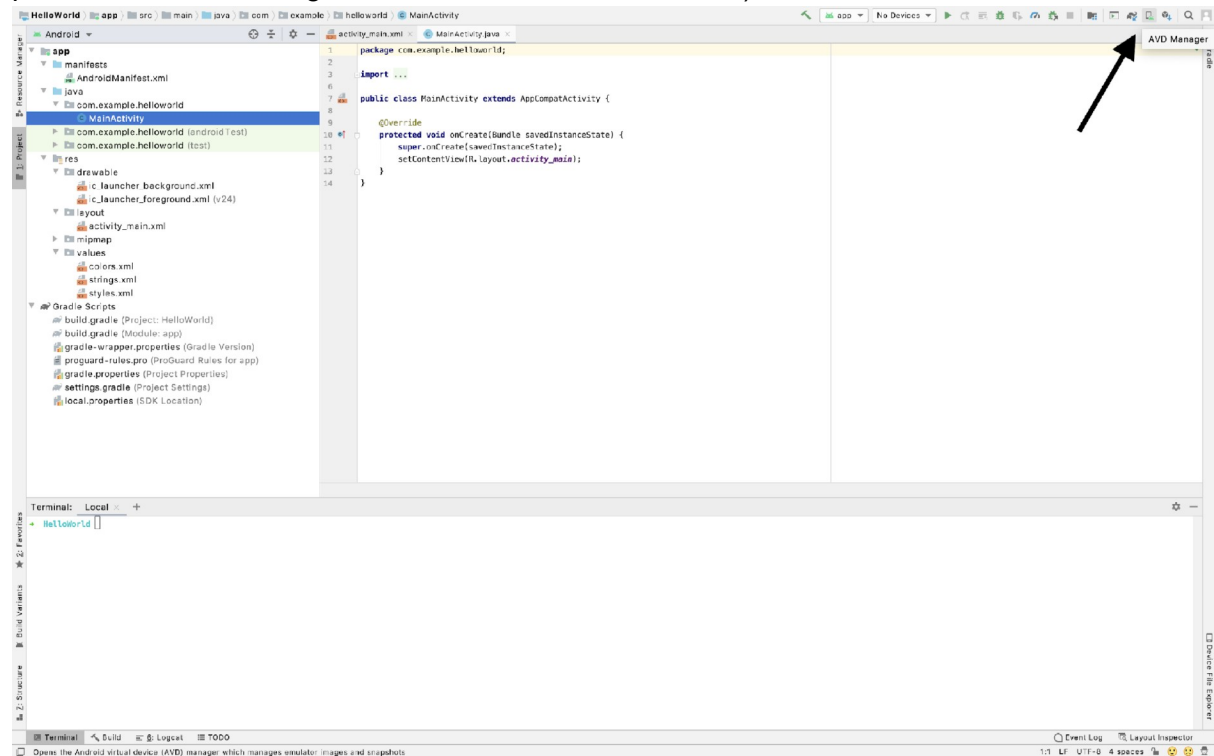
Pour le minimum SDK sélectionnez **celui de votre téléphone physique** si vous en avez un pour pouvoir builder votre application dessus plus tard. Sinon laissez celui sélectionné. Cela définit le niveau de fonctionnalité de votre application : s'il est trop élevé, peu de smartphones seront capables de faire tourner votre application, trop bas, vous n'aurez pas beaucoup de fonctions agréables à votre disposition.

NB : Vous pourrez toujours le changer plus tard par la suite.

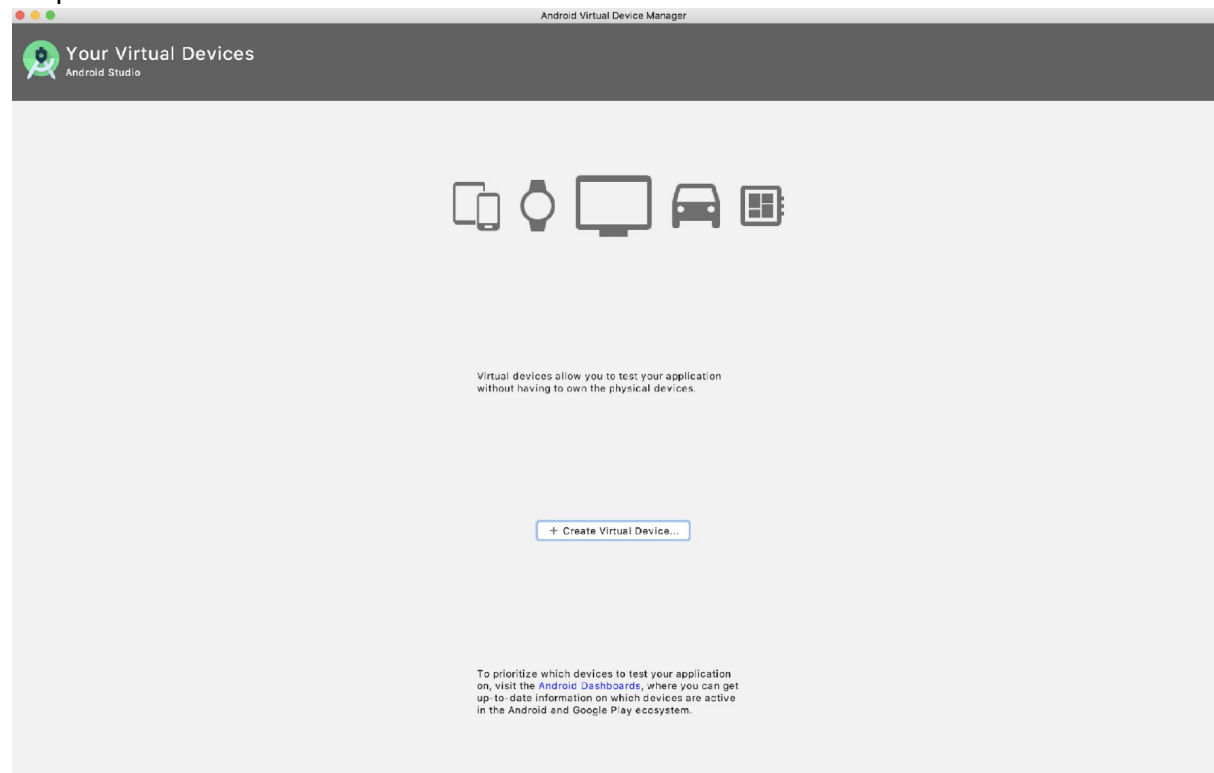
L'interface d'Android Studio est un peu déroutante au début car il y a beaucoup de choses à l'écran. Vous verrez rapidement qu'on ne se sert que d'une toute petite partie vite apprise. Il y a des boutons tout autour de la fenêtre : Project, Structure, Favorites, Build Variants, Terminal, Build, Logcat etc. Ce sont des boutons à bascule mutuellement exclusifs qui affichent des onglets ; celui qui est actif est grisé.

3) Création d'un émulateur

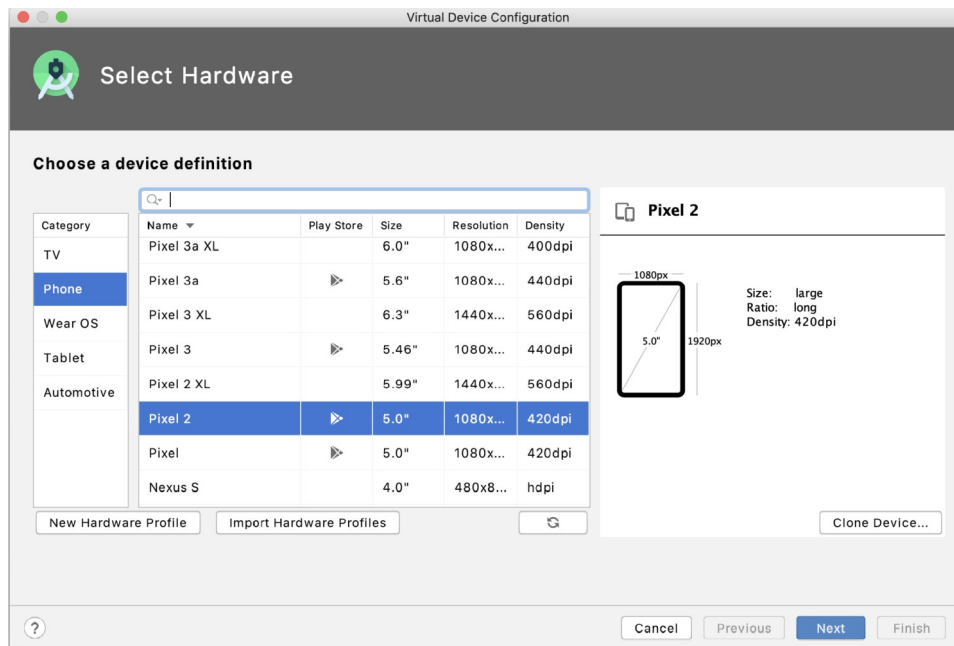
Pour ceux qui utilisent leur smartphone Android, passez votre chemin, allez directement page 6 si vous ne souhaitez pas utiliser d'émulateur. (ou si votre ordinateur a de faibles performances car très gourmand en ressources matériel).



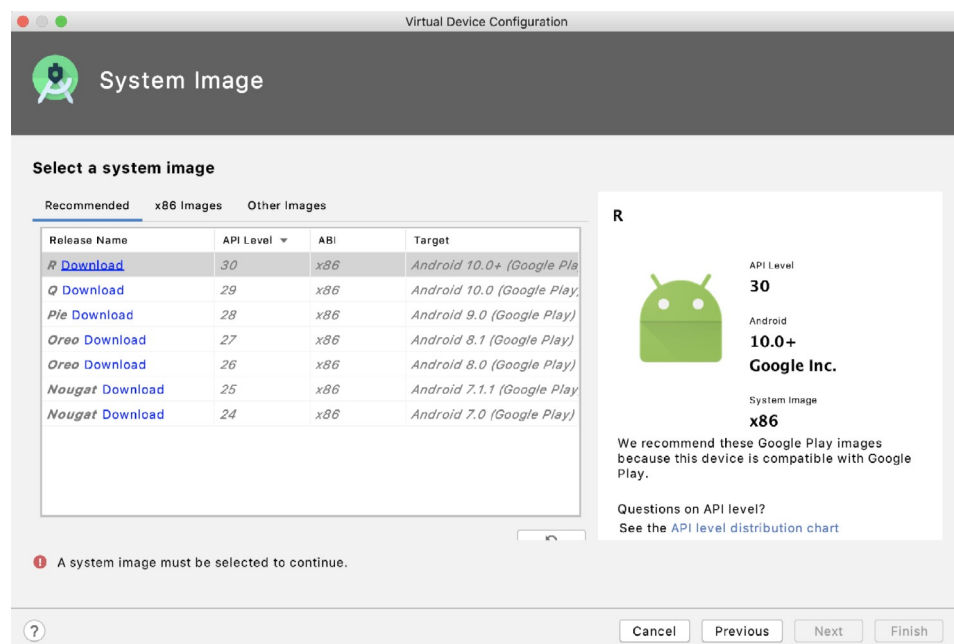
Cliquez sur **create virtual device** :



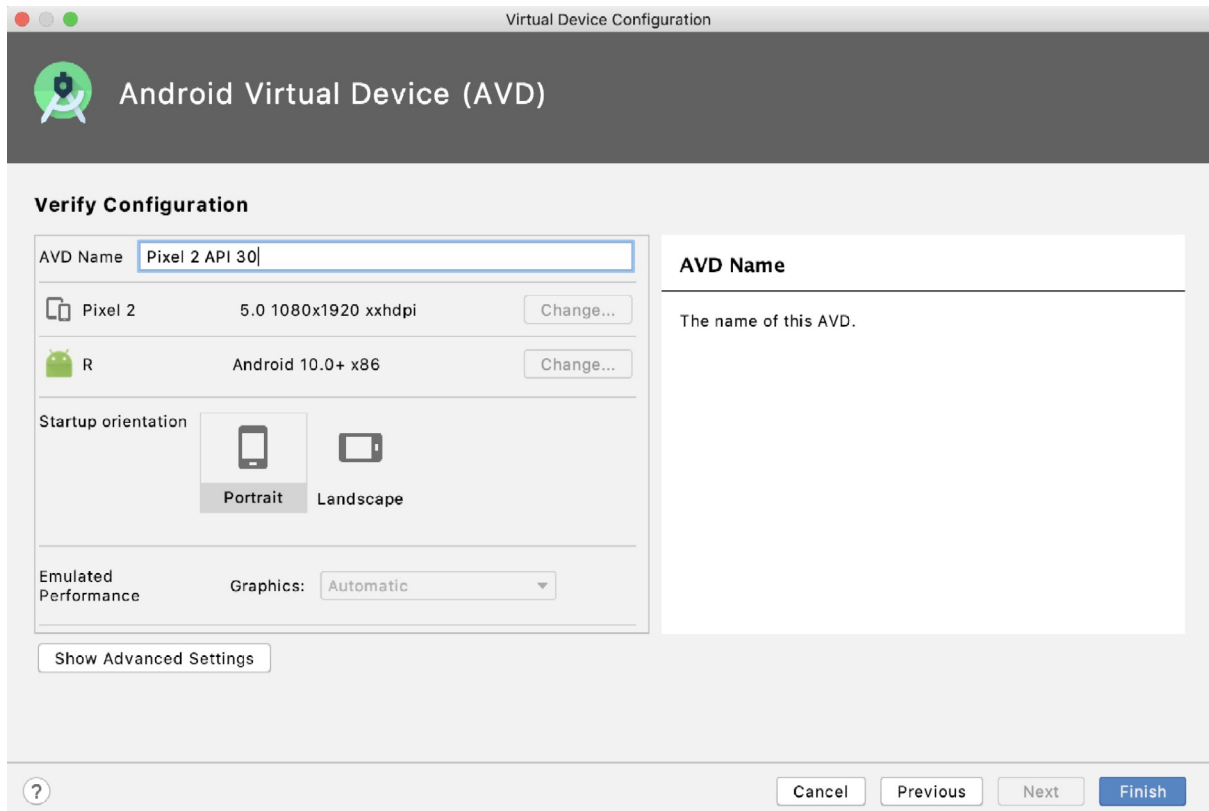
Choisissez un téléphone ou laissez la sélection par défaut, cliquez sur **Next**



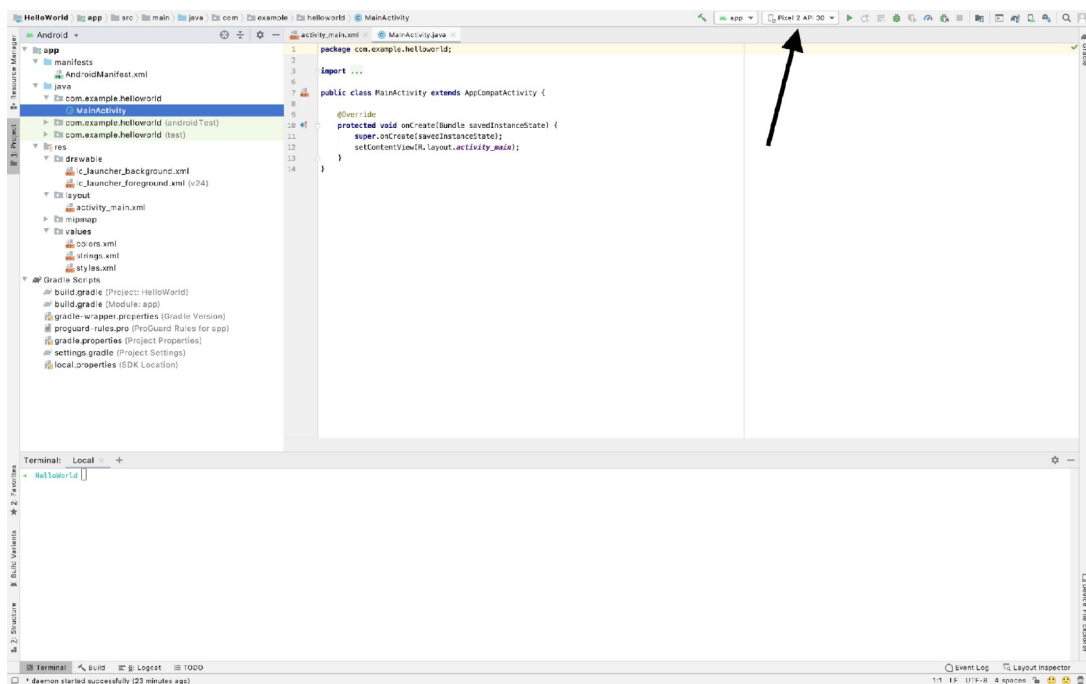
Si aucune image système n'est installée comme ici, sélectionnez celle qui correspond au minimum à celle que vous avez sélectionné à la création de ce projet (minimum SDK cf. page 3) le téléchargement se lance, attendez la fin...



Une fois le téléchargement fini, vous pouvez nommer votre émulateur dans la partie AVD Name et cliquez sur **Finish**.



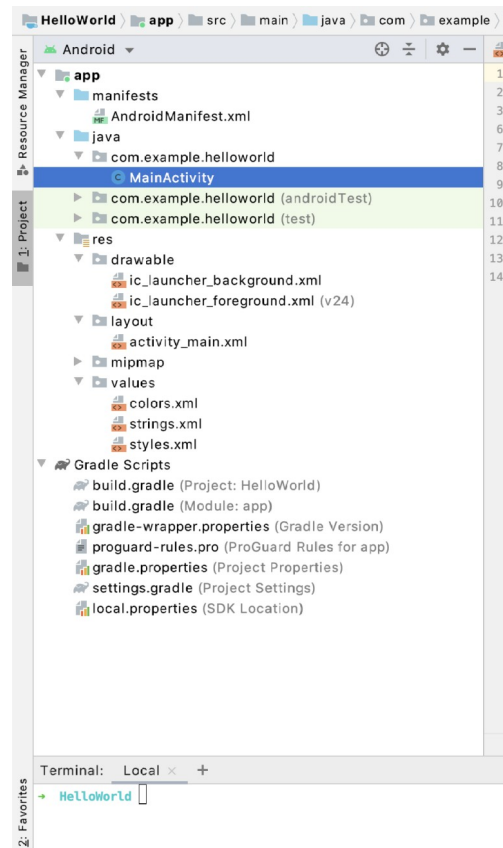
Une fois installé, vous verrez dans Android studio le nom de votre émulateur ou téléphone que vous avez branché. Pour ceux qui utilisent leur propre smartphone, [il faut activer en amont le mode développeur](#)



4) Structure du projet

La structure du projet est visible à gauche. Les fichiers sont arrangés de manière pratique, grâce au choix déroulant Android juste au dessus.

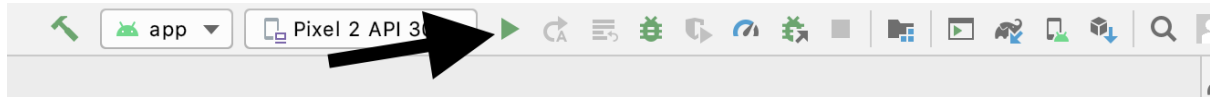
Prenez quelques instants pour déplier les dossiers manifests, java et res et regarder ce qu'il y a dedans.



- Le dossier du projet contient le premier build.gradle et le dossier **app**
- **app** contient le second build.gradle et un dossier **src**
- **app/src/main** contient le cœur du projet : manifeste, sources et ressources.
- **app/src/main/java** contient les dossiers correspondant au package que vous avez déclaré, avec les sources Java tout au bout du package
- **app/src/main/res/drawable** contiendra des images vectorielles utilisées dans votre projet
- **app/src/main/res/layout** contient les dispositions d'écran des activités de votre projet
- **app/src/main/res/menu** contient les menus contextuels et d'application du projet
- **app/src/main/res/mipmap** contient les icônes bitmap dans diverses résolutions
- **app/src/main/res/values** contient les constantes du projet, dont les chaînes de caractères avec toutes leurs traductions

5) Build (exécution) de l'application

Le lancement de l'application est simple. Il suffit de cliquer sur le bouton triangulaire vert dans la barre d'outils.



Ensuite, si vous avez plusieurs devices connectés, Android Studio vous demande quel émulateur il faut lancer. Il en faut un qui soit au moins du niveau d'API de votre application. Cochez **Use same device for futures launches**. Sinon il se lancera sur celui par défaut.

Votre application se lance directement sur votre émulateur ou sur votre smartphone. On observe ici un titre **HelloWorld** qui est le nom que vous avez donné à votre application ainsi que le texte centré "**Hello World !**".

6) Le LogCat : Affichage d'un message

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

```
package com.example.helloworld;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;

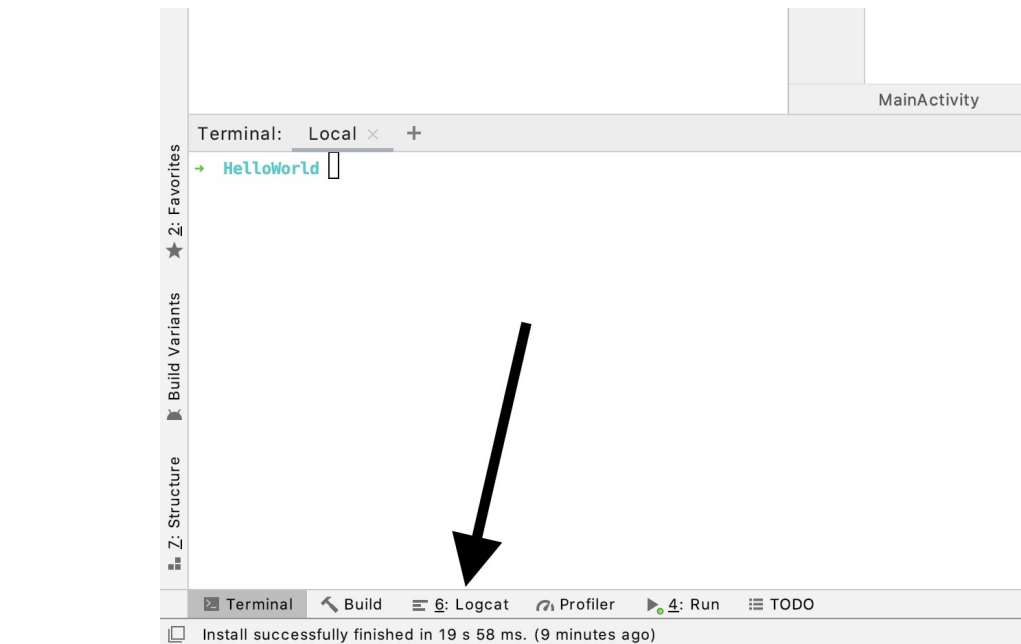
public class MainActivity extends AppCompatActivity {

    public static final String TAG = "hello";

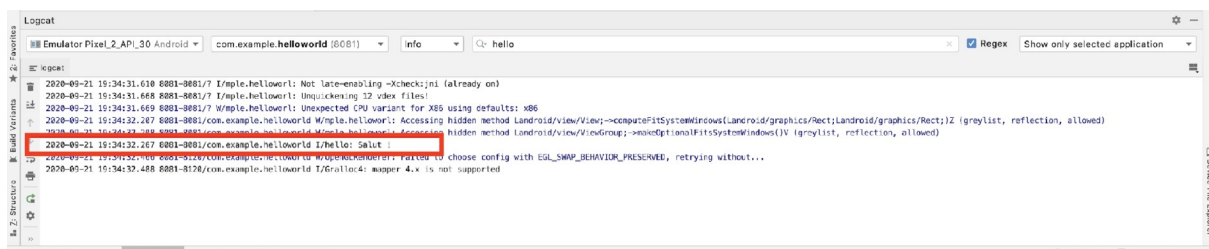
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Log.i(TAG, "Salut !");
    }
}
```

Ouvrir l'onglet Logcat



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.



Remplacez maintenant le code

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.i(TAG, "Salut !");
}
```

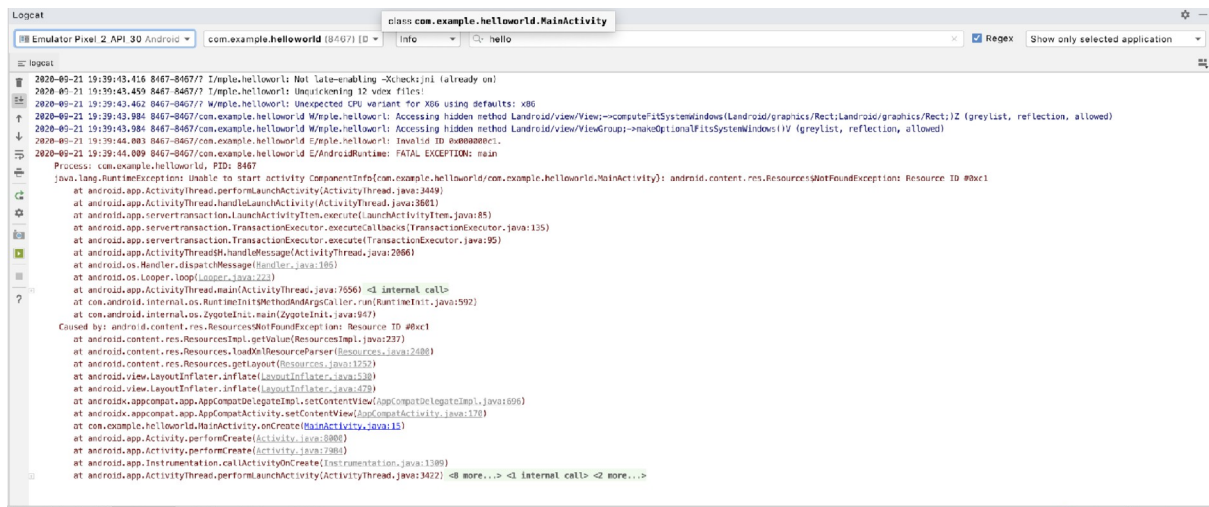
par :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(193);

    Log.i(TAG, "Salut !");
}
```

Lancez votre application et observez maintenant le logcat. Vous devriez obtenir une erreur affichée en rouge dans le logcat.



IMPORTANT : Tous les logs de crashes d'applications se retrouvent ici dans le logcat, très utile pour savoir ce qui a fait planter votre application. Regardez l'erreur plus en détail, le plus important se trouve souvent dès les 10 premières lignes :

```
2020-09-21 19:39:44.003 8467-8467/com.example.helloworld
E/mple.helloworl: Invalid ID 0x000000c1.
2020-09-21 19:39:44.009 8467-8467/com.example.helloworld
E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.example.helloworld, PID: 8467
    java.lang.RuntimeException: Unable to start activity
    ComponentInfo{com.example.helloworld/com.example.helloworld.MainActivity
    }: an      at
    androidx.appcompat.app.AppCompatActivityDelegateImpl.setContentView(AppCompatDel
    egateImpl.java:696)
        at
    androidx.appcompat.app.AppCompatActivity.setContentView(AppCompatActivity
    y.java:170)
        at
    com.example.helloworld.MainActivity.onCreate(MainActivity.java:15)
        at android.app.Activity.performCreate(Activity.java:8000)
        at android.app.Activity.performCreate(Activity.java:7984)
        at
    com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.
    java:592)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:947)
```

On peut retrouver le mot **Invalid ID**
ou encore un descriptif de ce qui s'est passé **Unable to start activity,**

mais surtout le fichier et la ligne exacte où le crash s'est produit.

`com.example.helloworld.MainActivity.onCreate(MainActivity.java:15)`

Reprenons maintenant ce fichier et la ligne incriminée :



Ici la méthode setContentView() nécessite l'id d'un layout pour afficher une vue, nous avons setté aléatoirement à 193 au lieu de R.layout.activity_main d'où ce crash. Remplacez maintenant le code par :

```
package com.example.helloworld;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    public static final String TAG = "hello";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        String text = "Bonjour Adrien";

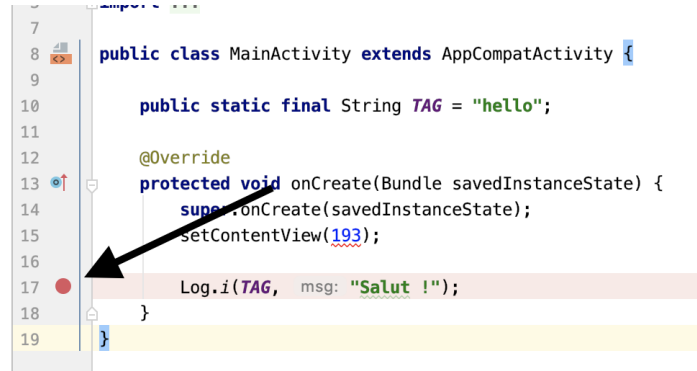
        Log.i(TAG, text);
    }
}
```

Nous avons simplement ajouté ici une variable texte contenant "Bonjour Adrien"

7) Lancement en mode debug

Le SDK android nous délivre un outil magique pour le développeur : le debugger. Il va permettre de mettre des points d'arrêts dans le code. Ce qui signifie que lorsque vous allez lancer votre application, le programme va s'arrêter à l'endroit exact où vous aurez placé ce(s) point(s) d'arrêt(s).

Plaçons maintenant un point d'arrêt dans la méthode onCreate à la ligne 19 pour cela cliquez simplement dans la marge à côté du numéro de ligne.

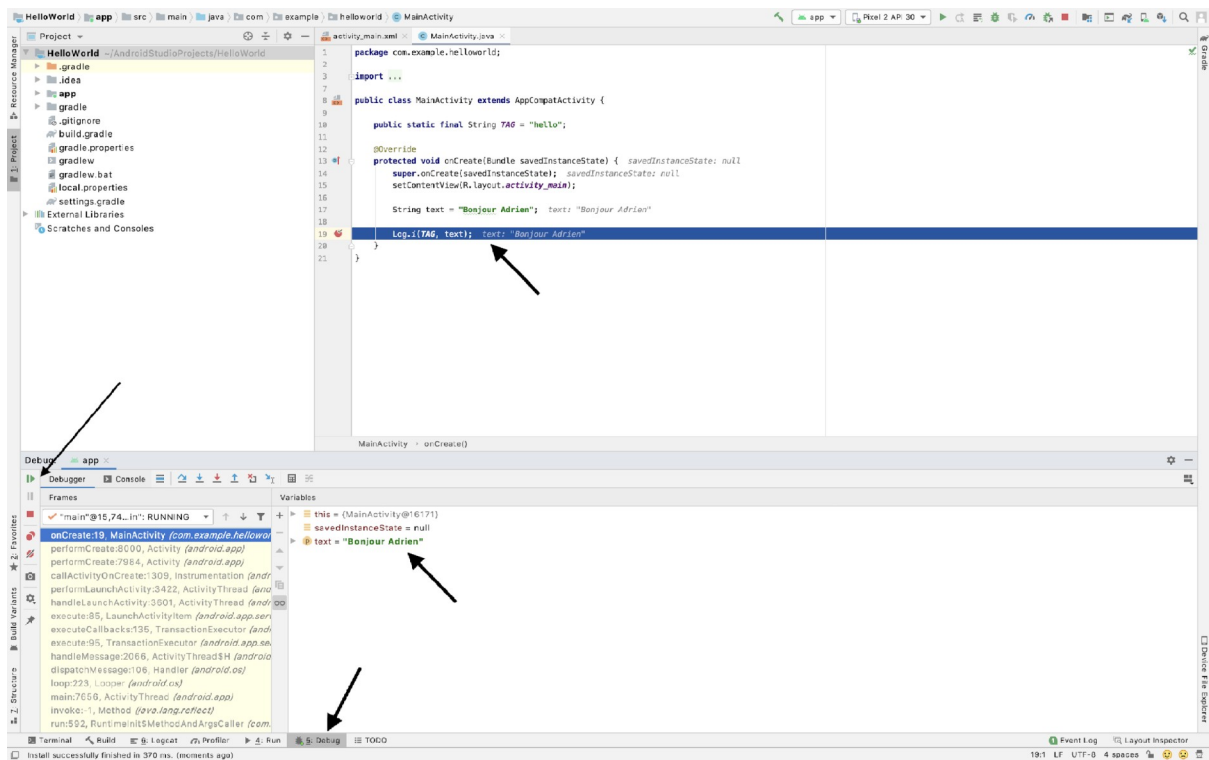


Lançons maintenant notre application en mode debug, pour cela cliquez sur l'icône en forme d'insectes à côté de celle pour lancer le programme.

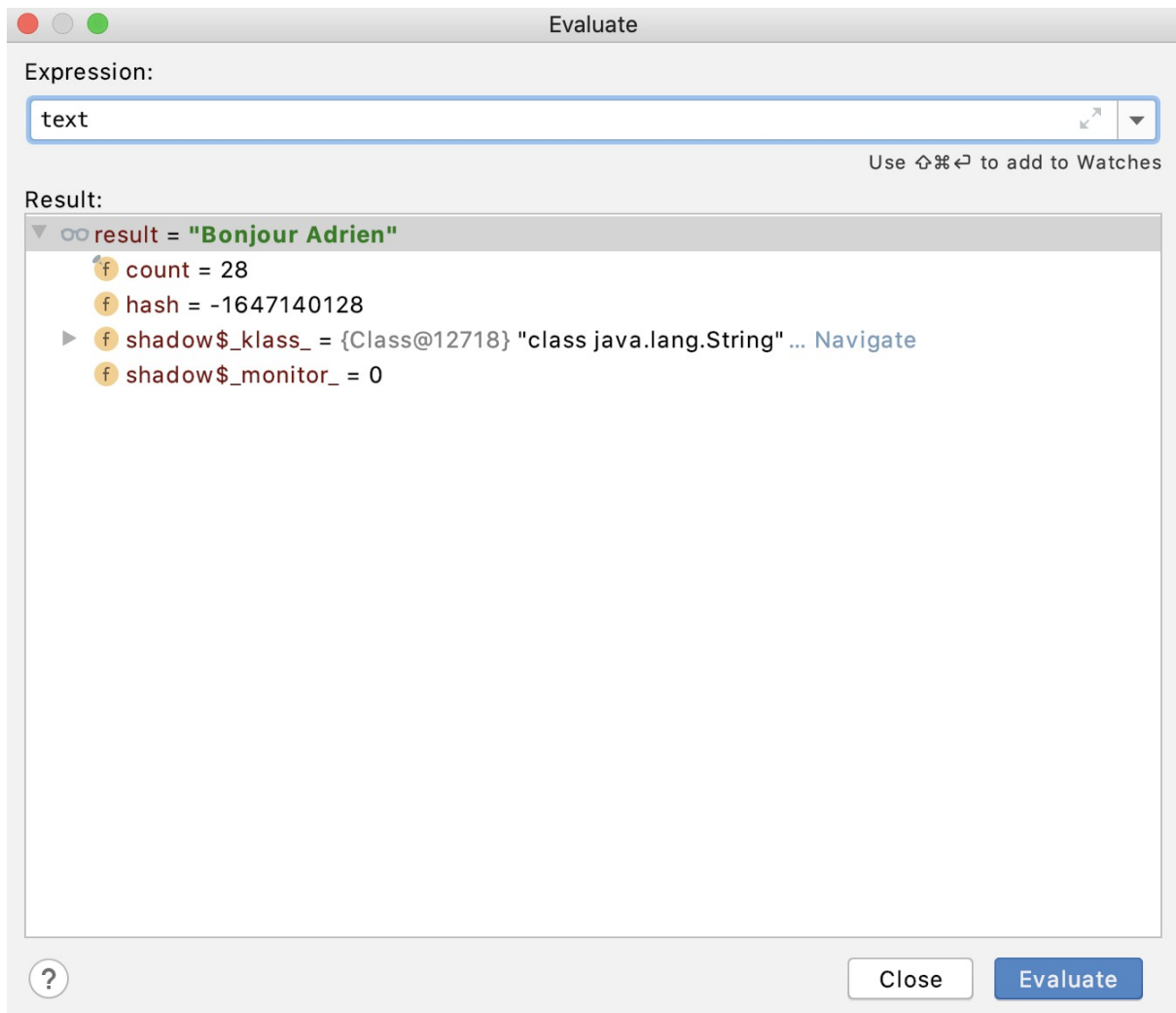


L'application est alors "lancée" en mode debug, ce qui signifie qu'elle est à l'écoute de points d'arrêts.

Si tout s'est bien passé, vous devriez avoir votre application avec une page blanche et votre ligne de code surlignée en bleu dans Android Studio. Ouvrez l'onglet Debug si cela n'est pas déjà fait. Vous pouvez alors observer à l'instant T les valeurs de vos variables dans cet onglet. Très pratique pour trouver où est le problème par exemple. Pour passer au point d'arrêt suivant ou continuer l'exécution, cliquez sur la flèche à gauche de l'onglet debug.



NB: Vous pouvez aussi faire un clic droit à tout moment en mode debug dans android studio puis **evaluate expression** et saisir votre variable voulue ou une expression complexe.



Exercice 1:

Rajoutez ceci dans la méthode onCreate() de MainActivity :

```
int v = 54;
int n = v / 9 - 2;
Log.i(TAG, "n = "+n);
int f = factorielle(n);
Log.i(TAG, n.toString() + "! = "+f);
```

Ajouter la méthode suivante :

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

Mettez un point d'arrêt sur la première instruction, int v = 54; puis lancez en mode débogage

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).