



# Applications Mobiles

---

420-5GM-BB - Cours 1

Pierre Prades & Mathieu Brodeur-Béliveau





## Qui suis-je?

- Pierre Prades  
[pierre.prades@bdeb.qc.ca](mailto:pierre.prades@bdeb.qc.ca)
- Enseignant au collège depuis 2019
- Coordonnateur des stages de Technique Informatique
- Consultant en conception de logiciels
- Développeur depuis 2004
- Passe-temps : voyages, films/séries, mécanique/bricolage, plein air, pêche.

A person wearing a dark pinstriped suit, white shirt, and dark patterned tie. Instead of a head, there is a large, solid black question mark. The background is a light, neutral color.

Qui êtes-vous?

# Qui êtes-vous?

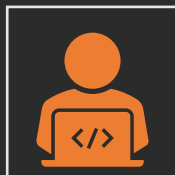
---



Nom, Prénom



Visée après votre DEC  
en informatique  
(université, travail,  
certification)



Travaux/projets  
connexes en  
informatique



Emploi (été ou autre)



Une réalisation dont  
vous êtes fier/ une  
passion

# Déroulement des cours - Notes importantes

---

- Arrivez à l'heure et soyez présent au cours c'est 70% de votre étude
- Respectez vos collègues et le professeur
- Soyez intègre - Aucune tolérance sur le plagiat
  - L'intégrité est une valeur importante
  - L'équité aussi
- Corrigez vos fautes, le français, c'est important (10% important!)



# Présentation du plan de cours

---



## Qu'entend-on par plagiat dans ce cours?



- Ré-utiliser du code fourni par quelqu'un d'autre sans l'identifier, donc s'en attribuer la propriété
- Ne pas citer ses sources
- Laisser quelqu'un utiliser ou voir son code, vous devez donc prendre des moyens raisonnables pour protéger votre code



# Plagiat suite

---

Le but d'un travail ou d'un examen est de permettre l'acquisition de compétences individuelles

Vous devez remettre un travail original

Est-ce que la collaboration est permise?

- Oui, sur le matériel vu en classe
- Ne doit pas inclure les détails d'implémentations qui eux doivent être originaux

Peut-on utiliser du code « open source » ou que l'on prend sur l'internet (ex.: [stackoverflow.com](https://stackoverflow.com))?

- Oui, mais doit être clairement identifié!!



# Détection de plagiat

---

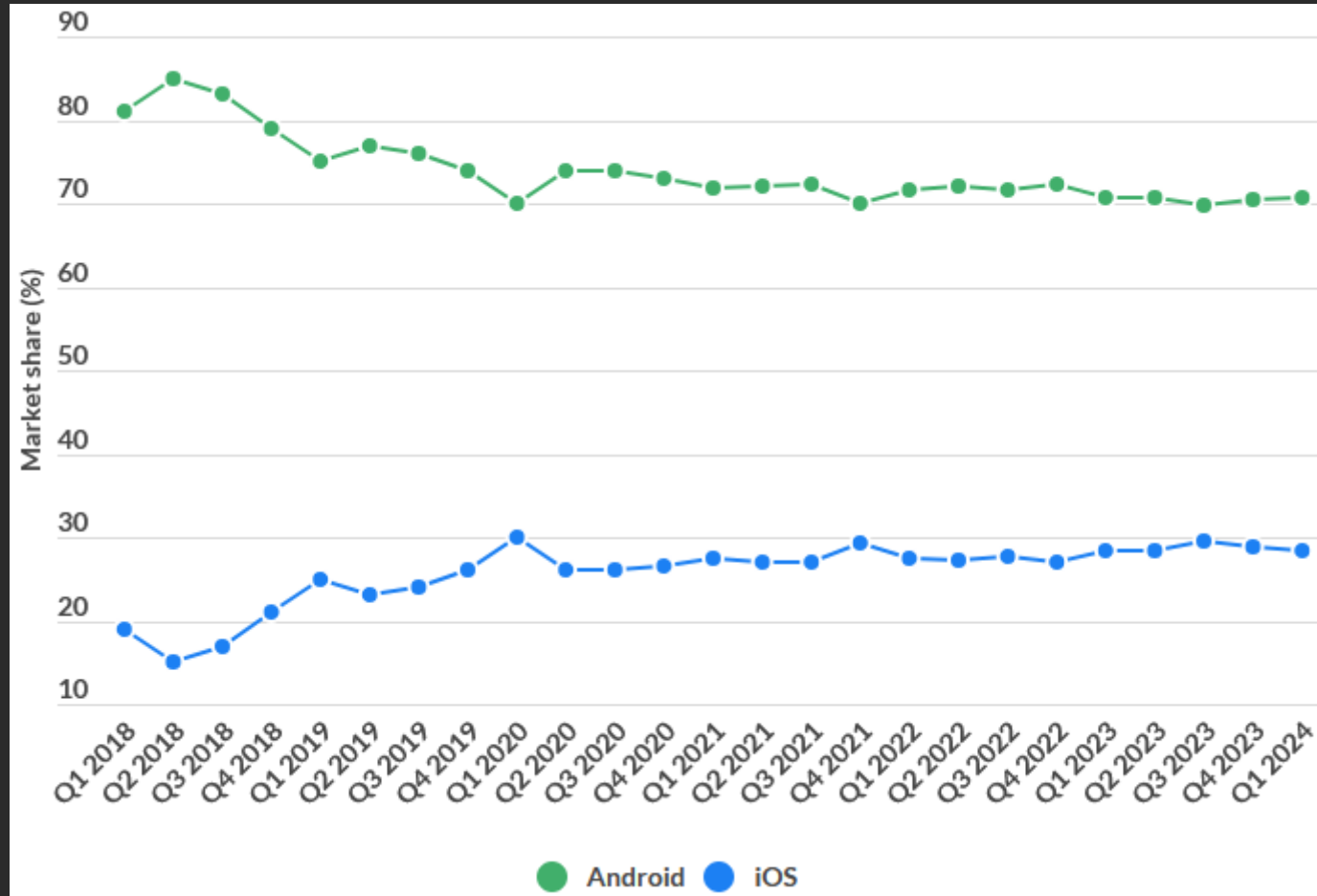
- Utilisation d'un outil de comparaison de code pour tous les TP et questions d'examen
- Modification de noms de variables et des commentaires ne font pas de votre travail un travail original
- Comparaison de l'originalité de l'implémentation (structure du code) pour tous les travaux jugés similaires

# Qu'est-ce qu'Android?

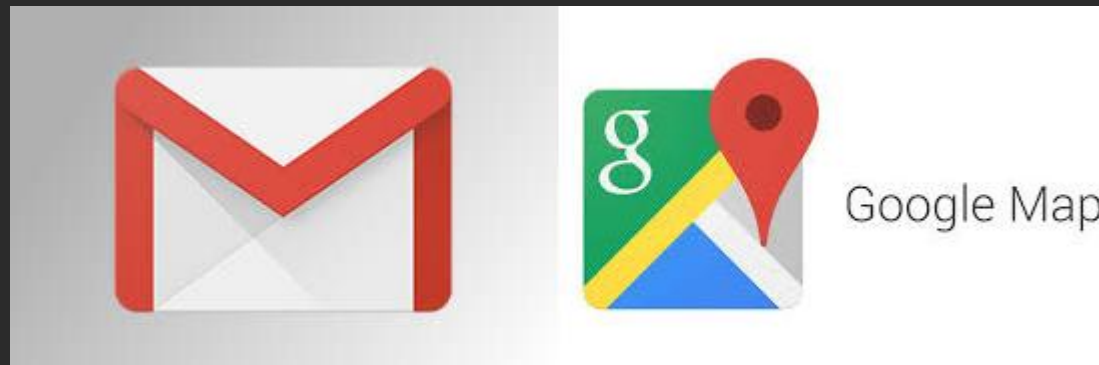
---

- Android est une plateforme mobile, construite sur un **noyau Linux**. C'est la plateforme mobile qui est présentement la plus utilisée dans le monde (environ 71% des parts de marché global). Celle-ci a été développée à l'origine par la compagnie Android, qui a été achetée par Google en **2005**.
- Le système d'exploitation Android est facilement adaptable et ouvert, c'est pourquoi une multitude d'entreprises décident d'utiliser ce système pour leurs appareils, que ce soit des **téléphones cellulaires**, des **tablettes** ou des **voitures**. De plus, le noyau Linux est sécuritaire et a depuis longtemps fait ses preuves.

- <https://infogram.com/ios-vs-android-marketshare-1hxr4zx013edo6y>



- L'environnement Android est constitué non seulement du système d'exploitation, mais aussi d'applications propriétaires Google installés sur tous les appareils, qui peuvent être lancés par ces appareils. Vous pouvez prendre pour acquis que Google Mail, Google Maps et autres applications Google sont installées sur les appareils Android.
- Pour le moment, le moteur de recherche Google vient installé dans tous les téléphones cellulaires, même ceux de Microsoft. Cependant, Google vient de perdre un procès qui va les empêcher de payer pour être installé par défaut.

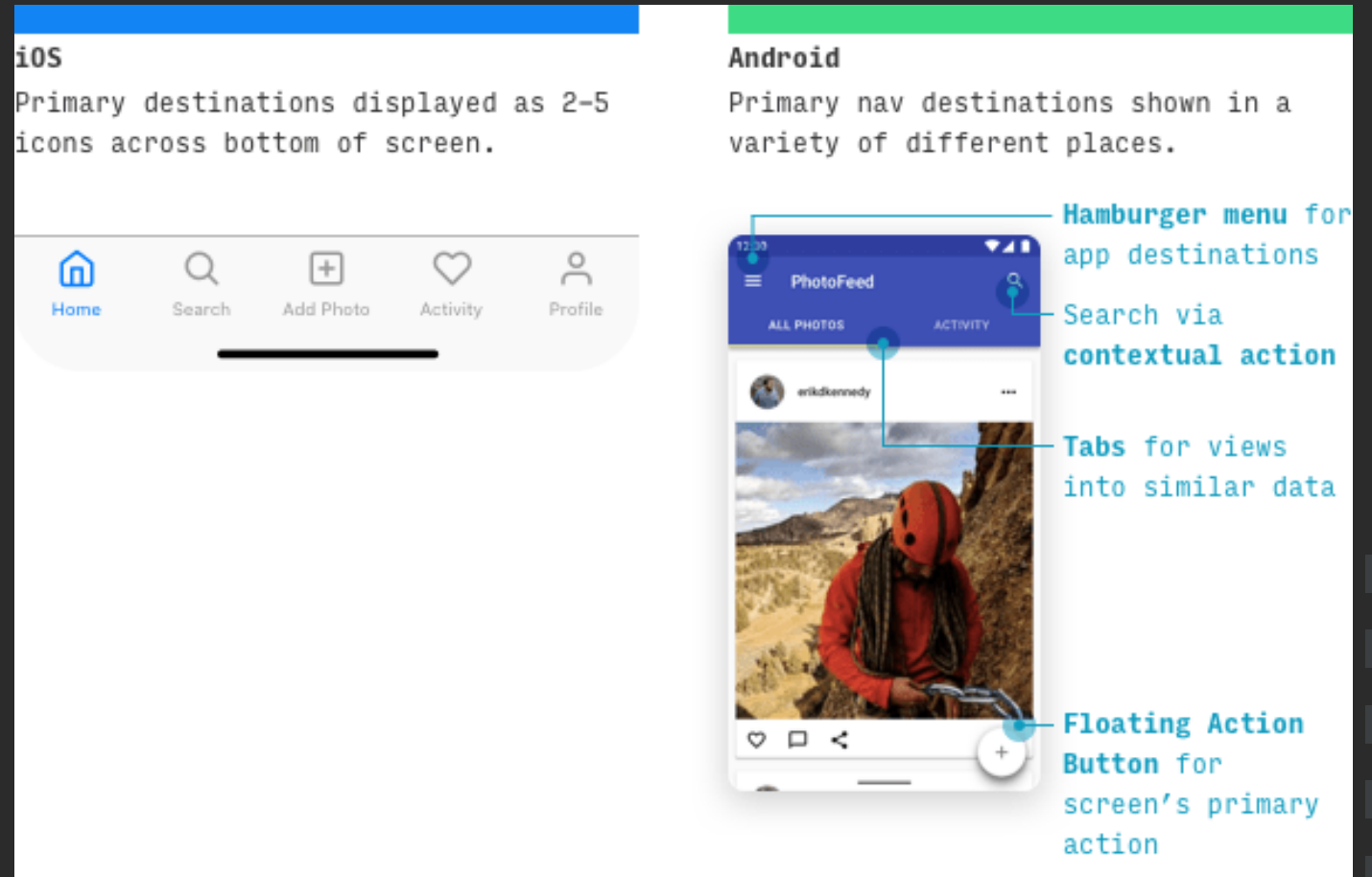


# Différences entre iOS et Android

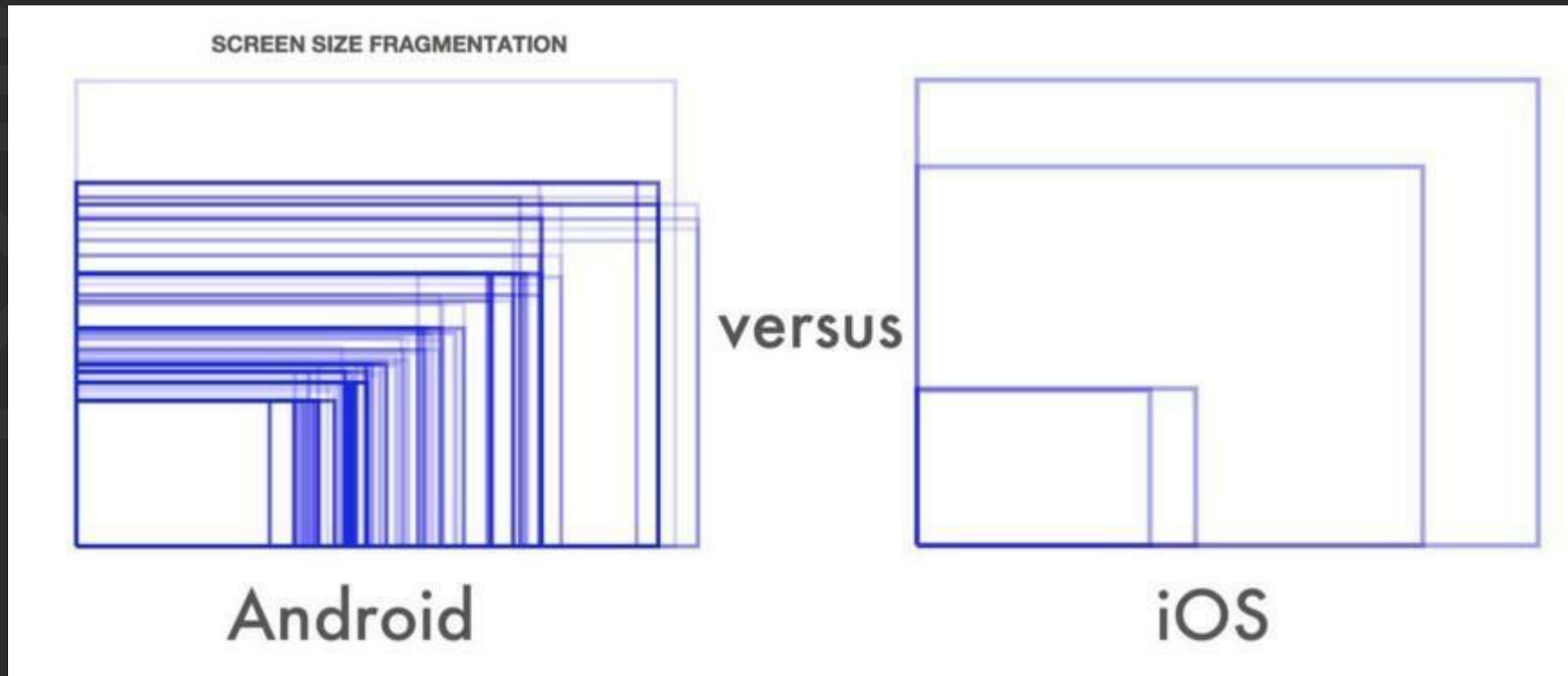
iOS	Android
Environnement beaucoup plus contrôlé. Il faut absolument s'inscrire au programme de développement d'Apple (\$99US/an).	Environnement beaucoup plus ouvert. Frais unique de \$25 pour s'inscrire au Google Play.
Il y a beaucoup plus de restrictions. Apple vérifie chaque application pour s'assurer du contenu.	Google a mis en place un système de vérification pour éviter certaines violations des règles et les malwares.
Apple reçoit une partie des revenus des achats intégrés aux applications.	Les revenus pour Google sont générés par les publicités dans ses propres applications.
Swift et Objective-C sont les seuls langages de développement pour iOS. Les libraries peuvent être dans un autre langage. Il est nécessaire d'avoir une machine sur iOS pour compiler son application.	Java, Kotlin, C, C++ peuvent être utilisés pour le développement.
Les outils de développements sont moins nombreux. Le plus populaire est XCode IDE.	Plusieurs IDEs sont disponibles. Le plus populaire est Android Studio.

# Différences entre iOS et Android

- iOS est plus simpliste et intuitif.
- Android est plus flexible et personnalisable.
- Convention de conception différent!



# Différences entre iOS et Android



# Versions d'Android

- (Vidéo) [Introducing Android](#) (avant qu'il existe)
- Depuis 2008 nous avons vu l'apparition de cellulaires qui roulent sur Android, et chaque année le nombre d'appareils et de types d'appareils différents augmente de manière significative.
- Contrairement à l'iPhone d'Apple, il y a un très grand éventail d'appareils physiques et de versions du système d'exploitation à considérer lorsqu'on décide de développer des applications sur Android.
- Plus la version d'Android est récente, plus le nombre de fonctionnalités ("*features*") qui sont offertes au développeur et à l'utilisateur est important.



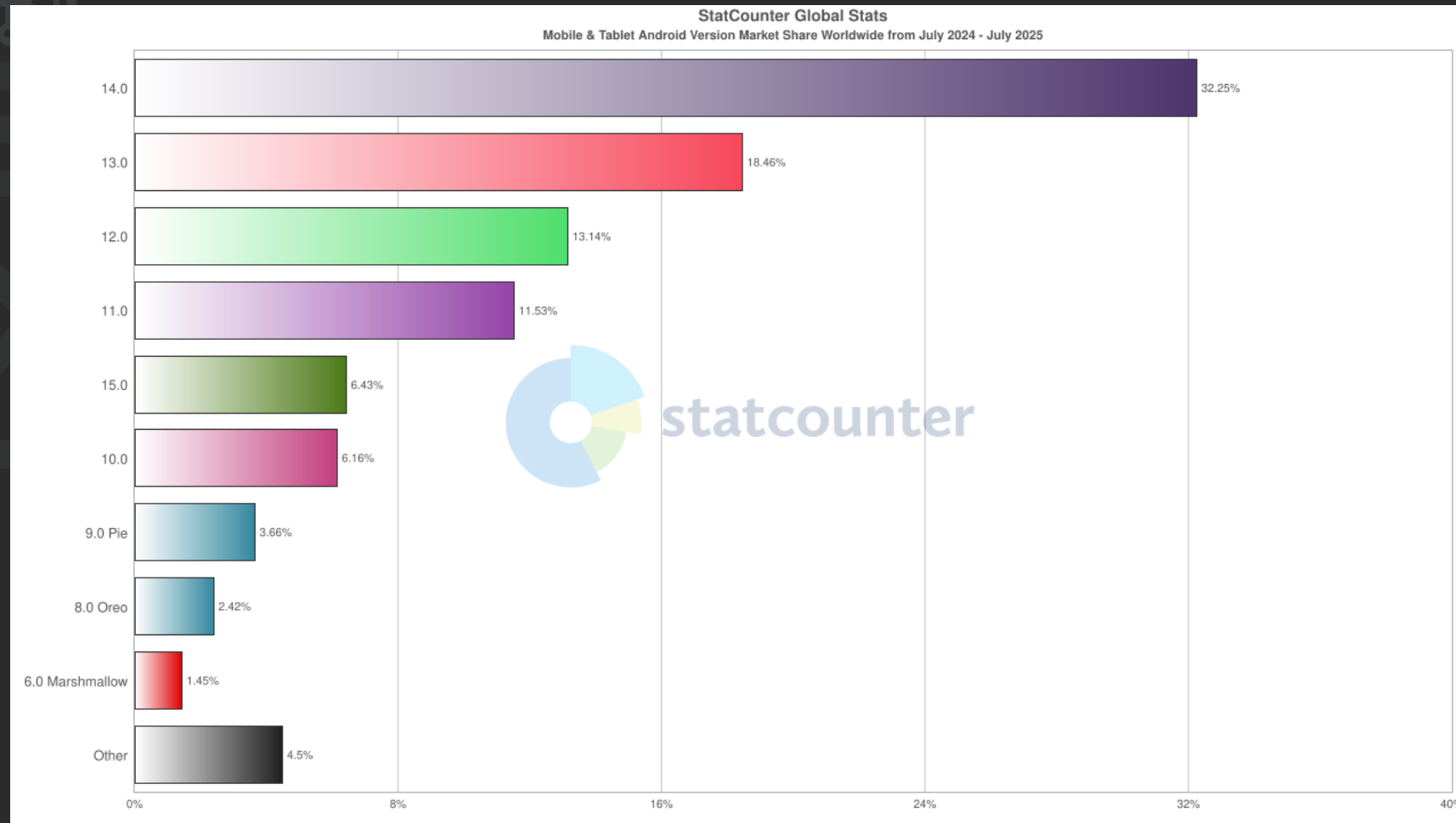


# Versions d'Android

---

- Avant de développer toute nouvelle application sur Android il faut déterminer les caractéristiques que l'on veut utiliser dans l'application à développer, et quelle version minimale d'Android permet toutes ces caractéristiques. Plus la version d'Android supportée est basse, plus le nombre d'utilisateurs potentiel est élevé, mais moins il y a de features et d'aide disponible pour le programmeur.
- Chaque version d'Android est compatible avec les versions qui le précèdent. Les versions sont donc **rétro compatible**.

- Distribution des versions en date du mois de juillet 2025  
(Source : <https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide/#monthly-202407-202507-bar>)





Évolution du GUI standard d'Android au fil des années et versions

# Coûts de développement (iOS versus Android)

---

- Parce que la part de marché d'Android est beaucoup plus importante, les app Android sont plus téléchargées que celles d'iOS.
- Cependant à cause du modèle d'affaires différent sur Android ce n'est que depuis 2017 que les applications Android génèrent plus de revenus que les applications iOS (source App Annie).

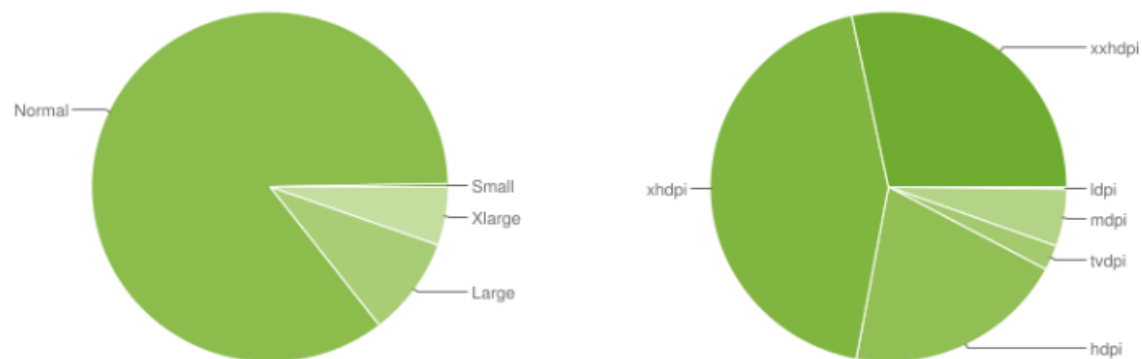
# Types d'appareils différents

---

- Lorsque vous développerez une application il faudra considérer le type d'appareil qui sera utilisé, ainsi que ses caractéristiques physiques.
- Par exemple, les tablettes, les cellulaires, montres et autres appareils ont de base une taille d'écrans différents, mais même avec le même type d'appareils il peut y avoir des disparités importantes au niveau de la résolution et de la grosseur d'écran. Même les proportions peuvent différer. Vous devez considérer et supporter ces différences si vous voulez maximiser le taux d'adoption de vos programmes par les utilisateurs.
- De plus, il faut prévoir si notre application doit supporter le mode paysage et/ou portrait, selon l'orientation de l'appareil.

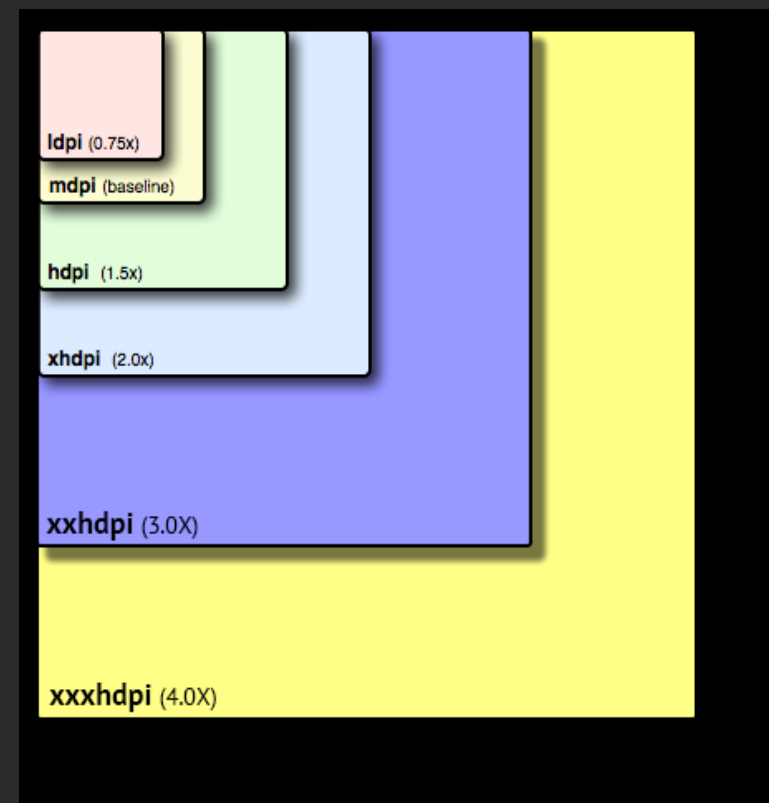
- Distribution des types d'écrans (juillet 2020)

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	0.2%				0.1%		0.3%
Normal		0.5%	0.3%	17.6%	40.7%	26.1%	85.2%
Large		1.8%	2.0%	0.6%	2.5%	2.2%	9.1%
Xlarge		3.0%		1.9%	0.5%		5.4%
Total	0.2%	5.3%	2.3%	20.1%	43.8%	28.3%	



Data collected during a 7-day period ending on July 13, 2020.

Any screen configurations with less than 0.1% distribution are not shown.






# « *Features* » principales par version d'Android

---

- Chaque version d'Android apporte son lot de nouvelles fonctionnalités, généralement incompatibles avec les versions antérieures (même s'il existe parfois des moyens de contourner ces limitations, donc d'utiliser des fonctionnalités récentes dans une version antérieure d'Android).
- Pour avoir un bon résumé des différences et ajouts entre les différentes versions de SDK d'Android, vous pouvez suivre ce lien:  
[https://fr.wikipedia.org/wiki/Historique\\_des\\_versions\\_d%27Android](https://fr.wikipedia.org/wiki/Historique_des_versions_d%27Android)
- Mais le plus simple est de directement consulter l'information à partir d'Android Studio, au moment de créer une nouvelle application :

 New Project ×

Empty Views Activity

Creates a new empty activity


Name

My Application


Package name

com.example.myapplication


Save location


C:\Users\matbr\AndroidStudioProjects\MyApplication2 

Language


Kotlin 


Minimum SDK

API 24 ("Nougat"; Android 7.0) 

 Your app will run on approximately 97.4% of devices.

[Help me choose](#)

Build configuration language 

Kotlin DSL (build.gradle.kts) [Recommended] 

Previous

Next

Cancel

Finish



## Android Platform/API Version Distribution

ANDROID PLATFORM VERSION		API LEVEL	CUMULATIVE DISTRIBUTION
4.4	KitKat	19	
5	Lollipop	21	99.7%
5.1	Lollipop	22	99.6%
6	Marshmallow	23	98.8%
7	Nougat	24	97.4%
7.1	Nougat	25	96.4%
8	Oreo	26	95.4%
8.1	Oreo	27	93.9%
9	Pie	28	89.6%
10	Q	29	81.2%
11	R	30	67.6%
12	S	31	48.6%
13	T	33	33.9%
14	U	34	13.0%

Last updated: May 1, 2024

## T

### New features

Tablet and large screen support  
 Programmable shaders  
 Color vector fonts  
 Predictive back gesture  
 Bluetooth LE Audio  
 Splash screen efficiency improvements  
 ART optimizations

### Behavior changes

OpenJDK 11 updates  
 Battery Resource Utilization  
 Media controls derived from PlaybackState  
 Permission required for advertising ID  
 Updated non-SDK restrictions

### Security and privacy

Safer exporting of context-registered receivers  
 Enhanced photo picker privacy  
 New runtime permission for nearby Wi-Fi devices  
 Exact alarms permission  
 Developer downgradable permissions  
 APK Signature Scheme v3.1  
 Better error reporting in Keystore and KeyMint

<https://developer.android.com/about/versions/13>

OK

Cancel

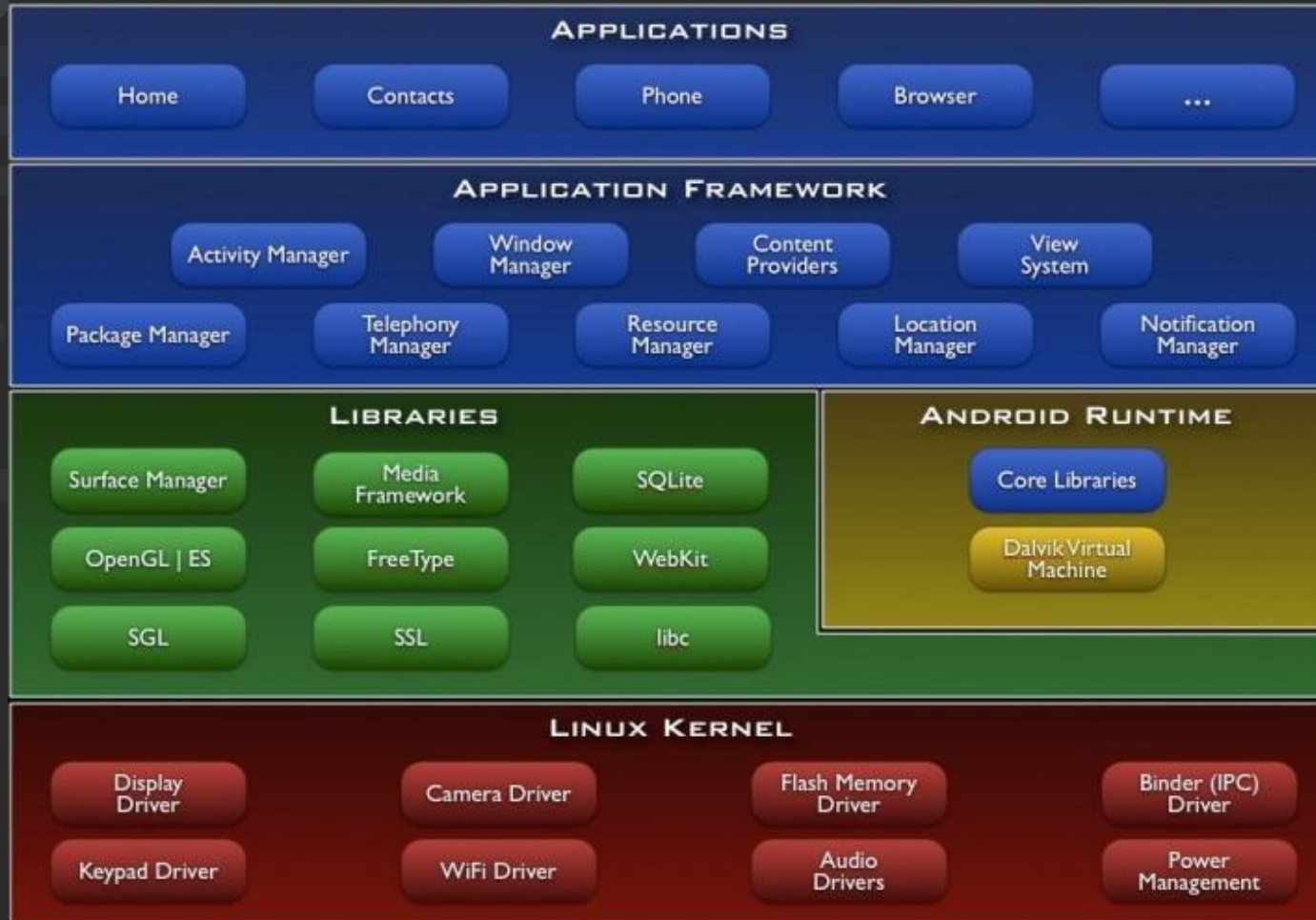


# Environnement Android – Point de vue du développeur

---



# La pile Android (Android Stack)



# Noyau Linux

---

- Le système Android est basé sur linux qui sert de couche d'abstraction matérielle.
- Le noyau Linux (Kernel) gère les périphériques, le matériel et tous les éléments de bas niveau.
- Linux a été choisi car il est sécuritaire, reconnu et fiable, gratuit et ouvert. De plus il est compatible avec une multitude de matériels différents.
- Cette couche est cachée de l'utilisateur.



# Bibliothèques (Libraries)

---

- Ce sont des outils (écrits en C et C++) qui ont été ajoutés pour aider au développement. Par exemple nous avons OpenGL qui permet de facilement programmer en 3D. SQL permet d'utiliser nativement des bases de données dans nos applications sans rien installer.
- C'est aussi ici que se trouvent les pilotes pour différents encodages audio et vidéo etc.



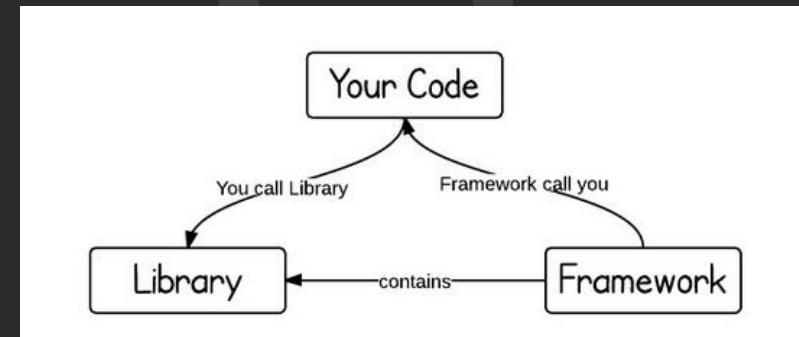
# Android Runtime (ART)



- Son principal composant est la machine virtuelle ART (anciennement Dalvik < 5.0) .
- Vous savez qu'une application Java roule sur une machine virtuelle Java (JVM). C'est grâce à cette machine virtuelle que les applications Java peuvent s'exécuter peu importe la machine et le système d'exploitation sur lequel elles fonctionnent.
- Android a fait sa propre **machine virtuelle** qu'elle a nommée « ART VM », qui est **optimisée pour Android** :
  - Elle consomme moins d'énergie
  - Elle est plus performante dans l'environnement Android que la machine virtuelle de Java car elle consomme moins de mémoire
- Comme ce n'est pas une machine virtuelle Java standard, il y a quelques différences entre du java pur et du java sous Android :
  - La machine virtuelle ART roule des fichiers ayant l'extension « .dex »
  - **Elle utilise des bibliothèques différentes de celles de java.** C'est pourquoi même si vous connaissez Java il faut revoir beaucoup de façon de faire qui sont différentes sous Android. Heureusement, les commandes de base Java demeurent inchangées : un if, else, switch, déclaration de classes, de packages, de méthodes ne change pas.

# Framework d'applications

- Ce sont des API qui ont été développées expressément POUR Android, elles sont codées en Java. C'est complètement nouveau et propre à Android, contrairement aux couches inférieures. Ça permet aux développeurs d'utiliser ces API qui simplifient énormément les actions courantes que l'on peut faire :
- Par exemple :
  - Gestion des écrans
  - Gestion des applications
  - Gestion du téléphone
  - Gestion des ressources matérielles
  - Etc.

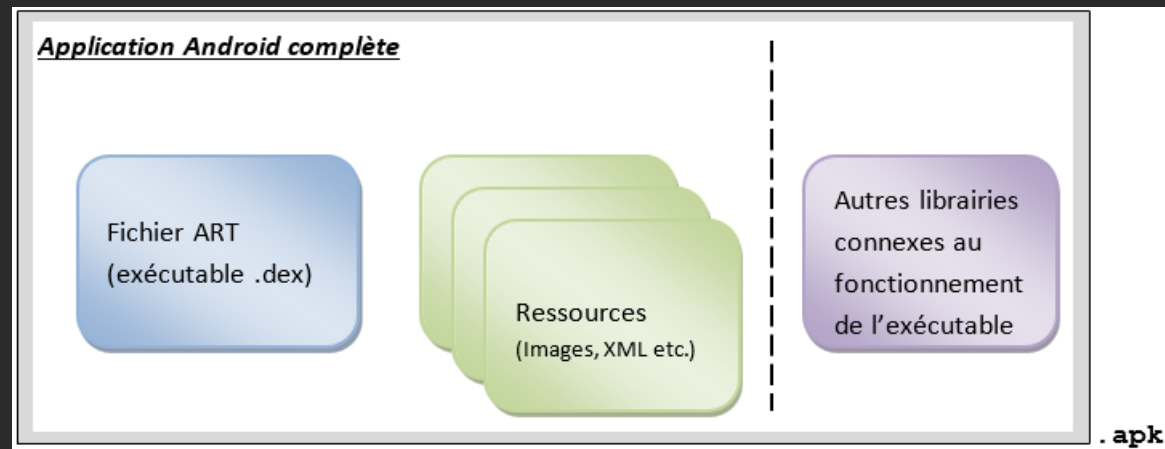




# Applications



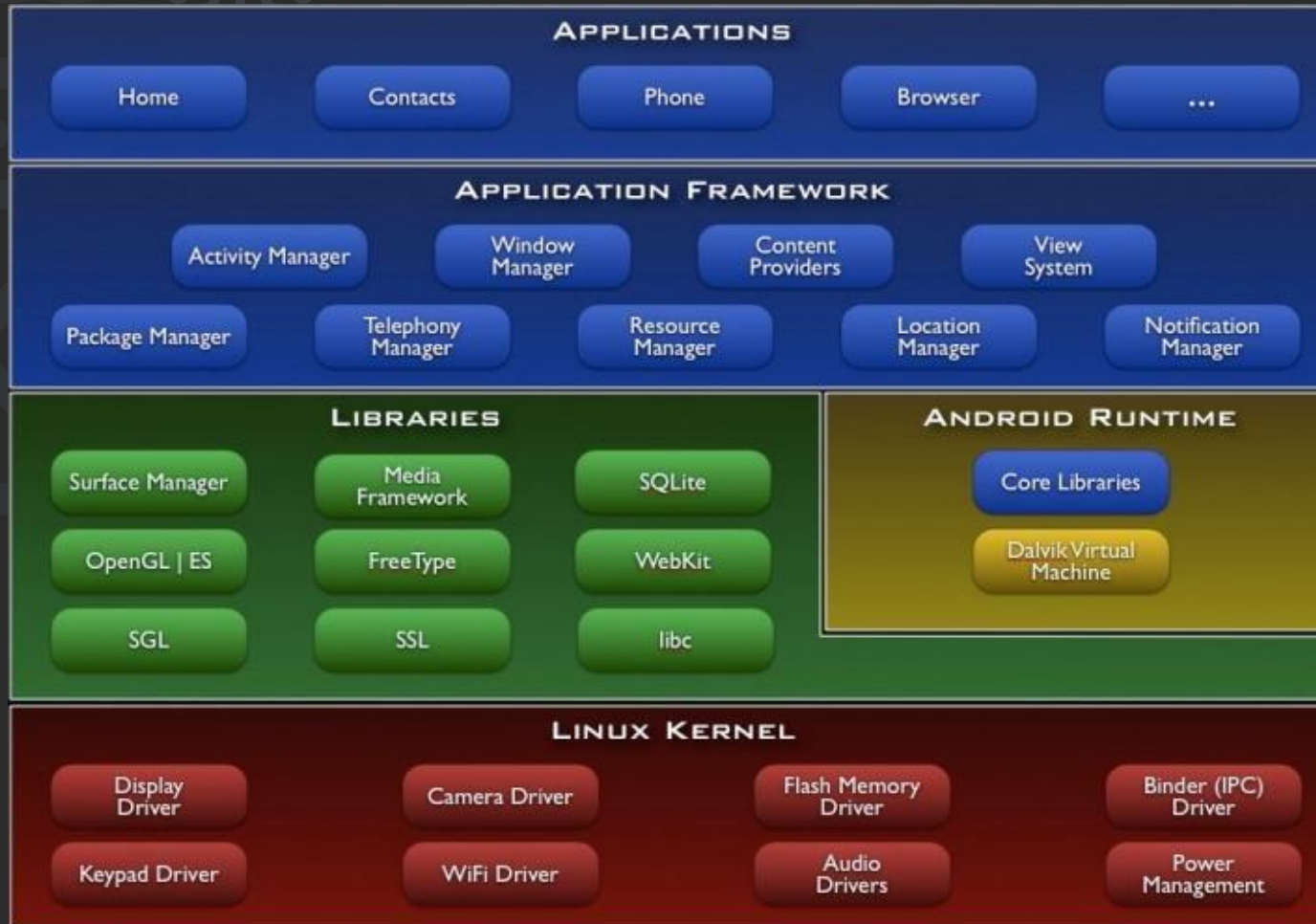
- Toute application est constituée de 2 (ou 3) parties :



- Le fichier DEX contient l'exécutable généré en Java.
- Les ressources contiennent tout ce qui est « statique », comme les images, des fichiers textes, des fichiers XML etc.
- Les autres librairies nécessaires au fonctionnement de l'application. Pourrait par exemple contenir un engin de physique pour le jeu « Angry Birds ».



# La pile Android (Android Stack)



- Angry Birds
- Gestion des périphériques
- Outils utiles (SQL, 3D)
- Abstraction matérielle invisible

# Remarque

---

- Le cours sera centré sur l'utilisation de **la machine virtuelle ART** avec des **applications Java**. C'est l'environnement par défaut de développement Android, mais ce n'est pas le seul.
- Les développeurs qui font des jeux « sérieux » et professionnels ne travaillent que très peu avec ART, et font de la programmation de plus bas niveau en langage C. Ça ne fera pas partie de la matière du présent cours.

# Abandon de Java au profit de Kotlin

---



- JetBrains, la compagnie qui produit IntelliJ, a créé le langage Kotlin.
- Il est maintenant le langage préféré de Google pour le développement d'Android.
- Java est toujours très populaire mais Google a tellement poussé pour l'utilisation de Kotlin que ce langage est maintenant désuet pour Android... Vous allez trouver beaucoup d'exemples de code en Kotlin et de moins en moins pour java.
- Le cours d'applications mobiles sera à présent donné en Kotlin.

# Environnements de développement (IDE)

---

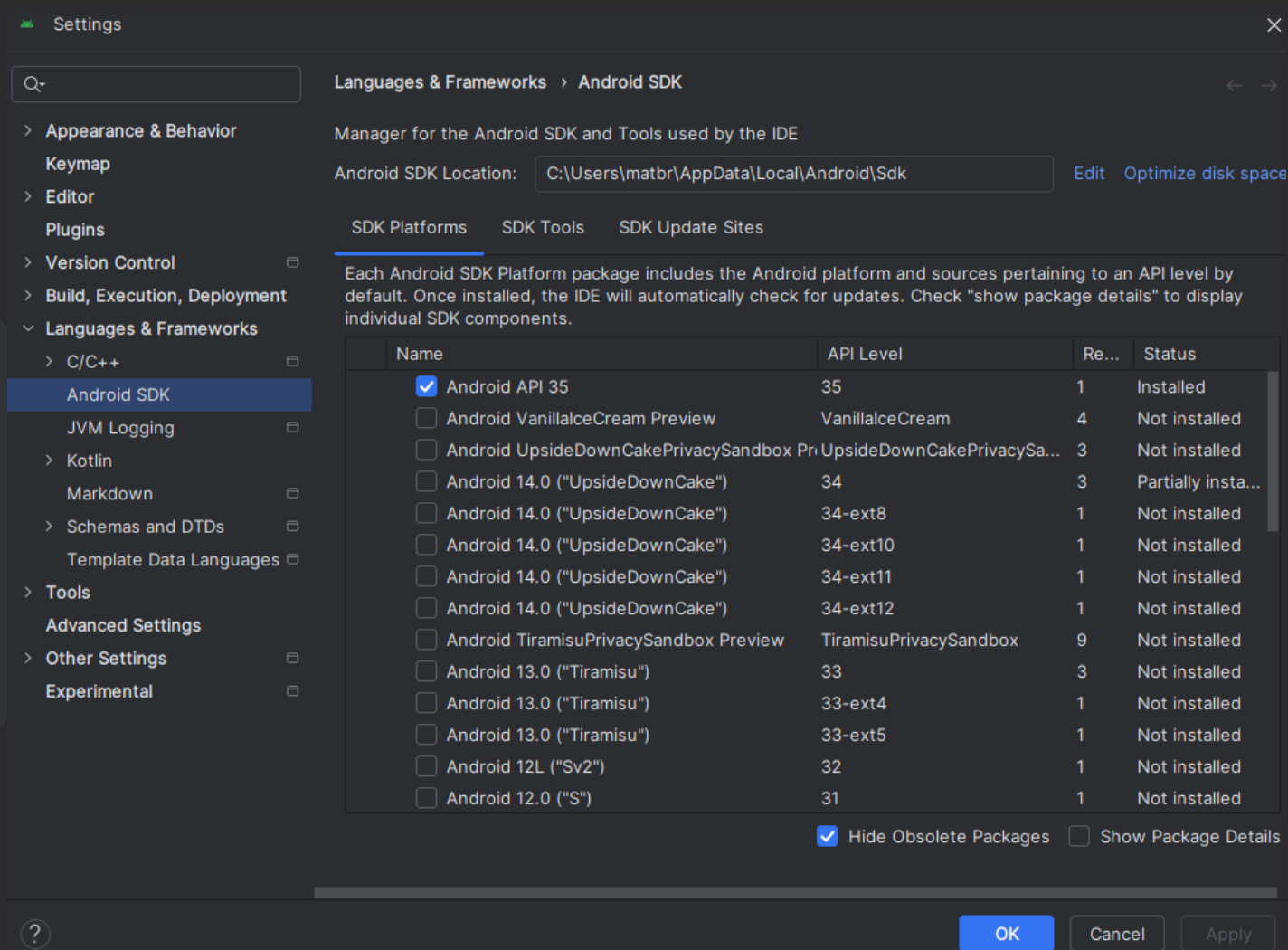
- Il existe plusieurs environnements qui permettent de développer facilement des applications sous Android, mais certains sont plus recommandés que d'autres:
  - **Android Studio (recommandé)**
  - Eclipse (package ADT)
  - IntelliJ
  - Netbeans (beaucoup moins d'intégration que les IDE précédents)
- Android Studio est l'IDE privilégié par Google comme c'est celui qui a le plus de plug-in et d'automatismes qui aident les programmeurs à être efficace, et c'est l'outil présentement développé et maintenu par Google.



# ATTENTION

---

- Les machines virtuelles sont plus lentes qu'un vrai appareil (malgré une nette amélioration de la performance des émulateurs!). Et des problèmes peuvent survenir lors de tests sur un vrai appareil.
- Vous allez trouver beaucoup d'information sur le NET. Dans le monde Android, ça bouge très vite, Il arrive que l'information trouvée soit déjà périmée. Si vous utilisez StackOverflow, vérifiez si l'information est toujours valide.
- Utilisez <https://developer.android.com/guide> comme référence si possible.



- Il suffit de cocher la version désirée et de cliquer sur OK



# Le langage Kotlin

---



# Fonctions et variables

---

- Les fonctions sont précédées du mot clé `fun`.
- Les variables du mot clé `var` et les constantes du mot clé `val`.
- Exemple:

```
fun main() {  
    val allo = 10  
    println(allo)  
}
```

- Vous pouvez voir qu'il n'y a pas de type déclaré!
- Il n'y a pas de ; à la fin des lignes non plus, mais vous pouvez les mettre



# Fonctions et variables

```
fun main() {  
    val allo = 10  
    println(allo)  
}
```

- Grâce à l'inférence de type on peut déclarer des variables sans spécifier de type, celui-ci est déterminé automatiquement.
- Ceci ne fonctionne donc pas:

```
fun main() {  
    val allo = 10  
    allo = "allo"  
    println(allo)  
}
```

- Il n'y a pas de ; à la fin des lignes non plus, mais vous pouvez les mettre

# Les patrons de chaines de caractères

---

- On peut afficher le contenu de variables dans des chaines de la façon suivante:

```
val clients = 10
println("Il y a $clients clients")
// Il y a 10 clients

println("Il y a ${clients + 1} clients")
// Il y a 11 clients
```



## Les types de base

### Categorie

### Types de base

Entiers

Byte, Short, Int, Long

Entiers non signés

UByte, UShort, UInt, ULong

Nombres réels

Float, Double

Booléens

Boolean

Caractères

Char

Chaînes de caractères

String

# Déclaration sans initialisation

- Kotlin permet de déclarer une variable sans l'initialiser en spécifiant son type après « : »
- Il faut cependant lui attribuer une valeur avant d'y accéder.
- Exemple:

// Variable déclarée sans initialisation

**val** valeur: Int

valeur = 3

// Variable typée explicitement et initialisée

**val** mot: String = "hello"

# Les collections

---

- Chaque collection peut être « mutable » (modifiable) ou en lecture seule.

Type de collection	Description
Listes	Collection ordonnée d'éléments
Sets	Collection non ordonnée d'éléments uniques
Maps	Ensemble de paires clé-Valeur où les clés sont uniques et ne referent qu'à une valeur

# Listes

---

- Pour créer une liste non modifiable utilisez la fonction `listOf()`
- Pour créer une liste modifiable utilisez la fonction `mutableListOf()`

// Liste non modifiable

```
val readOnlyShapes = listOf("triangle", "carré", "cercle")
```

// Liste modifiable avec déclaration explicite

```
val shapes: MutableList<String> = mutableListOf("triangle", "carré", "cercle")
```

# Accéder aux éléments

- Les listes sont ordonnées donc pour accéder à un élément d'une liste, utilisez l'opérateur d'accès indexé [].
- Pour obtenir le premier ou le dernier élément d'une liste, utilisez respectivement les fonctions `.first()` et `.last()`
- Pour obtenir le nombre d'éléments dans une liste, utilisez la fonction `.count()`
- Pour vérifier qu'un élément est dans une liste, utilisez l'opérateur `in`

```
val readOnlyShapes = listOf("triangle", "carré", "cercle")
println("Le premier élément de la liste est: ${readOnlyShapes[0]}")
// Le premier élément de la liste est: triangle

println("Le premier élément de la liste est: ${readOnlyShapes.first()}")
// Le premier élément de la liste est: triangle

println("Cette liste contient ${readOnlyShapes.count()} éléments")
// Cette liste contient 3 éléments

println("cercle" in readOnlyShapes)
// true
```

# Les structures de contrôle

```
if (a > b) {  
    max = a  
}
```

- If/else
- C'est comme en java, par contre pas d'opérateur ternaire « ?: »
- when est l'équivalent de switch

```
val obj = "Hello"  
  
val result = when (obj) {  
    "1" -> "One"  
    "Hello" -> "Greeting"  
    else -> "Unknown"  
}  
println(result)  
// Greeting
```

```
val obj = "1"  
  
val result = when (obj) {  
    "1" -> "One"  
    "Hello" -> "Greeting"  
    else -> "Unknown"  
}  
println(result)  
// One
```

```
val obj = "Allo"  
  
val result = when (obj) {  
    "1" -> "One"  
    "Hello" -> "Greeting"  
    else -> "Unknown"  
}  
println(result)  
// Unknown
```



# for in

---

- For utilise des plages de valeur:

```
for (number in 1..5) {  
    // number is the iterator and 1..5 is the range  
    print(number)  
}  
// 12345  
val cakes = listOf("carrot", "cheese", "chocolate")  
  
for (cake in cakes) {  
    println("Yummy, it's a $cake cake!")  
}  
// Yummy, it's a carrot cake!  
// Yummy, it's a cheese cake!  
// Yummy, it's a chocolate cake!
```

# While

---

- While et do/while ont la même syntaxe qu'en java.

```
while (x > 0) {  
    x--  
}
```

- Faire les exercices du site :

<https://kotlinlang.org/docs/kotlin-tour-welcome.html>