Licence 1

Informatique, Développement d'Application

Cours : Initiation au développement mobile

Séquence 2 : Prise de main de Android studio

UVS
UNIVERSITÉ VIRTUELLE DU SÉNÉGAL

PLAN

introduction

Introduction et configuration

- installation de IDE (Android studio)
- Configuration de IDE
- Installation des paquets supplémentaires et des mises à jours

Notre première application Android

- Création du projet “Uvs-initiation”
- Explication de l’arborescence du projet

Création d’un AVD

Exécution de l’application

INTRODUCTION

Sous titre du slide

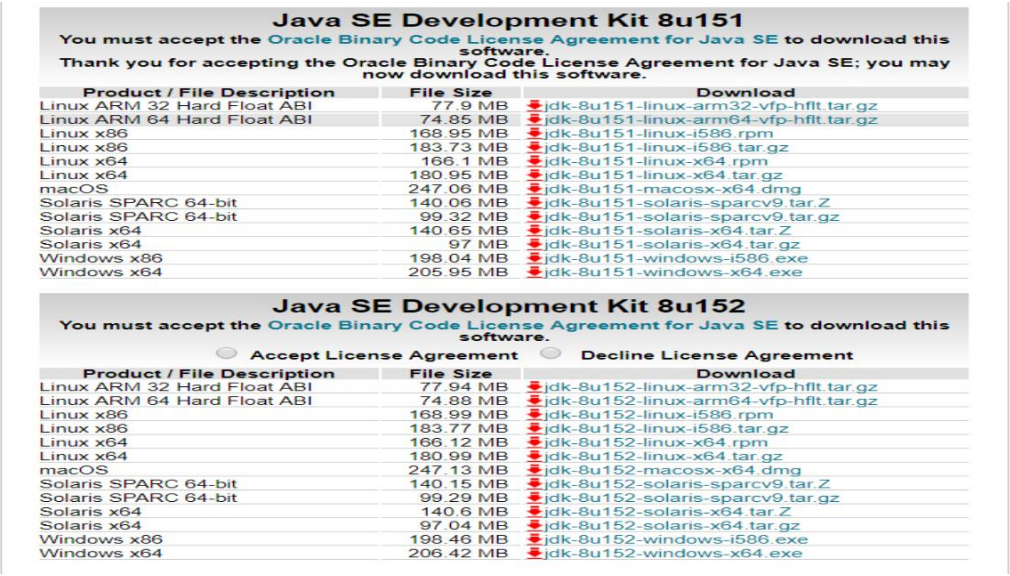
- Depuis décembre 2014, l'environnement de développement Eclipse, avec son module ADT ont prit leur retraite afin de faire place au tout nouvel environnement **Android Studio**

- Dans ce séquence nous verrons comment prendre en main ce nouvel environnement, très complet, surtout quelles principales nouveautés ont été apportées par cet IDE.

Installation de l'IDE Android Studio (1/4)

Dans cette section nous allons décrire la procédure d'installation d'un environnement de développement Android.

a. Téléchargez le dernier JDK (*Java Development Kit*) que vous pouvez trouver sur le site d'Oracle (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)



Java SE Development Kit 8u151

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.9 MB	jdk-8u151-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.85 MB	jdk-8u151-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.95 MB	jdk-8u151-linux-i586.rpm
Linux x86	183.73 MB	jdk-8u151-linux-i586.tar.gz
Linux x64	166.1 MB	jdk-8u151-linux-x64.rpm
Linux x64	180.95 MB	jdk-8u151-linux-x64.tar.gz
macOS	247.06 MB	jdk-8u151-macosx-x64.dmg
Solaris SPARC 64-bit	140.06 MB	jdk-8u151-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.32 MB	jdk-8u151-solaris-sparcv9.tar.gz
Solaris x64	140.65 MB	jdk-8u151-solaris-x64.tar.Z
Solaris x64	97 MB	jdk-8u151-solaris-x64.tar.gz
Windows x86	198.04 MB	jdk-8u151-windows-i586.exe
Windows x64	205.95 MB	jdk-8u151-windows-x64.exe

Java SE Development Kit 8u152

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.94 MB	jdk-8u152-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u152-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.99 MB	jdk-8u152-linux-i586.rpm
Linux x86	183.77 MB	jdk-8u152-linux-i586.tar.gz
Linux x64	166.12 MB	jdk-8u152-linux-x64.rpm
Linux x64	180.99 MB	jdk-8u152-linux-x64.tar.gz
macOS	247.13 MB	jdk-8u152-macosx-x64.dmg
Solaris SPARC 64-bit	140.15 MB	jdk-8u152-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u152-solaris-sparcv9.tar.gz
Solaris x64	140.6 MB	jdk-8u152-solaris-x64.tar.Z
Solaris x64	97.04 MB	jdk-8u152-solaris-x64.tar.gz
Windows x86	198.46 MB	jdk-8u152-windows-i586.exe
Windows x64	206.42 MB	jdk-8u152-windows-x64.exe

Installation de l'IDE Android Studio (2/4)

b. Désinstallez des éventuelles versions antérieures du
JDK c. Installer le nouveau JDK
d. Télécharger 'Android Studio. Il contient l'environnement de développement, SDK (*Software Development Kit*) Android avec la dernière version de la plateforme, ainsi qu'un émulateur

https://developer.android.com/studio/index.html

Installation de l'IDE Android Studio (3/4)

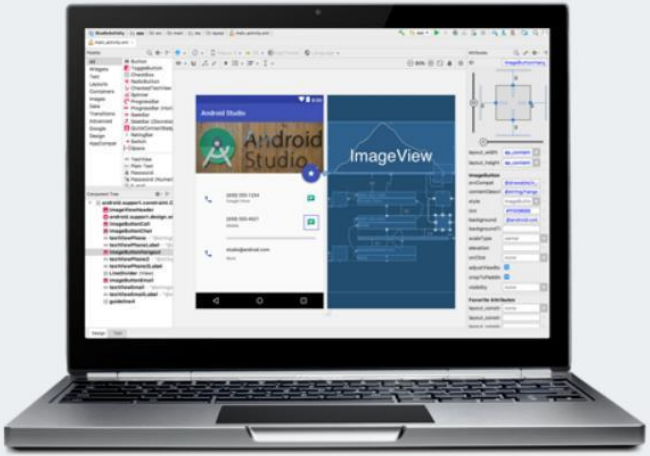
Android Studio

The Official IDE for Android

Android Studio provides the fastest tools for building apps on every type of Android device.

World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow you to focus on building unique and high quality apps.

DOWNLOAD ANDROID STUDIO
3.0 FOR WINDOWS (681 MB)



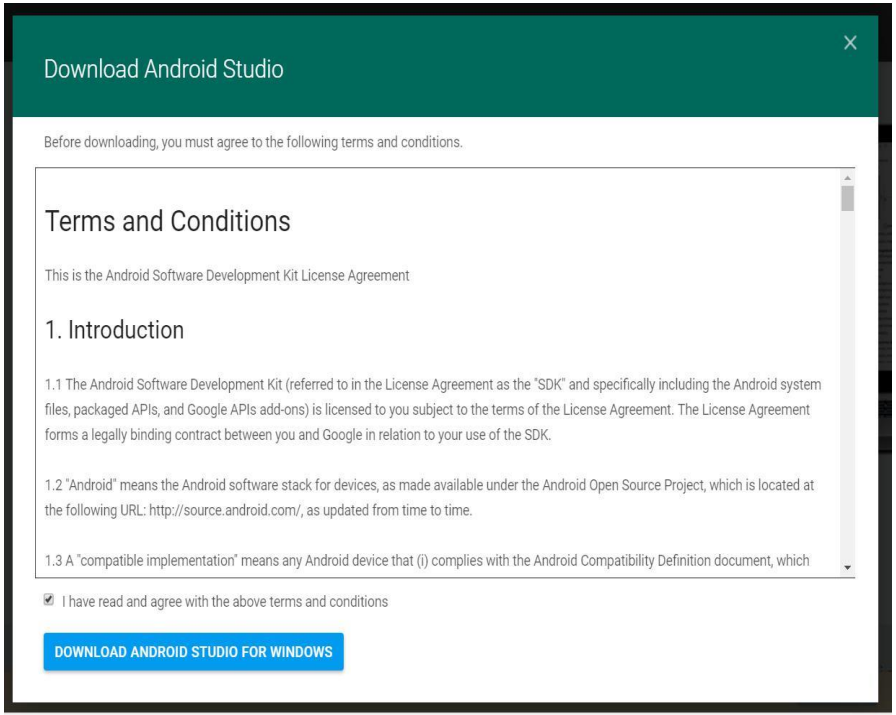
› Read the docs › See the release notes

Cliquez sur DOWNLOAD ANDROID STUDIO pour commencer le téléchargement

Installation de l'IDE Android Studio (3/4)

Accèptez les conditions et terminez le téléchargement de Android Studio

e. Lancez l'exécutable pour démarrer l'installation et suivez le é tapes



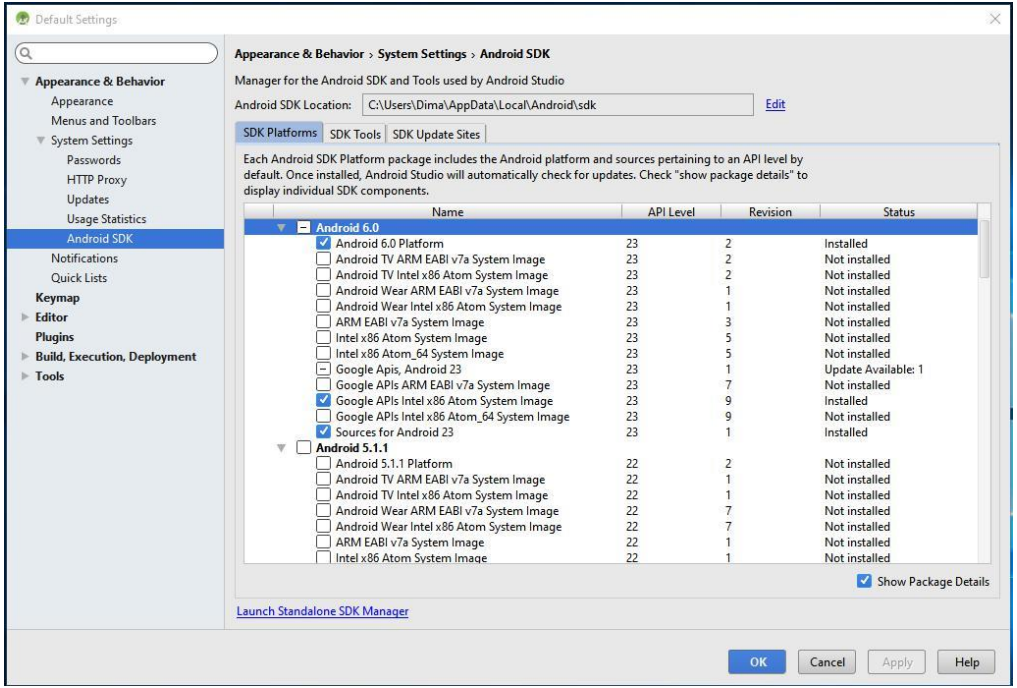
BN : Il faut exécuter les étapes dans l'ordre cité ci -dessous.(de a à e)

Configuration de l' IDE (1/3)

Installation des paquets supplémentaires et des mises à jours

- Lancez Android Studio
- Nous commencerons par nous assurer que nous possédons tout ce qu'il faut pour développer. Dans la page de démarrage, sélectionnez *Configure> SDK Manager*.
- Dans la figure suivante vous verrez la version du SDK installée (avec les mises jour disponibles) et aussi la version de l'API (*Application Programming Interface*) installée et la version du OS pour laquelle elle vous permettra de développer. Installez les éventuelles mises à jour.
- Assurez vous de cocher au moins un *System Image* pour l'émulateur

Configuration de l' IDE (2/3)



Configuration de l' IDE (3/3)

- **Dans l'onglet *SDK Tools* assurez vous d'avoir au moins**
  - Android SDK Build Tools
  - Android SDK Tools
  - Android SDK Platform Tools
  - Android Support Library
  - Android Support Repository
  - Google Repository
  - Google Play Services

NOTRE PREMIERE APPLICATION ANDROID (1/5)

Création d'un projet et d'une application ''uvs_intiation ''

a. Dans le menu *Quick Start*, sélectionnez *Start a new Android Studio Project*, et renseignez les champs comme dans la figure suivante:

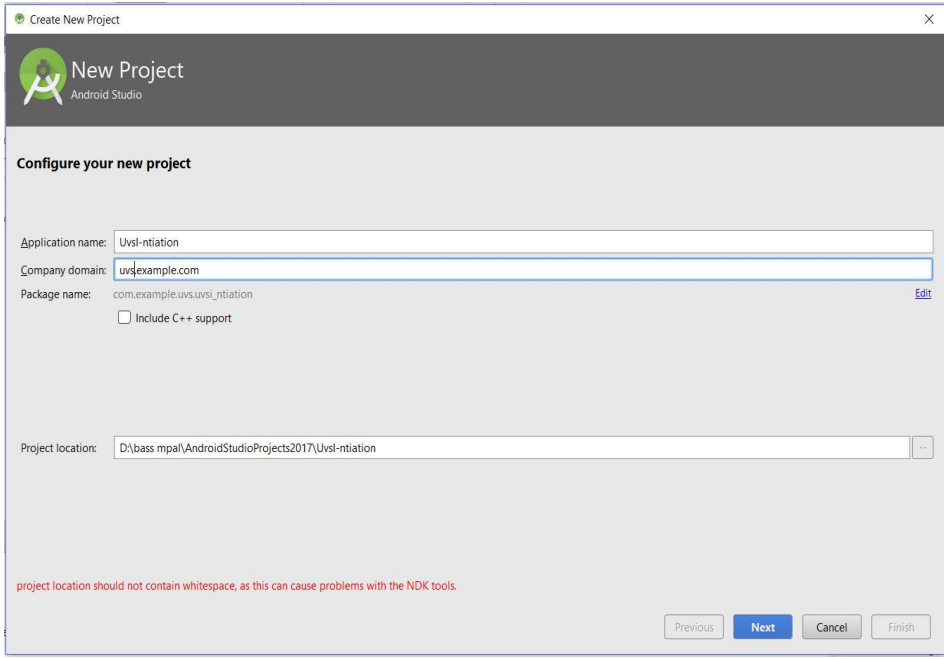
Application name : c'est le nom qui va apparaitre dans la liste des applications sur l'appareil et dans le Play Store.

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.

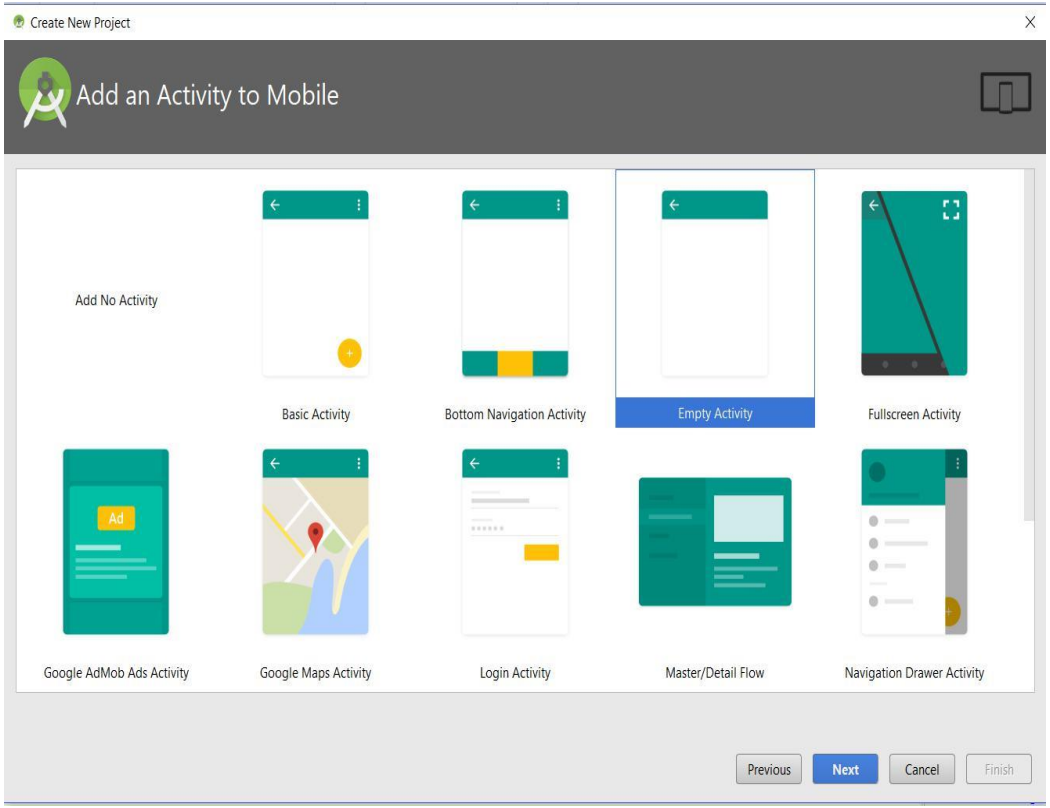
NOTRE PREMIERE APPLICATION ANDROID (2/5)

b . Cliquez sur *Next*. Nous arrivons à la création d'une activité (un écran avec une interface graphique).



NOTRE PREMIERE APPLICATION ANDROID (3/5)

c .Sélectionnez *Empty Activity* et cliquez *Next*



NOTRE PREMIERE APPLICATION ANDROID (4/5)

d. Renseignez les champs comme dans la figure suivante

Vous pourriez choisir l'utilisation de fragments, mais pour faire simple nous poursuivrons sans fragments.

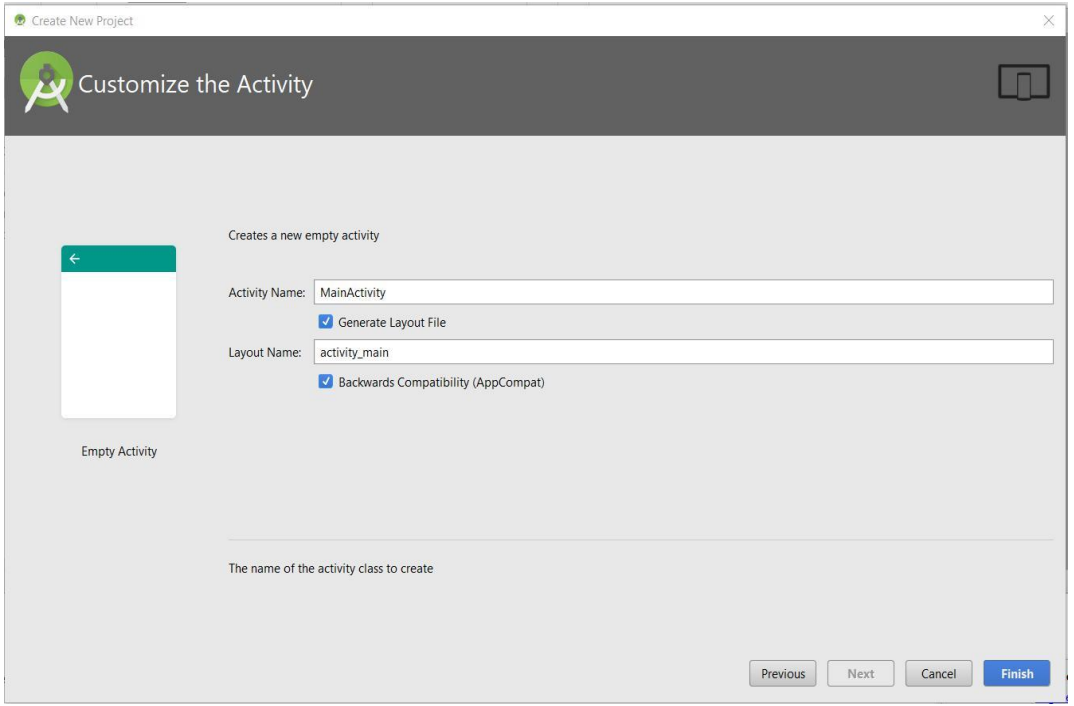
Chaque activité dispose d'un *layout* qui définit la façon dont les composants seront disposés sur l'écran.

Une activité peut être divisée en portions (ou fragments) chacune ayant son propre *layout*.

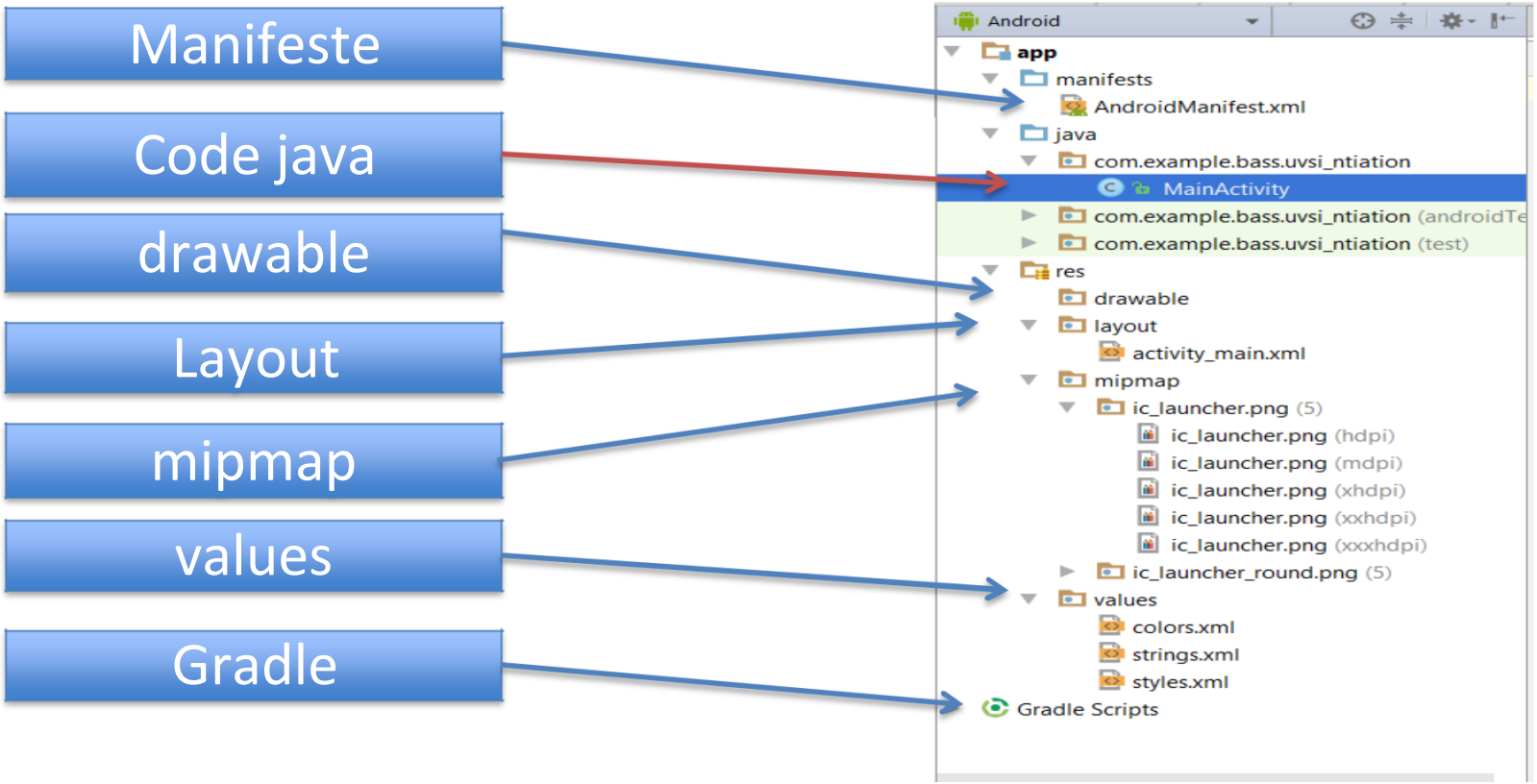
La notion de fragment a été introduite pour favoriser la réutilisabilité de morceaux d'activité (un fragment peut être définit une fois et réutilisé dans plusieurs activités)

NOTRE PREMIERE APPLICATION ANDROID (5/5)

e. Cliquez sur *Finish*, le projet est crée.



Explication de l'arborescence du projet (1/9)



Explication de l'arborescence du projét (2/9)

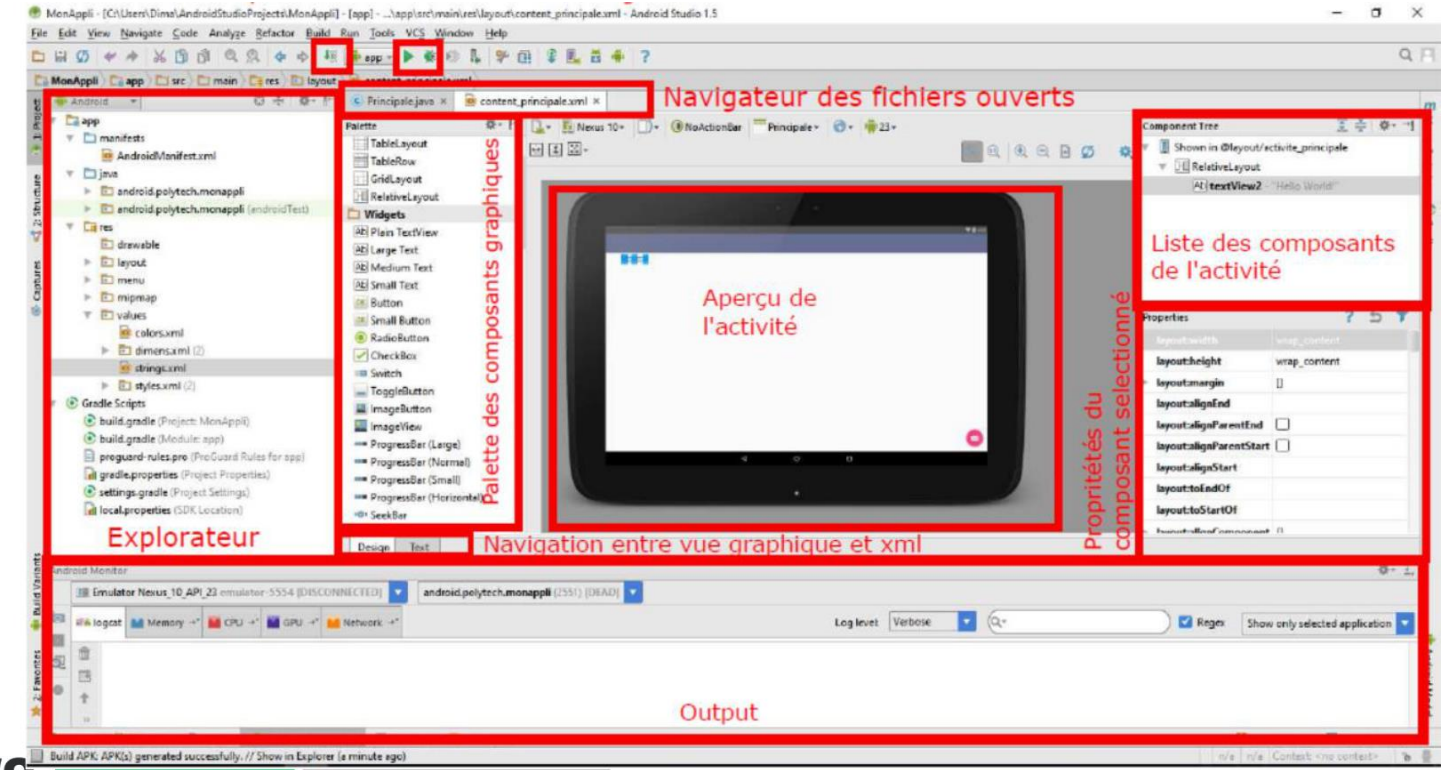
- Tout projet Android doit respecter une hiérarchie bien précise qui permettra au compilateur de retrouver les différents éléments et ressources lors de la génération de l'application.
  Cette hiérarchie favorise la modularité des applications Android.

- 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 (situé dans le répertoire

Explication de l'arborescence du projét (3/9)

Aperçu de l'interface Android Studio



Explication de l'arborescence du projét (4/9)

Manifests

- 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.

Explication de l'arborescence du projét (5/9)

Le code java

- Le code de la classe principale de l'application (*MainActivity.java*) est situé dans le sous dossier *com.example.bass.uvsi_nitiation* de *java*.

- C'est dans le dossier *java* que seront enregistrées toutes les classes que nous allons créer dans ce projet uvs_initiation

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

Explication de l'arborescence du projét (6/9)

Layout

- **layout** regroupe les fichiers XML qui définissent la disposition des composants sur l'écran. Il contient déjà, dès la création du projet, le *layout* de l'activité principale que nous avons créée.

- Deux fichiers XML sont crée par Android Studio : *activite_main.xml* et *content_main.xml*.

- Le premier définit l'apparence générale de l'activité : sa disposition, sa taille, sa barre d'outil, ainsi que son *layout* qui n'est autre que *content_main.xml*.

- Ce sera donc ce dernier que nous manipulerons pour disposer les composants de l'activité et créer notre interface graphique.

Explication de l'arborescence du projét (7/9)

Drawable

- **drawable** contient tout élément qui peut être dessiné sur l'écran : images (en PNG de préférence), formes, animations, transitions, etc..
- Cinq dossiers *drawable* permettent aux développeurs de proposer des éléments graphiques pour tout genre d'appareil Android en fonction de sa résolution. En populant correctement ces dossiers on peut ainsi créer des applications avec une interface qui s'adapte à chaque résolution d'écran avec un seul fichier .apk.
  - ldpi low-resolution dots per inch. Pour des images destinées à des écrans de basse résolution (~120dpi)
  - mdpi pour des écrans de moyenne resolution (~160dpi)
  - hdpi pour des écrans de haute résolution (~240dpi)
  - xhdpi pour des écrans ayant une extra haute résolution (~320dpi)
  - xxhdpi pour des écrans ayant une extra extra haute résolution (~480dpi).

Explication de l'arborescence du projét (8/9)

Menu, mipmap,values

- **menu** contient les fichiers XML définissant les menus
- **mipmap** contient les images de l'icône de votre applications sous différentes résolutions.
- **values** contient les fichiers XML qui définissent des valeurs constantes (des chaines de caractères, des dimensions, des couleurs, des styles etc.)

Explication de l'arborescence du projét (9/9)

Gragle

- **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.

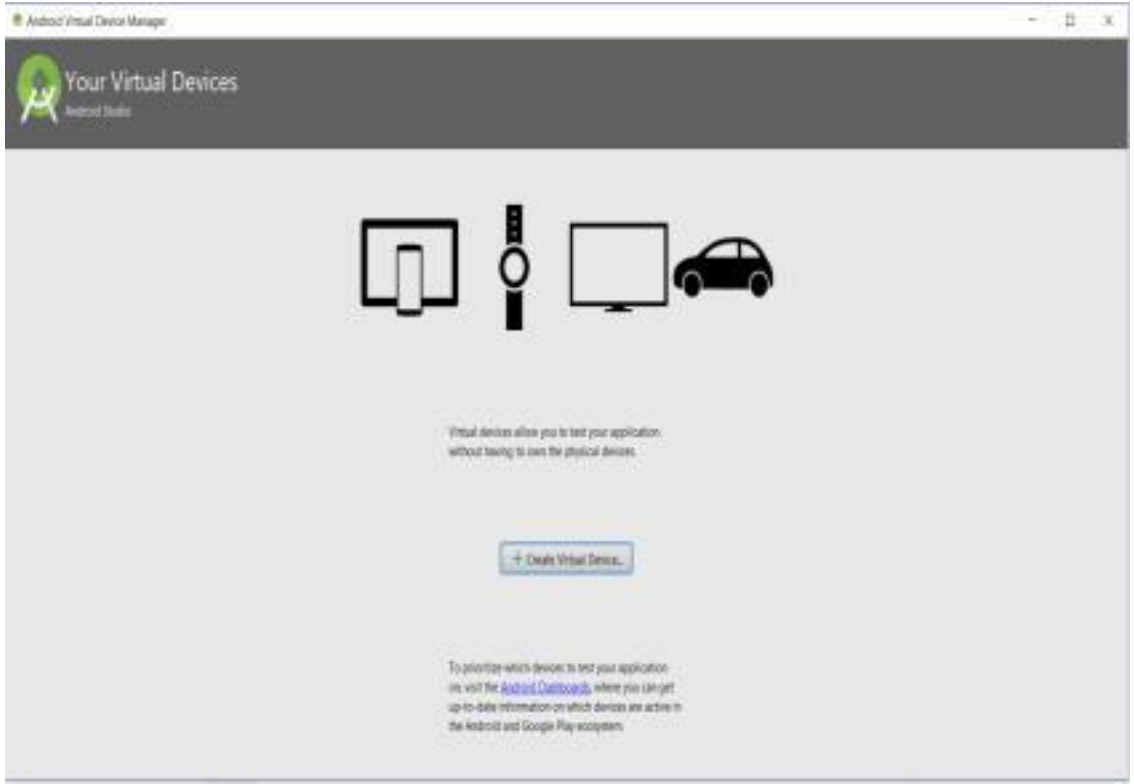
Création d'un emulateur AVD (2/5)

EMULATEUR

- Un émulateur permet de reproduire le comportement d'un appareil réel d'une façon virtuelle
- L'utilisation d'un émulateur nous évite d'avoir à charger à chaque fois l'application dans un appareil pour la tester
- On pourra ainsi lancer l'application dans l'IDE et elle s'exécutera sur un appareil virtuel appelé *Android Virtual Device* AVD qui émule le comportement d'un téléphone, une tablette ou autre
- Pour configurer un émulateur allez dans *AVD Manager*,

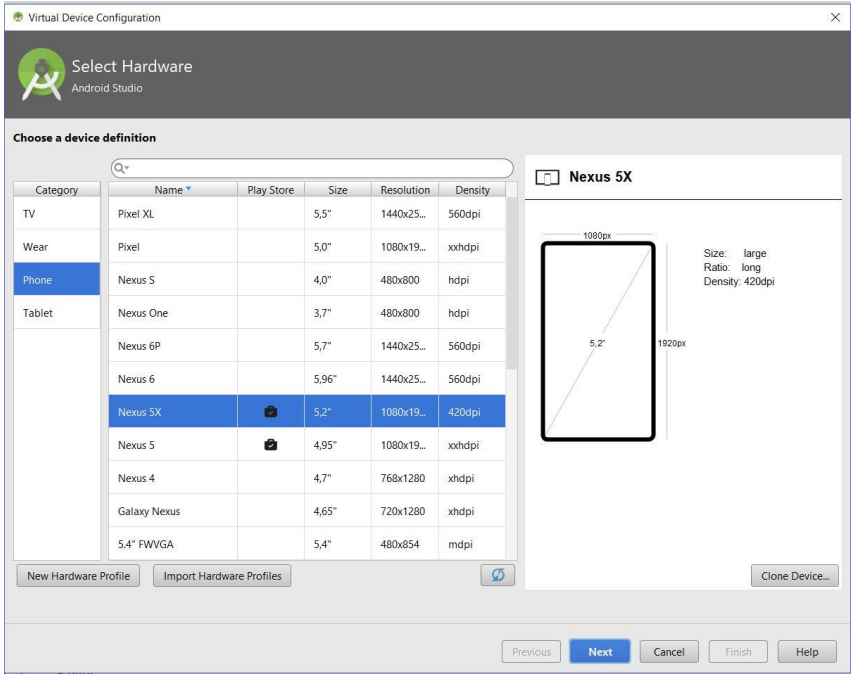
Création d'un emulateur AVD (2/5)

Cliquez sur creqte virtual divice



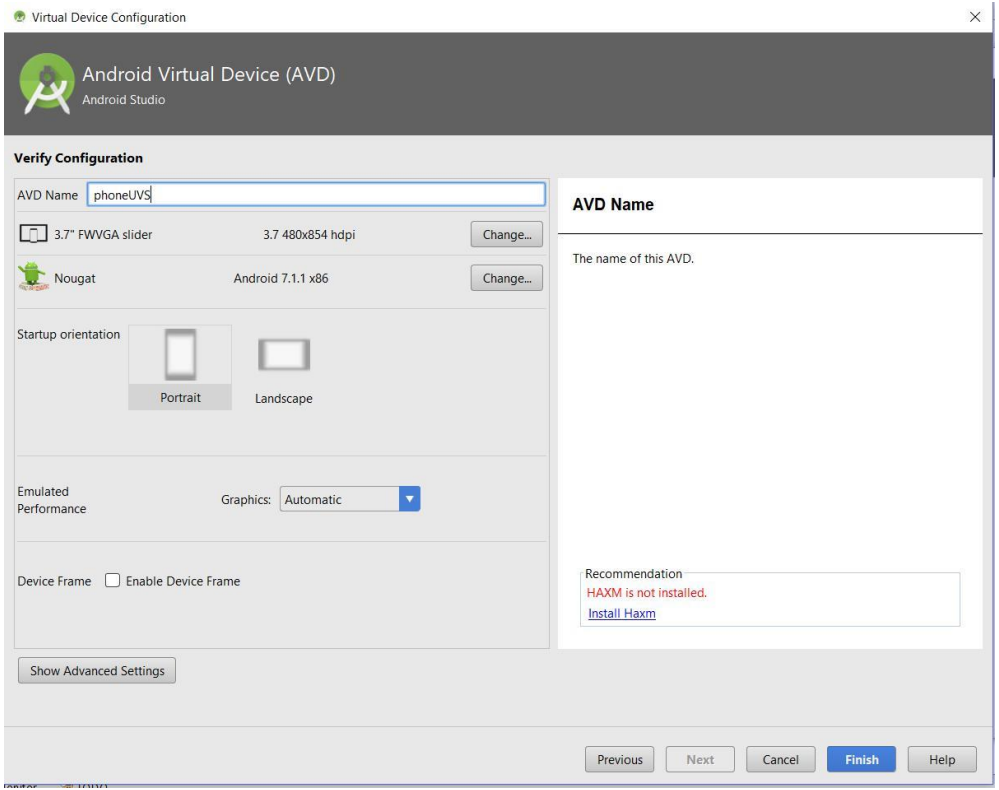
Création d'un emulateur AVD (3/5)

- Choisissez votre téléphone et cliquez Next



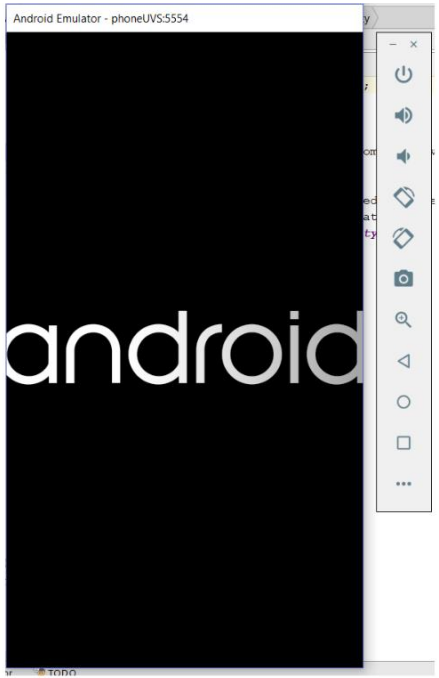
Création d'un emulateur AVD (4/5)

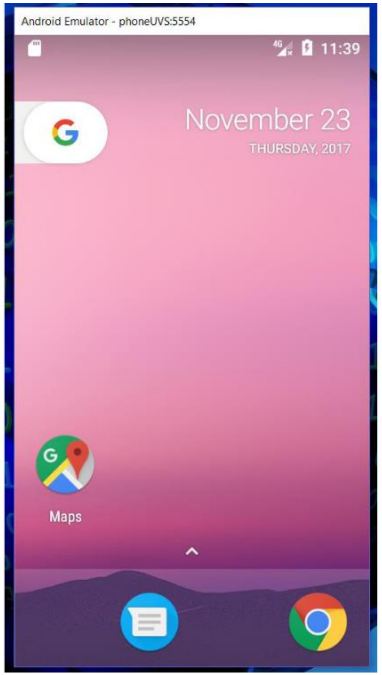
- Remonez le divice «phoneUVS» et cliquez sur finich

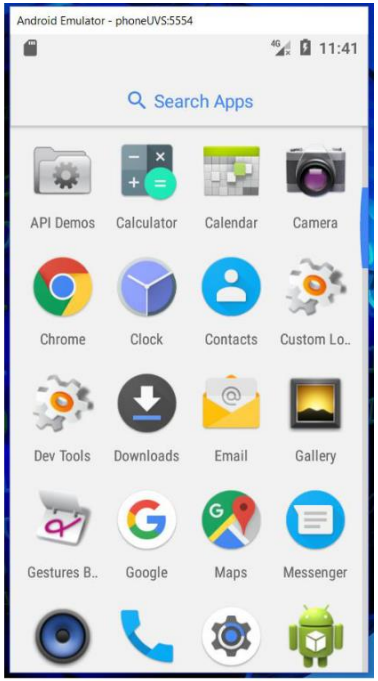


Création d'un émulateur AVD (5/5)

- L'émulateur se lance, ça peut prendre quelques minutes soyez patients.



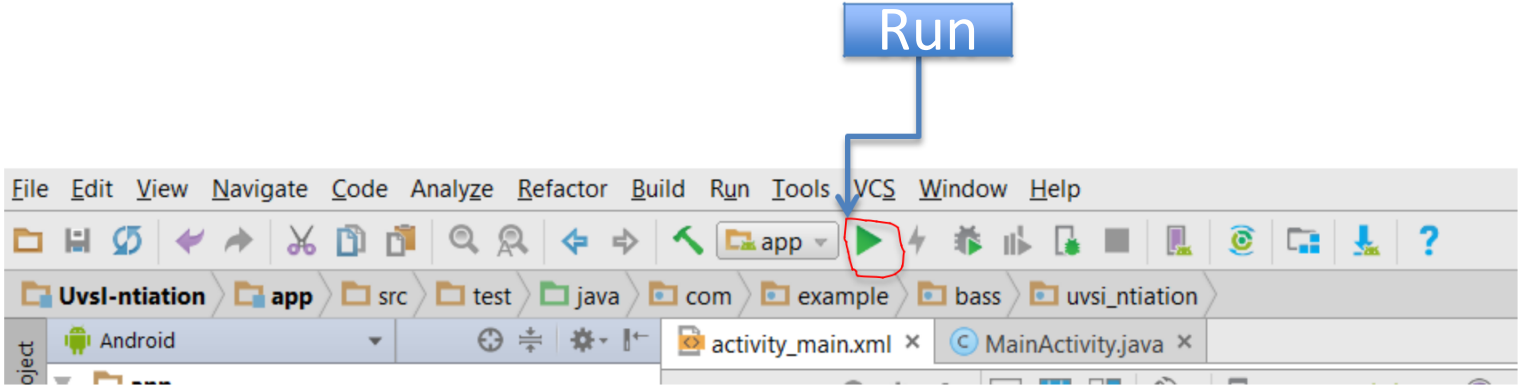




Exécution de l'application (1/)

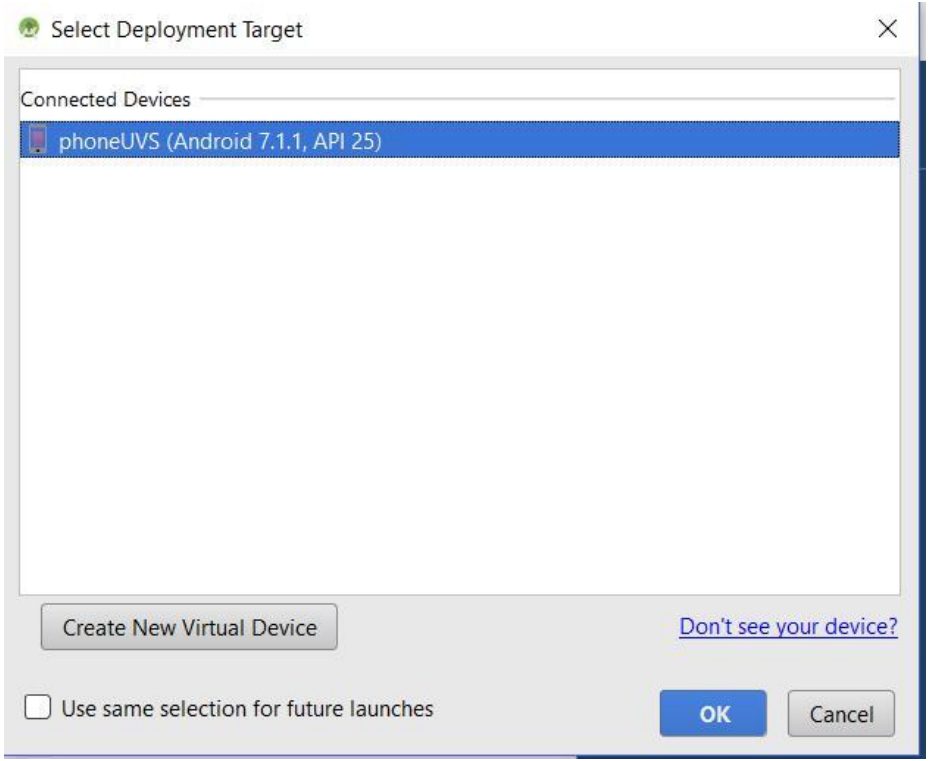
Exécution de l'application

- Pour lancer l'exécution sur l'émulateur, appuyez sur le bouton d'exécution



Exécution de l'application (1/4)

- Sélectionnez l'émulateur sur lequel vous souhaitez lancer l'application
- Vous pouvez cochez *Use same device for future launches* pour éviter d'avoir à sélectionner l'appareil à chaque lancement



Exécution de l'application (3/4)

Exécution de l'application Sur un appareil réel

- Connectez votre appareil par câble USB à l'ordinateur et installez le pilote sinécessaire
- Activez l'option de débogage USB sur l'appareil en allant dans les paramètres, sous développement ou options pour les développeurs
- Pour les versions supérieures à 4.2, cette option est cachée par défaut, pour la faire apparaitre, allez dans *Paramètres>A propos* .. et touchez *Numéro de build* sept fois.
  Retournez ensuite à l'écran *Paramètres* , vous verrez apparaitre *Options pour les développeurs*, rentrez y et activez le débogage.

Exécution de l'application (4/4)

- Lancez l'application depuis Android Studio comme précédemment. Si on vous demande de choisir l'appareil, sélectionnez *Choose a running device*, puis votre téléphone ou tablette. Android Studio installera l'application sur votre appareil et la lancera.

Titre du Slide